بسم الله الرحمن الرحيم

جامعة السّودان المفتوحة

برنامج علوم الحاسوب وتقنية المعلومات

أساليب البرمجة (1)

رمز المقرر ورقمه: حسب 302

إعداد المادة العلمية: **أ. د. محمد أحمد الأفندي**

تصميم تعليمي: د. نهى مدثر البحيري

التنضيد الطباعي: أ. مشاعر حسن بكري

التصميم الفني: أ. منى عثمان النقة

منشورات جامعة السودان المفتوحة، الطبعة الأولى 2004

جميع الحقوق محفوظة لجامعة السودان المفتوحة، لا يجوز إعادة إنتاج أيّ جزء من هذا الكتاب، وبأيّ وجه من الوجوه، إلاّ بعد الموافقة المكتوبة من الجامعة.

مقدمة المقرر

الحمد لله رب العالمين، والصلاة و السلام على خاتم الانبياء معلم البشرية؛ أما بعد.
مرحباً بك عزيزي الدارس، إلى مقرر (أساليب البرمجة) الذى يعتبر العمود الفقرى لدراسات البرمجة بمختلف اللغات وقد رأينا في هذا المقرر أن تكون لغة C++ هي النموذج الذي ندرس من خلاله أساليب البرمجة المختلفة سائلين المولى أن يجد فيه الدارس كل القواعد الأساسية التي تساعده في تعلم أي لغة برمجة.

الحاسوب جهاز إلكتروني مرن يمكن "تطويعه" أو "برمجته" لأداء الكثير من المهام في شتى المجالات. عملية "التطويع" هذه أو "البرمجة" تتم من خلال الأوامر التي يفهمها الحاسوب. كثير من التطبيقات الحاسوبية التي نراها اليوم ما هي إلا سلسلة من الأوامر التي يقوم الحاسوب بتنفيذها الواحد تلو الآخر.

في بداية استخدام الحاسوب كان المختصون يكتبون الأوامر مباشرةً بالمصطلحات التي يوفرها الجهاز، وهو ما يعرف في مصطلح الحاسوبيين ب "لغة الآلة". "لغة الآلة" في مجملها تتكون من أرقام "ست عشرية" أو "ثنائية" لها مدلول خاص عند الحاسوب. إذا أراد المختص أداء عملٍ ما باستخدام الحاسوب فإنه يقوم أولاً بصياغة أو تصميم "البرنامج" الذي يقود الحاسوب إلى أداء المهمة، ثم يقوم بتنزيل البرنامج على الجهاز ومتابعة تنفيذه. البرنامج ما هو إلا سلسلة من الأوامر التي تؤدي إلى إنجاز عملٍ ما على الحاسوب.

لم يمض وقتٌ طويل على إستخدام لغة الآلة حتى اتضح للمختصين أنهم بحاجة إلى أسلوبٍ أفضل للتعامل مع الحاسوب، ذلك أن البرامج المكتوبة بلغة الآلة يكتنفها الكثير من الغموض و التعقيد ويصعب مراجعتها و تدقيقها. من هذا المنطلق ابتكر المختصون أسلوباً آخر أطلقوا عليه "لغة التجميع". الجديد في لغة التجميع أن المصطلحات الرقمية استبدلت بمصطلحات "حرفية" تسهل قراءتها و مراجعتها. و بما أن الحاسوب لايفهم هذه المصطلحات الحرفية، فقد تم تصميم برامج وسيطة تقوم بترجمة هذه المصطلحات إلى الصورة الرقمية التي يفهمها الحاسوب.

بالرغم من أن لغة التجميع افضل بكثير من لغة الآلة، إلا أن المبرمجين ظلوا يعانون من مشكلة أخرى، هي أن معظم أوامر لغة التجميع لا تقبل إلا معاملاً واحداً أو معاملين على الأكثر. وهذا يعني أن المعادلات الجبرية و الأوامر الطويلة لابد من تجزئتها وتقسيمها إلى

عمليات بسيطة في برامج التجميع. وبدأ التساؤل: هل بالإمكان استحداث أسلوب برمجي يسمح بكتابة المعادلات الجبرية و الأوامر الطويلة مباشرةً دون تجزئتها؟ وكانت الإجابة على هذا التساؤل هو ظهور لغات البرامج "عالية المستوى" و أولاها لغة "فورتران" المشهورة في أوساط المهندسين. لغات البرمجة عالية المستوى تسمح بكتابة المعادلات الجبرية و الأوامر الطويلة مباشرةً، ثم تستخدم "ترجماناً" (Compiler) لتحويل هذه الأوامر إلى لغة الآلة التي يفهمها الحاسوب.

لغات البرمجة عالية المستوى نجحت نجاحاً باهراً وتطورت تطوراً سريعاً في غضون سنوات قليلة. فبعد Fortran ظهرت لغة COBOL ثم ALGOL 60، ثم ALGOL 68، ثم Pascal، Aida، C، C++، Java ....إلخ.

من أهم التطورات في تاريخ لغات البرمجة هو ظهور أسلوب "البرمجة التركيبية" (Structured Programming) كأسلوب رائد لتصميم البرامج. ظهرت بوادر هذا الأسلوب مع لغة ALGOL 60 في الستينات ثم ساد الساحة البرمجية بلا منازع حتى منتصف الثمانينات. أهم خصائص هذا الأسلوب هو النظرة التركيبية لتطوير البرامج (أي برنامج يتركب من أجزاء و مكونات يمكن تصميم كل منها على حدة ثم تجميعها لتعطي البرنامج الأب) و الطريقة الإنحدارية لتصميم البرامج.

في منتصف الثمانينات ظهر اسلوب آخر للبرمجة بدأ يحل تدريجياً محل البرمجة التركيبية. هذا الأسلوب هو أسلوب البرمجة "الموضوعية" أو "الكائنية" (Object Oriented Programming). الفكرة المحورية لهذا الأسلوب هو أننا في معظم التطبيقات نتعامل مع "كائنات" أو "مواضيع" من واقع الحياة. هذه المواضيع يمكن تمثيلها مباشرةً في برامجنا. مثلاً عند كتابة برنامج لتسجيل الطلاب في جامعة، فإنه من المناسب أن نستحدث كائنات برمجية تمثل الطلاب، الأساتذة، المقررات، القاعات ...إلخ. هذه الكائنات البرمجية تعطي مواصفات الكائنات الفعلية وتلحق بها العمليات و الدوال التي تؤثر على هذه المواصفات. أسلوب البرمجة الكائنية هو أسلوب العصر الذي تبنته معظم اللغات الحديثة مثل C++، Java و C Sharp.

الشيء الذي يميز C++ عن مثيلاتها هي أنها تجمع بين أسلوبي البرمجة التركيبية والبرمجة الكائنية ويمكن استخدامها للغرضين معاً. لغة C++ تم استحداثها في منتصف الثمانينات من قبل شركة "إي تي آن تي" وانتشرت بصورة سريعة وتم اعتماد الصورة القياسية لها

من قبل الهيئات العالمية المختصة. الكثير من الجامعات العالمية الآن تدرس هذه اللغة كلغة أساسية في برامج علوم الحاسوب و المعلومات.

هذا المقرر الذي بين يديك يعتبر مدخلاً مبسطاً للبرمجة التركيبية بلغة C++. أسلوب البرمجة الكائنية سنتعرض له بإذن الله في كتابٍ لاحق.

الأهداف العامة للمقرر

يتوقع منك عزيزي الدارس بعد دراستك لهذا المقرر أن:

- تتعرَّف مكونات الحاسوب ولغة الآلة.
- تتعرَّف المراحل العلمية لتطوير البرامج بصورة عملية بإستخدام نظام C++ Visual.
- تصميم برامج بلغة C++ لحل مسائل حسابية.
- تميز بين أساليب التفريع و الاختبار و التكرار.
- تحفظ البيانات في ملفات متتالية بسيطة.
- تصميم مصفوفات وإجراء العمليات الحسابية عليها باستخدام لغة C++.
- تتعرف على عناصر الدوال في C++ و كيفية تصميمها و إعداد البرامج لها.
- تستوعب أسلوب البرمجة التركيبية وتستخدمه في حل المسائل المعقدة.

محتويات المقرر

الوحدة	الموضوع	الصفحة
1	مدخل عام	1
2	مدخل إلى لغة أساسيات البرمجة والهيكل العام للبرامج C++	29
3	التحكم في مسار البرنامج : التفريع والتكرار	61
4	الملفات المتتالية البسيطة	99
5	المصفوفات	111
6	البرمجة التركيبية: الدوال	129
7	السجلات	157
8	المؤشرات	179
9	خوارزميات البحث والترتيب	203


الوحدة الاولى
مدخـــل عام

محتويات الوحدة

الموضوع	الصفحة
المقدمة	**4**
تمهيد	4
أهداف الوحدة	4
1. التركيب العام للحاسوب	**5**
2. كيفية إجراء العمليات داخل الحاسوب	**8**
3. الصورة الواقعية: لغة الآلة	**10**
4. برامج التجميع و المترجمات	**11**
5. المراحل العملية لتطوير البرامج	**13**
6. حل المسائل بإستخدام الحاسوب	**21**
7. دعامات أساسية	**21**
8. مراحل حل المسألة	**22**
1.8 مرحلة التحليل وتحديد المطلوب	22
2.8 مرحلة التصميم	22
3.8 مرحلة توليد البرامج	22
4.8 مرحلة الاختبار و التدقيق	23
5.8 مرحلة المتابعة	23
9. اسلوب مبسط لحل المسائل البسيطة	**24**
الخلاصة	**25**
لمحة مسبقة عن الوحدة التالية	**26**
مسرد المصطلحات	**26**
المصادر والمراجع	**27**

المقدمة

تمهيد

مرحباً بك عزيزي الدارس في الوحدة الاولى من مقرر أساليب البرمجة، في هذه الوحدة نسعى لمراجعة معلوماتك التى اكتسبتها من مقرر الحاسوب السابق وهو مبادئ علوم الحاسوب وربط هذه المعلومات وتوظيفها في عملية البرمجة.

برغم ما يشاع عن الحاسوب من إمكانات و مقدرات فإنه لا يعدو في واقع الحال أن يكون آلةً صماء، ضعيفة الإدراك، تنفذ ما يملى عليها من أوامر حرفياً. ما يقوم به الحاسوب يعكس مهارة "الإنسان" الذي يبرمجه و يقدم له التوجيهات. الحاسوب لا يبتكر الحلول و إنما يقوم بتنفيذ الأوامر التي تملى عليه في شكل خطوات مفصلة و توجيهات دقيقة. هذه الخطوات المفصلة و التوجيهات الدقيقة تسمى "برنامجاً" في مصطلح الحاسوبيين.

برمجة الحاسوب تشبه إلى حد كبير عملية ترويض الحيوانات أو التعامل مع الصم من حيث أن الطرف الآخر يحتاج إلى جهد كبير في التخاطب و التفاهم. المبرمج عليه أن يفهم بدقة طبيعة الآلة التي يتعامل معها حتى يتسنى له إملاء الأوامر عليها بالصورة التي تفهمها.

تحتوي هذه الوحدة على عرض المكونات التي لها علاقة بتنفيذ البرامج وطريقة عملها من خلال التعرف على لغة الآلة وكيفية اجراء العمليات داخل الحاسوب. أيضاً تشتمل الوحدة على المراحل العملية لتطوير البرامج وتدريبات عملية في الحاسوب.

أهداف الوحدة

عزيزي الدارس؛

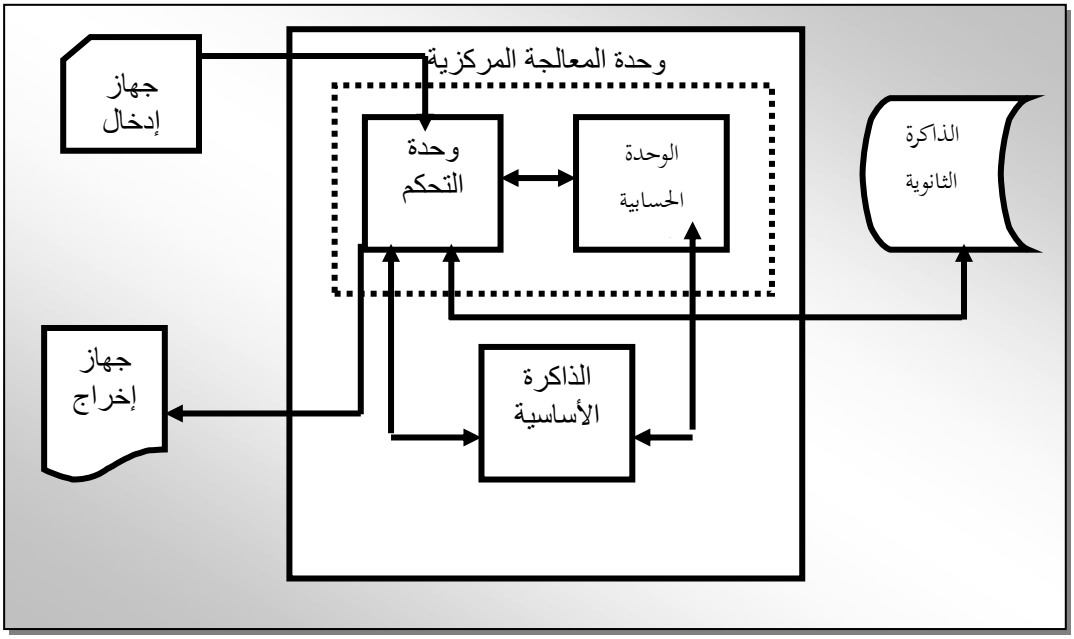
بعد فراغك من هذه الوحدة ينبغى عليك أن تكون قادراً على أن:

- **توظف** طبيعة الحاسوب و إمكاناته فى تصميم البرامج
- **تميز** بين لغة الآلة ولغة التجميع.
- **تطور** برامج C++ بإستخدام نظام Visual C++ .
- **تتبع** منهج عملي لتحليل وتصميم حل للمسائل بإستخدام الحاسوب.

1. التركيب العام للحاسوب

الشكل (1)

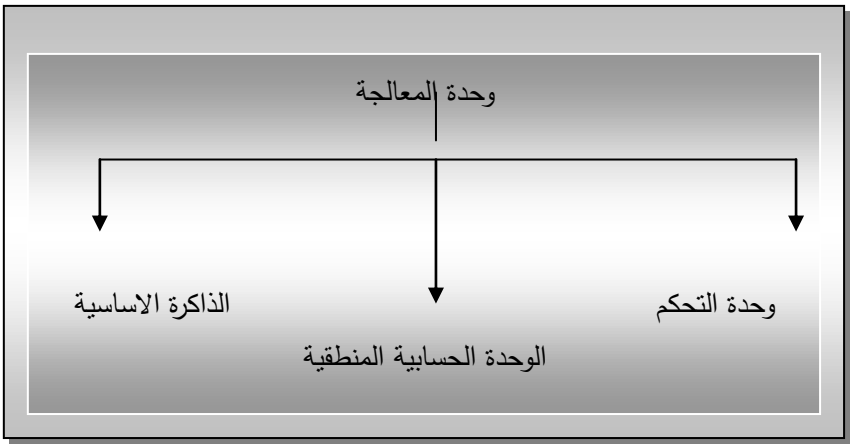
المكونات الأساسية لجهاز حاسوب نموذجي



كما هو واضح من الشكل أعلاه فإن الحاسوب يتكون من:

- **وحدة المعالجة المركزية CPU** التي تمثل قلب الحاسوب وتتكون بدورها من ثلاث وحدات هي:

1) وحدة التحكم.
2) الوحدة الحسابية المنطقية
3) الذاكرة الأساسية.

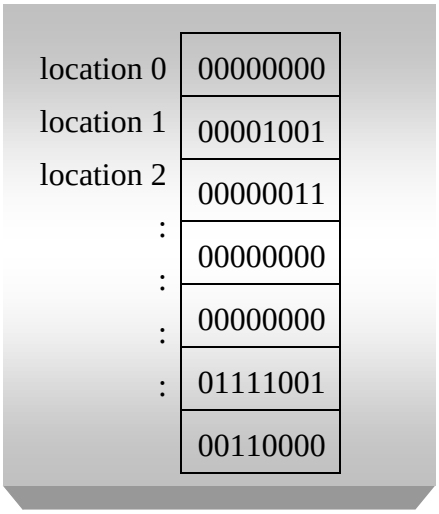


معظم الحاسبات المتداولة اليوم مبنية على فكرة "البرنامج المخزون"، و التي تقتضي أن يخزن البرنامج و البيانات المتعلقة به في الذاكرة قبل بدء المعالجة. الوظيفة الأساسية لوحدة التحكم هي إحضار أوامر البرنامج المخزون من الذاكرة ثم تحليلها وبدء تنفيذها. بالإضافة إلى ذلك تقوم وحدة التحكم بالتنسيق بين إجراء الحاسوب المختلفة لتمكنها من أداء دورها بالصورة المطلوبة. أما الوحدة الحسابية المنطقية فتضطلع، كما يدل اسمها، بإجراء العمليات الحسابية والمنطقية البسيطة والتي لا تتطلب أكثر من معاملين في المرة الواحدة. يدخل في ذلك عمليات الجمع، الطرح، الضرب، القسمة والمقارنة. في الفقرات أدناه نناقش كيف يتم إجراء هذه العمليات.

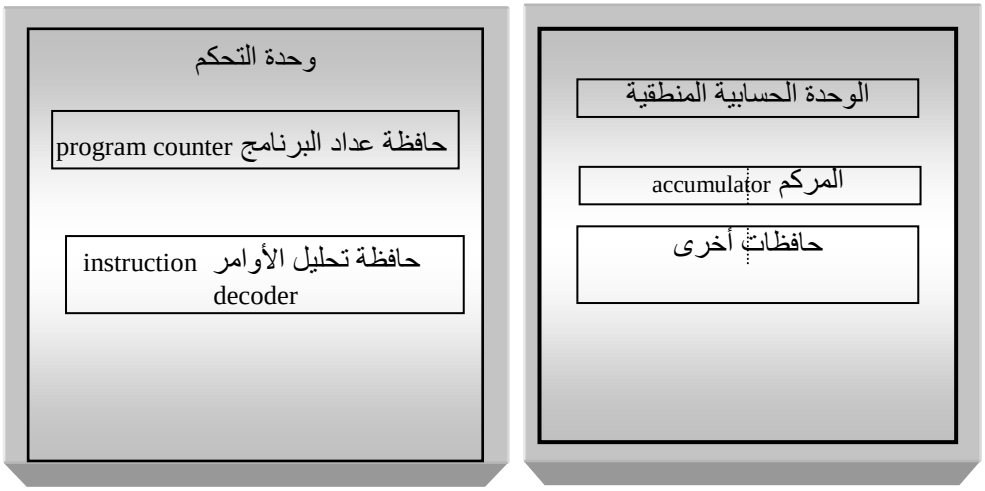
وحدات الإدخال و الإخراج مثل لوحة المفاتيح والشاشة والطابعة وغيرها من الوحدات التي تستخدم في إدخال البيانات أو إظهار النتائج.

الذاكرة الثانوية أو الذاكرة الدائمة مثل الأقراص الصلبة و الأقراص المرنة، وهي أبطأ بكثير من الذاكرة الأساسية إلا أنها تمتاز عنها باستمرارية تخزين المعلومات حتى بعد انفصال التيار الكهربائي.

الذاكرة الأساسية تتكون من مجموعة من الخانات التخزينية، كل خانة تتكون من مجموعة من الوحدات الثنائية التي تحتمل إحدى حالتين: إما صفر أو واحد. عدد الوحدات الثنائية في كل خانة تختلف حسب نوع الحاسوب، ولكن الشائع في مجال الحاسبات الشخصية أن كل خانة تتكون من ثمانية وحدات ثنائية. كل خانة تعرف بعنوان خاص (رقم) يستخدم في الوصول إليها للتخزين أو الاسترجاع:



تجدر الإشارة هنا إلى أن كل من وحدة التحكم والوحدة الحسابية المنطقية تحتوي على خانات خاصة من الذاكرة السريعة تسمى الحافظات registers. هذه الحافظات تشبه خانات الذاكرة الأساسية في الحجم و إمكانية الحفظ إلا أنها ليست لها عناوين و إنما تعرف بأسماء خاصة. أشهر انواع هذه الحافظات في وحدة المعالجة المركزية هي الحافظة التي تعرف بالمركم accumulator تستخدم بكثرة في إجراء العمليات الحسابية كما سنرى في الأمثلة أدناه.



2. كيفية إجراء العمليات داخل الحاسوب

عزيزي الدارس إن فكرة البرنامج المخزون التي ذكرناها في الفقرات السابقة تقتضي أن تخزن أوامر البرنامج و البيانات الابتدائية في الذاكرة قبل بدء التنفيذ. و عليه فإن الخطوة الأولى في تنفيذ أي برنامج هي تحميله في الذاكرة الأساسية.

بعد تحميل البرنامج في الذاكرة الأساسية تقوم وحدة التحكم بتحليل أوامر البرنامج الواحد تلو الآخر بعد تحميل الأمر في حافظة خاصة داخل وحدة التحكم تسمى حافظة التحليل. عملية التحليل تنتج عنها معرفة نوع الأمر وتحديد مكانه وكيفية تنفيذه.

الأوامر التي تتعامل معها وحدة المعالجة المركزية في غاية البساطة ولاتخرج في الغالب الأعم عن الأنواع التالية:

- **عمليات نقل بيانات من منطقة إلى أخرى داخل وحدة المعالجة المركزية. مثلاً تحميل بيانات من الذاكرة إلى وحدة المعالجة المركزية.**
- **عمليات حسابية بسيطة مثل الجمع و الطرح والضرب.**
- **عمليات التفريع و المقارنة.**
- **عمليات الإزاحة الجانبية لبعض المعاملات داخل المسجلات.**

الجدول أدناه يعطى أمثلة لبعض الأوامر التي تستخدم في كثيرٍ من الحاسبات.

جدول (1)

المقصد	الأمر
خزن قيمة معينة في الذاكرة	STORE
حمِّل قيمة معينة من الذاكرة إلى المركم	LOAD
أضف قيمة معينة إلى ما هو موجود في المركم، ضع الناتج في المركم	ADD
اطرح قيمة معينة مما هو موجود في المركم، ضع الناتج في المركم	SUB
اضرب قيمة معينة بما هو موجود في المركم، ضع الناتج في المركم	MULT
اقسم ما هو موجود في المركم على المعامل المصاحب،ضع الناتج في المركم	DIV
أزح القيمة الموجودة في المركم إلى اليسار عدد من الثنائيات	SHL
أزح ما هو موجود في المركم إلى اليمين عدد من الثنائيات	SHR
فرِّع إذا كان الناتج صفر	BZ
فرِّع إذا كان الناتج أكبر من صفر	BG
فرِّع إذا كان الناتج أصغر من صفر	BL

دعنا الآن عزيزي الدارس نأخذ رحلة مع الحاسوب عبر بعض العمليات البسيطة لنفهم كيف يقوم بدوره. هب أننا طلبنا من الحاسوب أن يقوم بحساب قيمة X حسب العلاقة:

$$X = A * B + C$$

حيث A=12، B=5، C=9.

في البداية لابد من تخزين الأوامر التي تنجز هذه العملية و قيم A,B,C داخل ذاكرة الحاسوب. بما أن الحاسوب لا يستطيع إنجاز العمليات المركبة، فلابد من تجزئة العملية الجبرية أعلاه إلى عمليات بسيطة ذات معامل الواحد. تجزئة التعبير الجبري أعلاه تقود إلى سلسلة العمليات المكافئة أدناه:

LOAD A	حمل A في المركم
MULT B	اضرب B في A
ADD C	أضف C إلى A*B
STORE X	خزن الناتج في العنوان X في الذاكرة

3. الصورة الواقعية: لغة الآلة

كتبت الأوامر أعلاه باستخدام الرموز و الحروف العادية تسهيلاً على الدارس. هذه الأوامر لا يستطيع الحاسوب فهمها و تنفيذها إلا إذا حولت إلى لغة الآلة machine language وهي مجموعة من الرموز الثنائية التي يفهم الحاسوب معناها ومغزاها وتأتي معه من المصنع. مثلاً العمليات الأربعة التي استخدمناها في الفقرة أعلاه قد تُمثل في حاسوب افتراضي بالرموز الثنائية التالية:

LOAD	00010000
STORE	00010010
ADD	00100111
MULT	00101001

بناءً على ذلك تتحول سلسلة الأوامر الأربعة التي كتبناها لحساب قيمة X في الفقرة أعلاه إلى السلسلة الثنائية التالية:

00100000	00000100	→ LOAD A
00101001	00001000	→ MULT B
00100111	00001100	→ ADD C
00010010	00010000	→ STORE X

في هذا المثال الافتراضي، افترضنا أن A مخزنة في الخانة 4 (00000100) في الذاكرة، وأن B مخزنة في الخانة 8 (00001000) و C مخزنة في الخانة 12 (00001100) و X يتم تخزينها في الخانة 16 (00010000) في نهاية هذا البريمج.

صدق أو لا تصدق أن المبرمجين كانوا يستخدمون هذه اللغة الثنائية مباشرةً في التعامل مع الحاسوب عند ظهوره في الأربعينات ولكن الصورة الآن تغيرت كثيراً كما سترى في الفقرات أدناه.

4. برامج التجميع والمترجمانات

Assemblers and Compilers

الذي يتضح من الفقرات أعلاه هو أن:

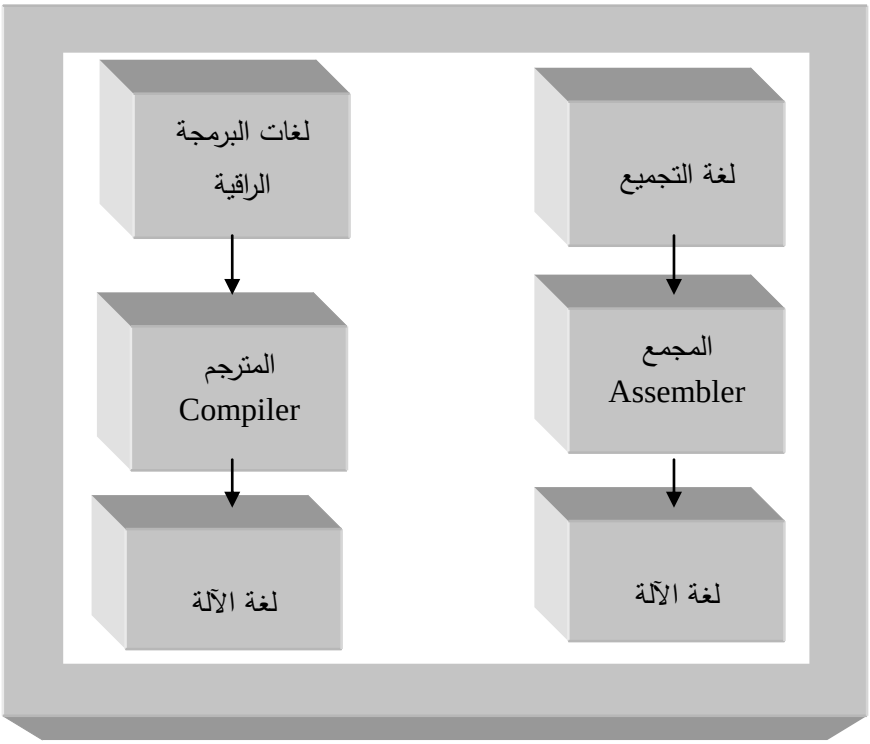
- الحاسوب لا يستطيع أن ينفذ العمليات المركبة مباشرةً إلا بعد أن تقسم إلى جرعات صغيرة لا تنطوي على أكثر من معامل أو معاملين.
- الحاسوب لا يفهم إلا لغة الآلة وهي مجموعة من الرموز الثنائية المركبة في الجهاز من المصنع.
- العمليات التي يمكن أن ينفذها الحاسوب مباشرةً محدودة كماً وكيفاً ولا تخرج عن العمليات الحسابية و المنطقية البسيطة.

السؤال الذي يطرح نفسه: كيف استطاع المبرمجون عبر السنوات الماضية التعامل مع الحاسوب؟ وكيف يستطيع الحاسوب التعامل مع هذا الكم الهائل من البرامج و التطبيقات المعقدة التي نراها اليوم، بل كيف تم إنتاج هذه البرامج و التطبيقات بادئ ذي بدء؟

الإجابة تكمن في حيلة بشرية أخرى: "المجمعات و المترجمات". إذا كان الحاسوب لا يفهم إلا لغة الآلة و لا يستطيع التعامل إلا مع عمليات بسيطة، دعنا نكتب البرامج ونصيغ الحلول بالصورة التي تريحنا كبشر ثم "نترجم" هذه البرامج و الحلول إلى لغة الآلة باستخدام برامج حاسوبية خاصة. المجمعات و المترجمات عبارة عن برامج حاسوبية تقوم بعملية الترجمة من لغات البرمجة المعروفة إلى لغة الآلة.

تستخدم المجمعات في ترجمة برامج لغة التجميع إلى لغة الآلة.

أما المترجمات فتستخدم في ترجمة البرامج المكتوبة بلغات البرمجة عالية المستوى (C, Pascal) إلى لغة الآلة.



وعليه لكي تبرمج الحاسوب بإحدى لغات البرمجة الراقية مثل C++، لابد من توفر مترجم اللغة Language Compiler في البداية. هناك العديد من المترجمات التي ننتجها جهات تجارية و غير تجارية لكل لغة من لغات البرمجة وما عليك إلا أن تختار المترجم الذي يعجبك و يناسب احتياجاتك من حيث الفاعلية و سهولة الاستخدام ورخص الثمن.

أسئلة تقويم ذاتي

برهن الحقائق الآتية:

1. عقل البشر أفضل من وحدة المعالجة المركزية للحاسوب.
2. ذاكرة الحاسوب أفضل من ذاكرة البشر في معظم الأحيان.

5. المراحل العملية لتطوير البرامج

☜ سنعرض في الوحدات القادمة الكثير من الأمثلة و التمارين البرمجية التي تتطلب التنفيذ المباشر و المتابعة. **البرامج المعطاة في هذا الكتاب تم تصميمها حسب المواصفات القياسية للغة C++ ويمكن تنفيذها باستخدام أي نظام قياسي للغة C++.** هناك الكثير من أنظمة C++ في الساحة الآن، منها الحر ومنها التجاري، منها مايعطي المترجم فقط، ومنها مايعطي بيئة متكاملة للتطوير. في هذه الفقرة نشرح بصورة عملية استخدام نظام Visual C++ من شركة مايكروسوفت، رجاء أن يستفيد الدارس منه في تنفيذ البرامج المرفقة والتمارين. وبما أن هذا النظام في غاية التعقيد، ويعطي بيئة مترامية الأطراف لتطوير النظم الكبيرة، فإننا سنتعرض فقط لسمة واحدة من سماته وهي: تطوير البرامج النصية البسيطة (Console Applications).

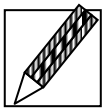
البرنامج البسيط التالي سيستخدم كوسيلة لتوضيح خطوات التطوير:

```
#include <iostream.h>
int main()
{ cout << "Assalamu Alaikum Wa Rahmatu Allah \n";
return 0;}
```

☜ **تنبيه:**

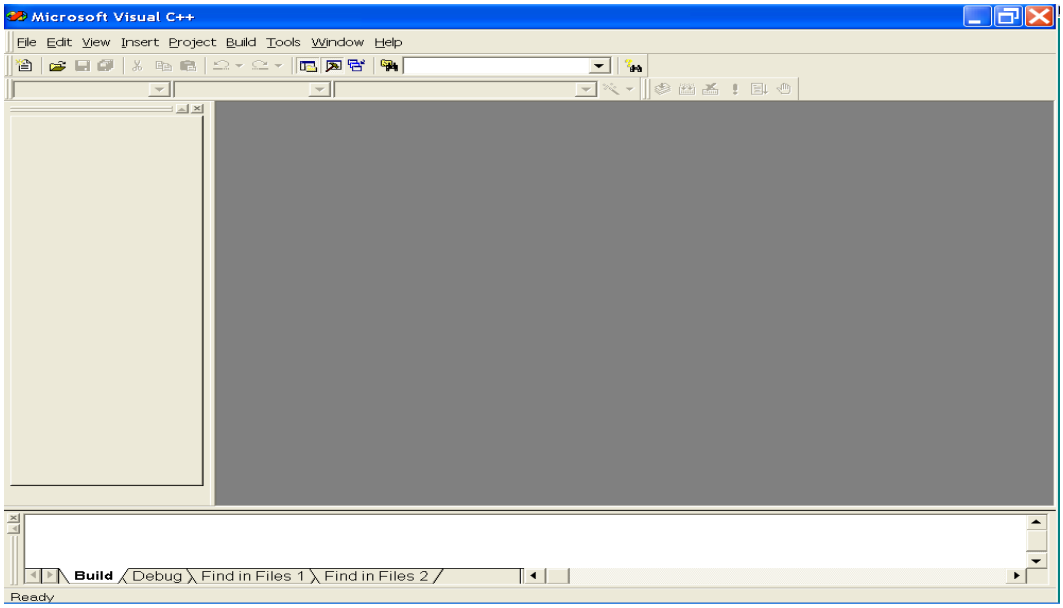
هناك نسخة تجريبية للطلاب من نظام Visual Studio 6 تعطى مجاناً. يمكن الحصول على هذه النسخة أو الحصول على النظام الكامل وتنزيله على الجهاز قبل البدء في التدريبات التالية.

تدريب (1)

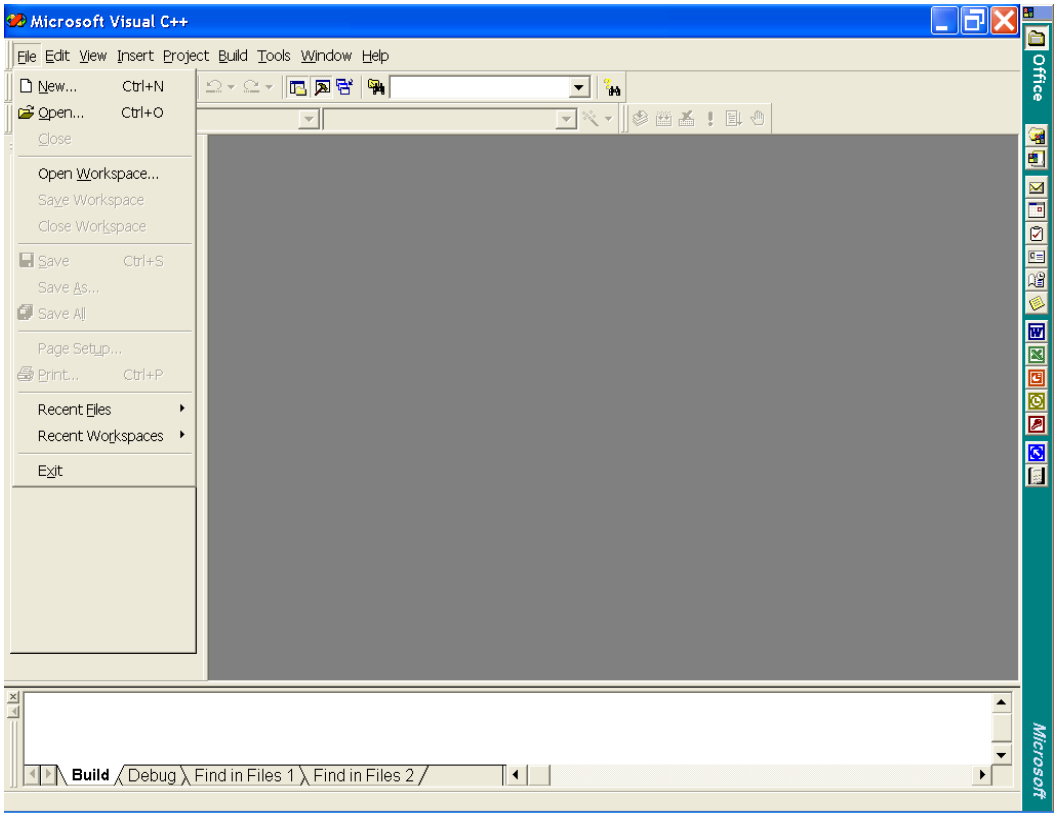


تطوير البرامج البسيطة باستخدام نظام Visual C++:

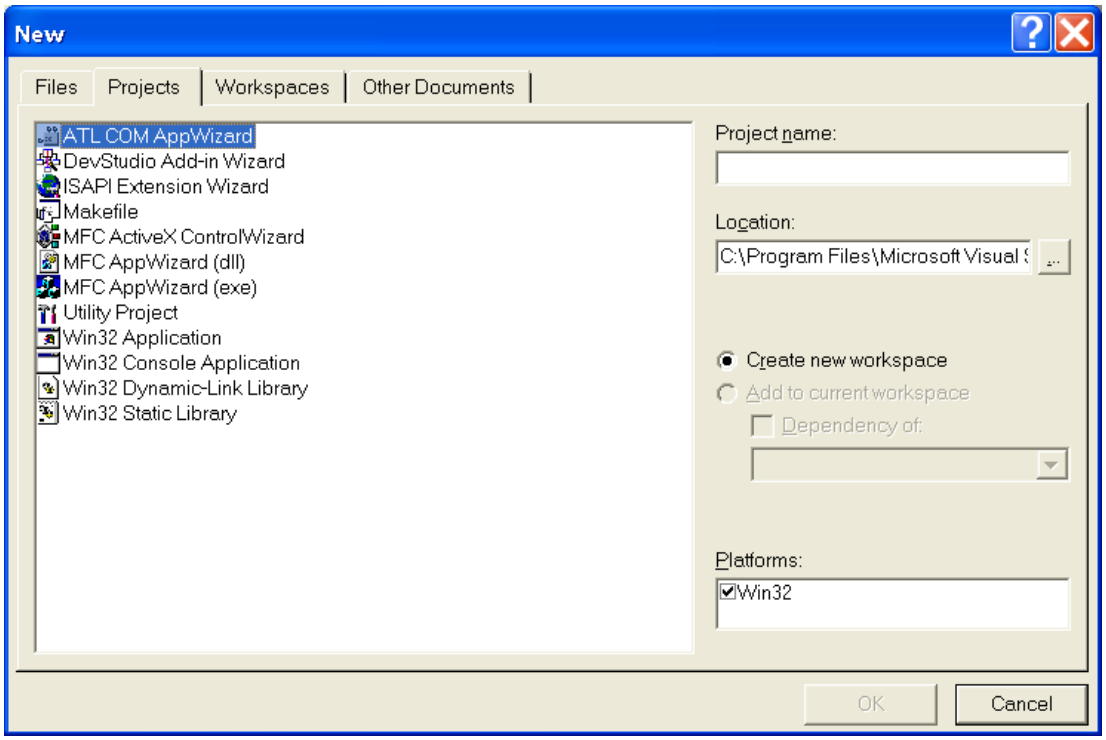
افتح نظام Visual Studio كما هو موضح في الشكل أدناه



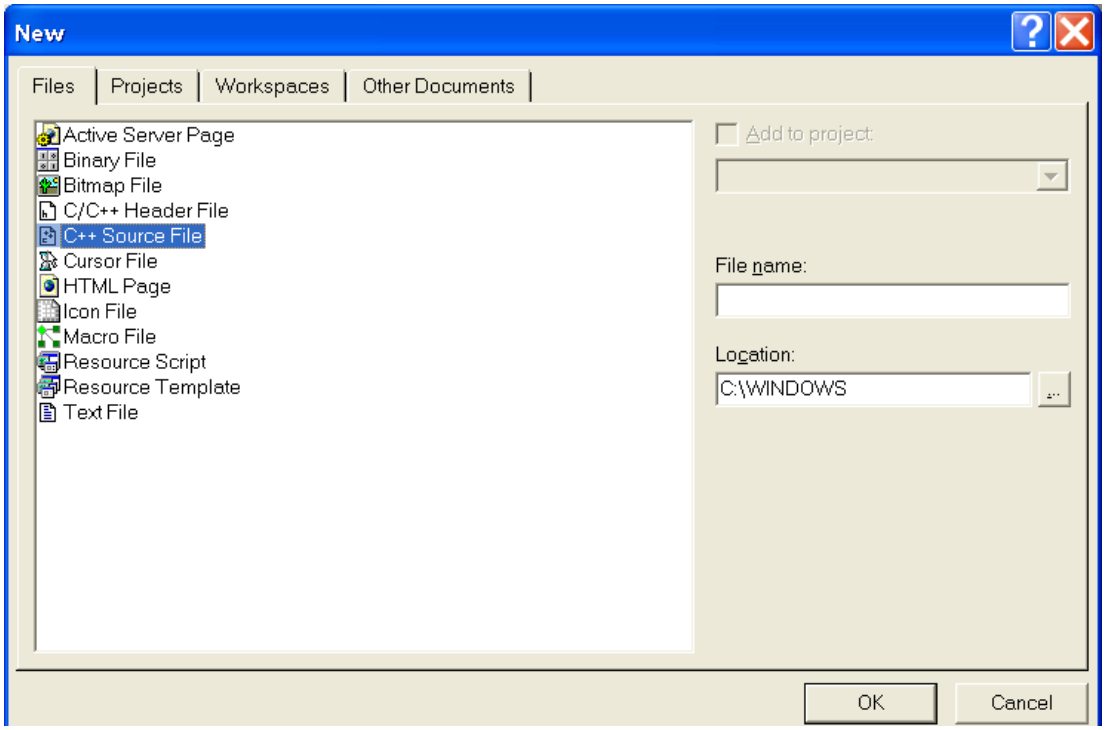
انقر على خيار الملفات File و اختر New من الخيارات التي تظهر:



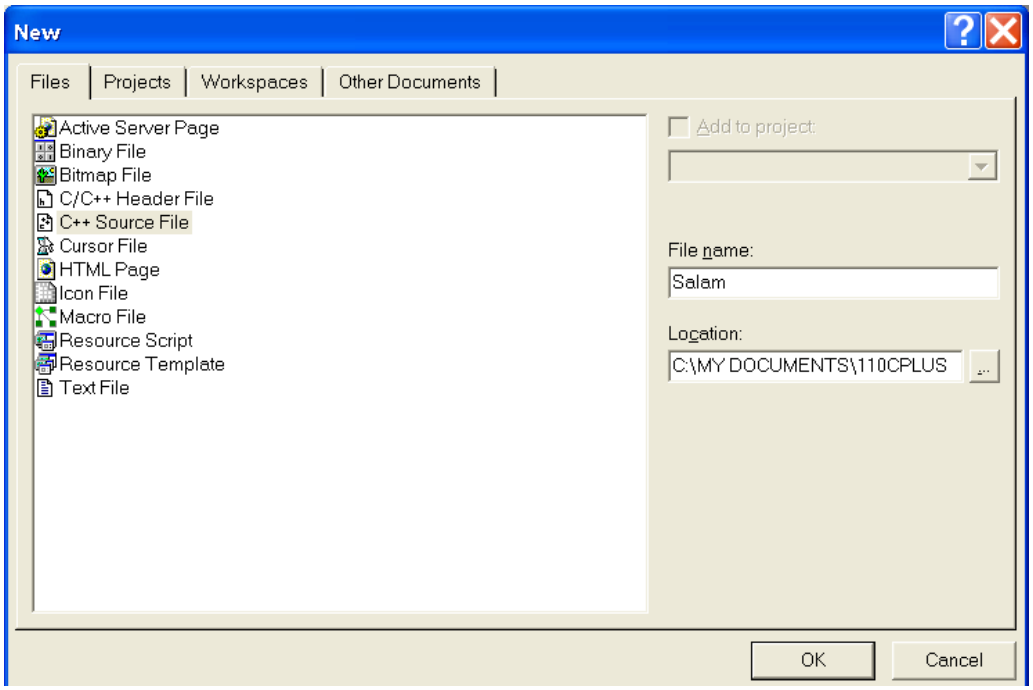
بعد النقر على خيار New تظهر النافذة التالية:



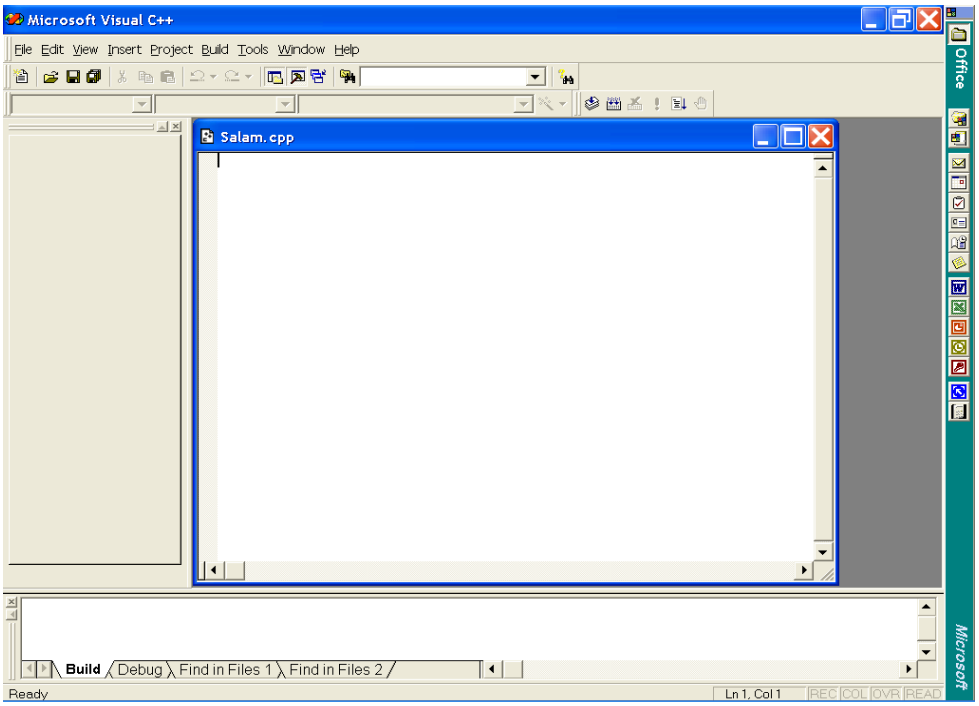
تحول إلى لوحة الملفات Files في النافذة أعلاه ,واجعلها اللوحة النشطة، ثم اختر C++ Source File كما هو موضح في الشكل أدناه.



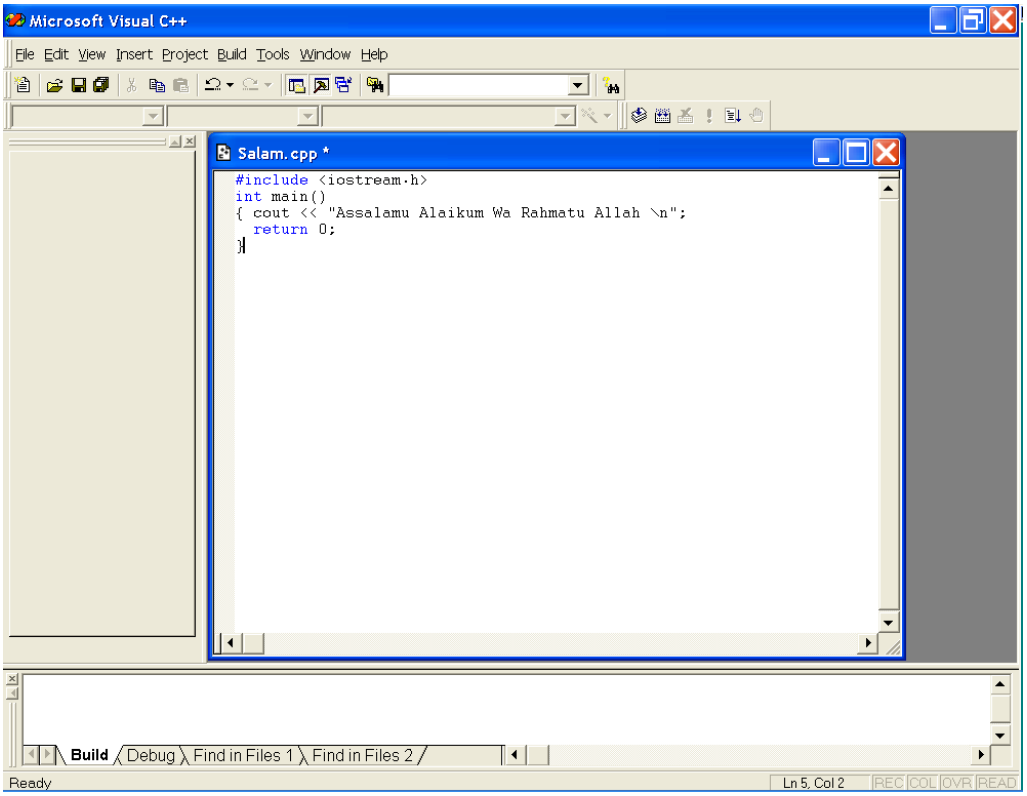
1- اختر اسماً مناسباً للمشروع البرمجي المزمع و اكتبه في صندوق File Name على يمين نافذة الحوار كما هو مبين في الشكل أدناه:

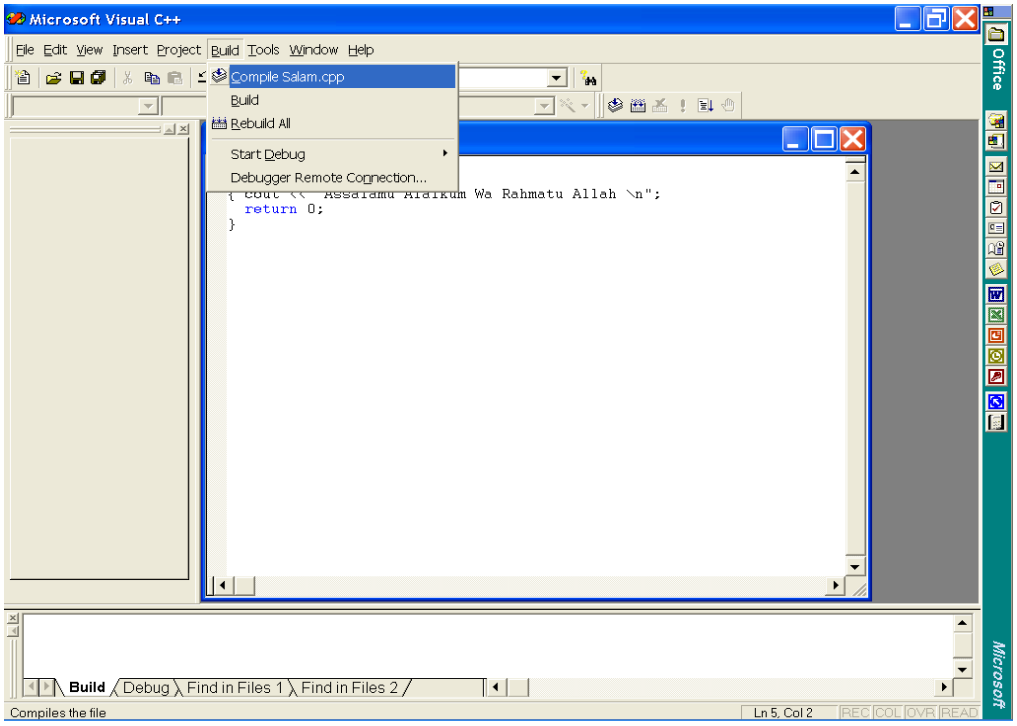


انقر على مفتاح ok لتحصل على النافذة التالية:

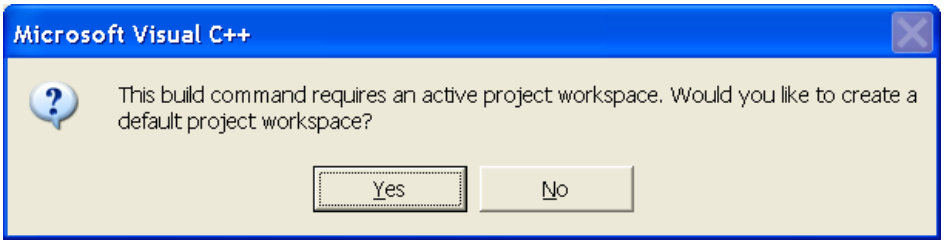


2- اكتب البرنامج البسيط في النافذة الجديدة كما هو موضح أدناه:

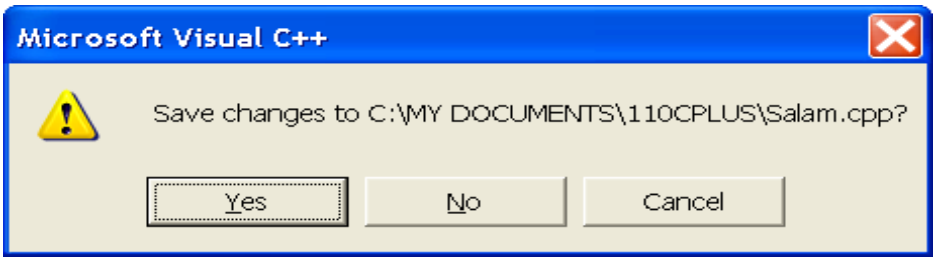




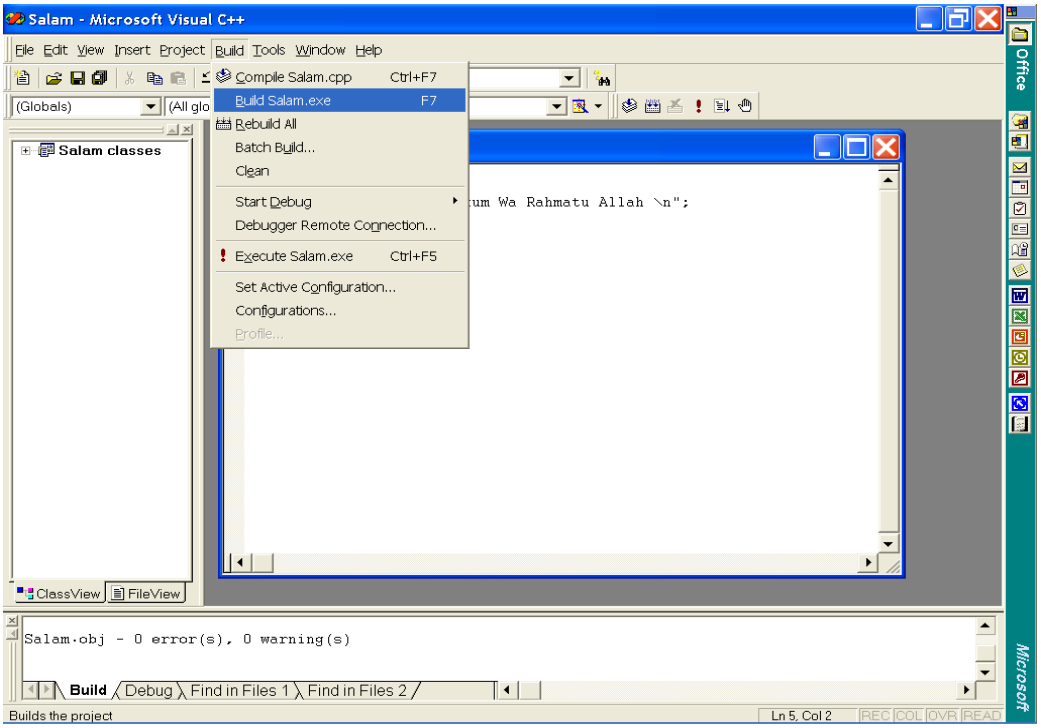
3- إذهب إلى خيار Build من قائمة الخيارات الأساسية و اختر Compile Salam.cpp كما هو موضح في الشكل أعلاه. بعد النقر على خيار Compile Salam.cpp تأتيك الرسالة التالية:

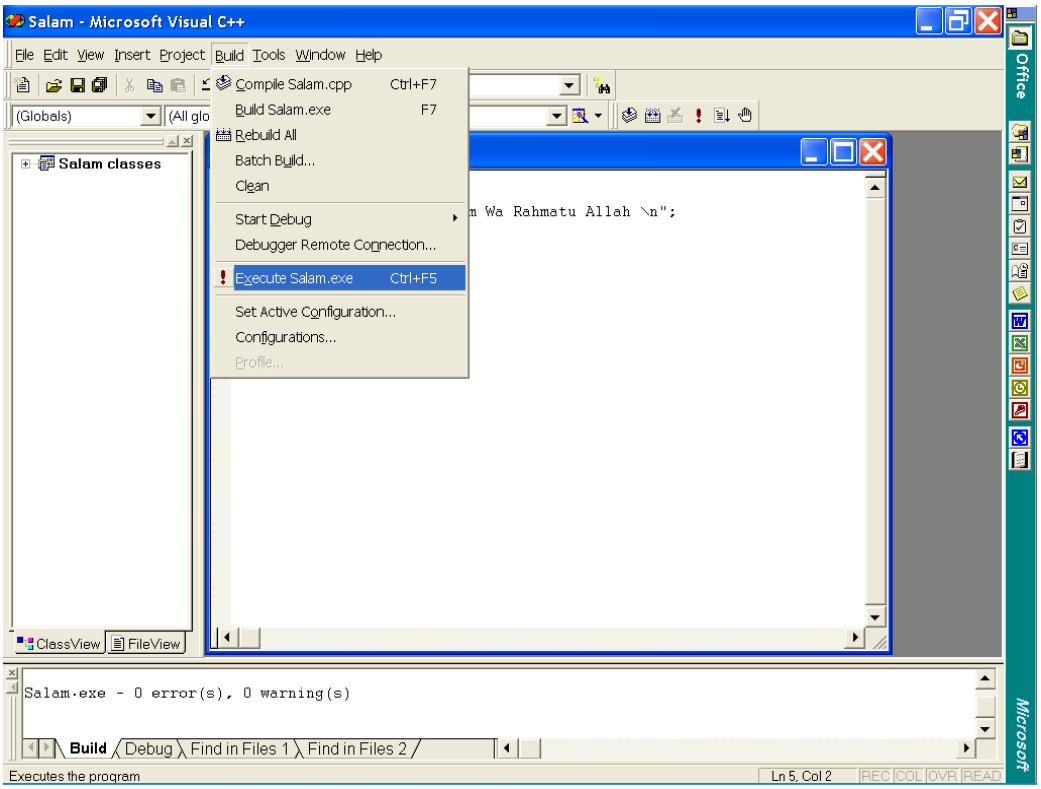


انقر على الزر "Yes" تأتيك الرسالة التالية:

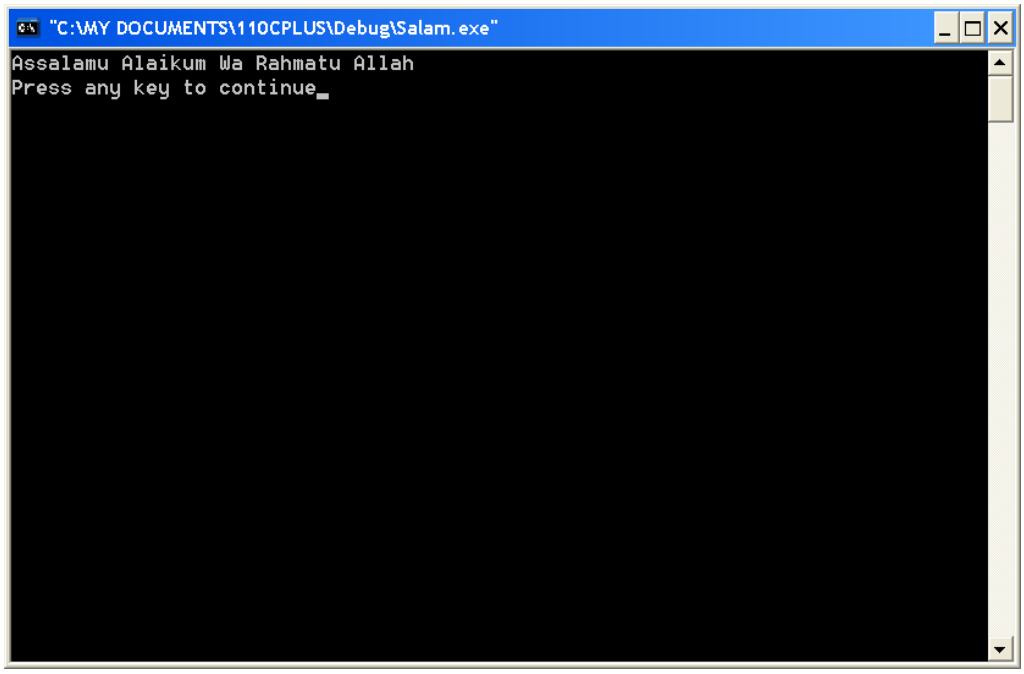


اضغط على الزر "Yes" مرة أخرى لتكتمل عملية الترجمة وتحصل على رسالة أسفل الشاشة أنه ليست ثمة أخطاء أو تحذيرات (أنظر الشكل أدناه). بعدها انقر خيار Build Salam.exe من خيار Build في القائمة الرئيسية:


Salam - Microsoft Visual C++
File Edit View Insert Project Build Tools Window Help
Compile Salam.cpp Ctrl+F7
Build Salam.exe F7
Rebuild All
Batch Build...
Clean
Start Debug
Debugger Remote Connection...
Execute Salam.exe Ctrl+F5
Set Active Configuration...
Configurations...
Profile...
Salam classes
ClassView FileView
Salam.obj - 0 error(s), 0 warning(s)
Build Debug Find in Files 1 Find in Files 2
Builds the project
Ln 5, Col 2


Salam - Microsoft Visual C++
File Edit View Insert Project Build Tools Window Help
Compile Salam.cpp Ctrl+F7
Build Salam.exe F7
Rebuild All
Batch Build...
Clean
Start Debug
Debugger Remote Connection...
Execute Salam.exe Ctrl+F5
Set Active Configuration...
Configurations...
Profile...
Salam classes
ClassView FileView
Salam.exe - 0 error(s), 0 warning(s)
Build Debug Find in Files 1 Find in Files 2
Executes the program
Ln 5, Col 2

عملية بناء ملف التنفيذ تتم بعد مرحلة الترجمة. بعد عملية البناء يمكننا تنفيذ البرنامج بالضغط على خيار Execute Salam.exe من خيار Build في القائمة الأساسية. عند تنفيذ البرنامج نحصل على النافذة أدناه:



6. حل المسائل باستخدام الحاسوب

في بداية استخدام الحاسوب لم تكن هناك أساليب محددة أو طرق معينة لحل المسائل على الحاسوب، بل كان المختصون يهرعون إلى كتابة البرامج مباشرة بعد فهم المسألة. ولكن سرعان ما تبين للكثيرين أنه لابد من تبني أساليب منظمة و منسقة لتصميم الحلول وصياغتها قبل تحويلها إلى برامج، ذلك أن المبرمج سرعان ما ينسى تفاصيل ما كتب ويصعب عليه متابعة ومراجعة البرامج التي كتبها هو نفسه، دع عنك البرامج التي كتبها الآخرون. أضف إلى ذلك أن صعوبة المسائل الآن ازدادت وصار من الصعب أن يقوم شخصٌ واحدُ بتصميم نظام حاسوبي متكامل، بل لابد من تعاون أكثر من شخص لتصميم النظام. كل هذه الدوافع وغيرها أدت إلى أن يبحث المختصون عن طرق وأساليب تساعد في تنظيم عملية تصميم البرمجيات وجعلها أكثر جدوى وفاعلية. المجهودات في هذا الاتجاه أدت إلى ظهور علم كامل من علوم الحاسوب يسمى "علم هندسة البرمجيات". في هذا الفصل نعرض لبعض الأساليب البسيطة التي تساعد في صياغة الحل بصورة مبسطة بغض النظر عن لغة البرمجة المستخدمة.

7. دعامات أساسية

صياغة وتصميم حلول المسائل على الحاسب تتبني على دعامتين أساسيتين:

- فهم طبيعة الحاسوب وإمكاناته: وهذا يعني أن مصمم الحل لابد أن يكون من ذوي الاختصاص في مجال الحاسب أو ذوي الإطلاع المناسب على تركيب الحاسب وكيفية التعامل معه.
- فهم المسألة المراد حلها: وهذا يتأتى بتحليل المسألة تحليلاً مفصلاً ومعرفة متطلباتها الحاسوبية تمهيداً لصياغة الحل. بالطبع تحليل المسألة لابد أن يؤسس على المعرفة بطبيعة الحاسوب.
- اتباع نهج سليم لصياغة الحل قبل تحويله إلى لغة البرمجة. هذا النهج ينبغي أن يكون مستقلاً ما أمكن عن لغة البرمجة المستخدمة.

☜ وقد قام المختصون في مجال هندسة الحاسبات بنشر الكثير عن أساليب التحليل والتصميم و أصبحت هناك مدارس مختلفة ومناهج يمكن للدارس الإطلاع عليها من مصادرها.

8. مراحل حل المسألة

بغض النظر عن النهج و الأسلوب المتبع في حل المسألة، فإنه من المتفق عليه في أوساط هندسة البرمجيات أن حل المسائل حاسوبياً يمر بالمراحل التالية:

1.8 مرحلة التحليل وتحديد المطلوب

في هذه المرحلة يتم تحليل المسألة بدقة ليتم تحديد المطلوب و المتطلبات الحاسوبية للمسألة. توثق نتيجة هذه المرحلة باستخدام المصطلحات والرسومات المتعارف عليها في الطريقة المتبعة.

2.8 مرحلة التصميم

في هذه المرحلة تتم صياغة حل المسألة باستخدام مصطلحات و اصطلاحات طريقة التصميم بعيداً عن لغة البرمجة. قد ينتهي المصمم في هذه المرحلة بأكثر من رسم وأكثر من نص. المهم هنا صياغة الخوارزميات المعنية بصورة واضحة ودقيقة. كما ذكرنا في الفقرات أعلاه هناك أكثر من مدرسة و أكثر من طريقة في هذا المجال. فمثلاً هناك المدرسة التركيبية والمدرسة الموضوعية وينضوي تحت كل مدرسة أكثر من طريقة و أكثر من نهج، ولكلٍ مصطلحاتها ورموزها.

3.8 مرحلة توليد البرامج

في هذه المرحلة يتم توليد البرامج بصورة شبه تلقائية من نماذج التصميم الناتجة في المرحلة السابقة. معظم الطرق المستخدمة تحاول قدر الإمكان أن تجعل هذه المرحلة عملاً روتينياً لا يستغرق من المبرمج أي نوعٍ من الجهد. الشيء الآخر الذي تركز عليه طرق التصميم

هو إعطاء المبرمج الحرية في اختيار لغة المبرمجة التي تروقه. فمثلاً يمكن للمبرمج أن يولد البرامج المطلوبة في لغة باسكال أو لغة السي بعد انتهاء مرحلة التصميم دون أن يضطر المصمم إلى تغيير النماذج الناتجة.

4.8 مرحلة الاختبار و التدقيق

في هذه المرحلة يتم اختبار البرامج المولدة باستخدام عينات مختارة من البيانات و النتائج. يتم اختيار هذه العينات بطريقة علمية حسب نوع المسألة ومجال التطبيق. هناك أسلوبان مشهوران لاختبار البرامج: أسلوب الصندوق الزجاجي الذي يركز على اختبار البرامج من الداخل، وأسلوب الصندوق الأسود الذي يركز على اختبار البرامج باعتبار المدخل والناتج فقط دون الدخول في تفاصيل الخوارزميات. عملية الاختبار تتم في عدة مراحل بناءً على نوع التطبيق والمستفيد النهائي.

5.8 مرحلة المتابعة

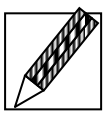
تبدأ هذه المرحلة بعد وصول البرنامج إلى المستفيد ولا تنتهي إلا بعد أن يستغنى من البرنامج كلياً. تشمل عمليات هذه المرحلة إصلاح الأخطاء التي تكتشف بعد بدء الاستخدام الواقعي وعمليات التحسين وعمليات التطوير التكيفي والوقائي.

9. أسلوب مبسط لحل المسائل

☜ المراحل أعلاه ينبغي أن يلتزم بها في تصميم النظم الحاسوبية بجميع أنواعها، صغيرةً كانت أم كبيرة، طالما أن هناك مستفيد نتعامل معه. بالنسبة للمسائل البسيطة الواردة في هذه المذكرة سنلتزم قدر الإمكان بالمراحل الثلاثة الأولى. المرحلة الرابعة (اختبار البرامج) نغطى منها جزءاً محدوداً ونترك البقية لك **عزيزى الدارس**. مرحلة المتابعة لا تنطبق على مسائل هذا المنهج المبسطة، ولكن يمكن ممارستها في مشاريع عملية تتولد من هذا المقرر. نموذج حل المسائل الذي نتبناه يتكون من المراحل التالية:

1) **مرحلة التحليل:** في هذه المرحلة نحدد المطلوب، المعطيات و المتغيرات الوسيطة التي ستستخدم في الحل.
2) **مرحلة التصميم:** في هذه المرحلة نعطى وصفاً للخوارزميات المطلوبة باللغة العربية العادية مع استخدام أسماء المتغيرات وبعض المصطلحات.
3) **مرحلة توليد البرنامج:** في هذه المرحلة يتم توليد البرامج المطلوبة من الخوارزميات التي صممت في المرحلة السابقة.

تدريب (2)



1. كيف تختلف الذاكرة الثانوية عن الذاكرة الأساسية ؟
2. أذكر الخطوات المتبقية داخل الحاسوب لاجراء العملية الحسابية التالية:

$$Y = (N - L) \ / \ M$$

حيث: M = 4 , N = 25 , L = 1

الخلاصة

أخي الدارس، لقد احتوت الوحدة الأولى بعنوان (مدخل عام) على وصف لطبيعة مكونات الحاسوب ووظائف أجزائه وتعتبر هذه الوحدة تهيئة للدارس حتى يستوعب مادة المقرر (أساليب البرمجة).

ولذا عزيزى الدارس يجب دراسة هذه الوحدة بجدية فقد اشتملت على كيفية تطوير برامج C++ باستخدام أمثلة عملية ومنهج علمي مبسط لتحليل المسائل وتصميم الحلول الحاسوبية لها والدعامات الأساسية لتصميم حلول للمسائل بواسطة الحاسوب.

لمحة مسبقة عن الوحدة التالية

عزيزي الدارس، في الوحدة الثالثة التى بعنوان (أساليب البرمجة و الهيكل العام للبرامج) سنتعرض لقواعد لغة C++ الاساسية في البرمجة و الهيكل العام للبرامج وسنقوم بتطبيق عدد مقدر من الأمثلة العملية على الحاسوب حتى نتمكن من استيعاب أساليب البرمجة.

إجابات التدريبات

تدريب (2)

1- تختلف الذاكرة الثانوية عن الذاكرة الاساسية بأنها تمتاز باستمرارية المعلومات حيث أنها تحتفظ بالمعلومات حتى بعد انفصال التيار الكهربائي كما يسهل حملها من مكان لآخر (الأقراص الصلبة، والأقراص المرنة).

2- الحل:

الخطوات هي:

Load N جملة M في المركز

SUB L اطرح L من N

DIV M اقسم الناتج على M

STORE Y خزن الناتج Y في الذاكرة.

مسرد المصطلحات

وحدة المعالجة المركزية Control Process Unit

تشمل وحدة التحكم والوحدة الحسابية المنطقية والذاكرة الأساسية ووظيفتها احضار البرامج والبيانات واجراء العمليات الحسابية والتنسيق بين أجزاء الحاسوب المختلفة.

الحافظات Registers

هي خانات خاصة توجد في الذاكرة السريعة تستخدم لحفظ البيانات ولها أسماء خاصة.

المركم Accumulator

هي من أشهر أنواع الحافظات توجد في وحدة المعالجة المركزية وتستخدم في إجراء العمليات الحسابية.

لغة الآلة Machine Language

هي مجموعة من الرموز الثنائية التي يفهم الحاسوب معناها ومغزاها وتأتي معه من المصنع.

برامج التجميع Assemblers

تكون لغة التجميع من اختصارات سهلة الذكر أو الرموز المختصرة مثل MULT و ADD ويجب ترجمة البرنامج المكتوب بلغة التجميع قبل تنفيذه ويستخدم برنامج خاص لذلك يسمى المجمع (Assembler).

برامج الترجمة Compilers

هو عبارة عن برنامج يحول البرنامج المصدري المكتوب بلغة عالية المستوى الى برنامج هدفى مكتوب بلغة الآلة.

مترجم اللغة Language Compiler

عبارة عن برنامج يقوم بترجمة وتنفيذ جملة واحدة في الوقت الواحد بمجرد ادخالها الى الحاسوب.

البرامج النصية البسيطة Console Application

هذه برمجيات تطوع الحاسوب لتنفيذ وظائف مفيدة تامة مثل معالجة النصوص وقواعد البيانات والجداول الالكترونية.

المصادر والمراجع

1- Deictl, H. & Deitel, P., **C++ How to Program.** Third Edition, Prentice Hall: 2001.
2- Savitch, W. **Problem Solving With C++: The Object of Programming**. Third Edition, Addison Wesley: 1999.
3- Friedman, F. and Koffman, E. **Problem Solving, Abstraction, and Design Using C++** (2 nd Edition). Addison Wesley: 1996.
4- Lippman, S. lajoie, J. **C++ Primer,** (3rd Edition). Addison Wesley: 1998.
5- Horstmann, C. **Computing Concepts With C++ Essentials.** John Wiley: 1997.
6- Stroustrup, B. **The C++ Programming Language.** (3 rd edition), Addison Wesley: 1990.
7- Ellis, M. and Bjarnc Stroupstrup, **The Annotated Reference Manual.** Addison Wesley: 1990.
8- Tcale, S. C++ IOS treams Handbook. Addison Wesley: 1993.
9- Meyers, S. **Effective C++: 50 Specific Ways to Improve Your Programs and Designs** (2 nd edition). Addison Wesley: 1996.
10- Meyers, S. **More Effective C++: 35 New Ways to Improve Your Programs and Designs.** Addison Wesley. 1996.
11- Schildt, H**. C++ From the Ground up.** Mc Graw Hill. 2003.
12- Savitch, W. **Absolute C++.** Addison Wesley: 2002.
13- Cogswell, J. **C++ all – in – one Desk Refcrence for Dummies.** 2002.


الوحدة الثانية
مدخل إلى لغة C ++
أساسيات البرمجة والهيكل العام للبرامج

محتويات الوحدة

الموضوع	الصفحة
المقدمة	**32**
تمهيد	32
أهداف الوحدة	32
1. أساسيات البرمجة	**33**
1.1 البرنامج الأول: برنامج التحية	37
2.1 البرنامج الثاني وملاحظات عليه	37
2. التعريف بقواعد لغة ++C	**39**
3. الهيكل العام لبرامج لغة ++C	**40**
4. المتغيرات البسيطة في لغة ++C وكيفية تعريفها	**41**
5. تسمية المتغيرات و الثوابت في لغة ++C	**43**
6. كيف يحصل البرنامج على البيانات؟	**47**
7. كيف يطبع البرنامج البيانات: التحكم في شكل الناتج؟	**47**
8. العمليات الحسابية و الدوال البسيطة في لغة ++C	**49**
9. أسبقية العمليات الحسابية في لغة ++C	**50**
10. الدوال الجاهزة في لغة	**52**
11. المتغيرات وذاكرة الحاسوب	**52**
12. إسناد القيم الى المتغيرات: جملة الإسناد	**53**
13. جملة الإحلال الذاتي	**54**
14. عوامل الاضافة و التنقيص المختصرة في لغة ++C	**54**
15. عوامل الاحلال الذاتي المختصرة	**56**
الخلاصة	**58**
لمحة مسبقة عن الوحدة التالية	**58**
مسرد المصطلحات	**59**
المصادر والمراجع	**60**

المقدمة

تمهيد

عزيزي الدارس،

مرحبا الى هذه الوحدة الثانية من مقرر اساليب البرمجة. تحتوي هذه الوحدة على مبادئ اساسيات البرمجة وتعريف قواعد لغة C++ والهيكل العام لبرامج لغة C++ .وشرح مبسط للمتغيرات البسيطة وكيفية تعريفها داخل لغة C++ وتسمية المتغيرات والثوابت وكيفية استعمالها فى ادخال البيانات واجراء العمليات الحسابية عليها ثم طباعتها مستخدمين فى ذلك أوامر لغة C++.

وفى الاقسام الأخيرة من الوحدة تعرضنا لدراسة المتغيرات وذاكرة الحاسوب واسناد القيم الى المتغيرات والتعريف بالفرق بين جملة الاسناد والاحلال الذاتى ولذلك قمنا بعرض عوامل الاضافة والتنقيص وعرض جدول لتوضيح عوامل الاحلال الذاتى المختصرة . هذه الوحدة تحوي الكثير من البرامج التى يجب على الدارس ان يقوم بادخالها فى الحاسوب واختبارها لذا نرجو منك عزيزى الدارس أن تعطيها وقتاً أطول لاتقان المهارات التى يحتويها المقرر.

مرحباً بك في هذه الوحدة التى سنتعرض فيها لأساسيات البرمجة و الهيكل العام للبرامج عبر لغة C++.

أهداف الوحدة



عزيزي الدارس،

بعد دراستك لهذه الوحدة ينبغي أن تكون قادراً على أن:

- **توظف** الهيكل العام للبرامج في لغة C++ في تصميم أي برنامج.
- **تستخدم** أوامر إدخال البيانات وطباعة النتائج و كيفية التحكم في شكل النتائج.
- **تستخدم** التعابير الحسابية البسيطة وجمل الاسناد.

1. أساسيات البرمجة

كتابة البرنامج هي المرحلة الثالثة في نموذج حل المسائل الذي تبنيناه في الوحدة الثانية أعلاه. في هذه المرحلة يتم تحويل الخوارزمية التي تم تصميمها في المرحلة السابقة إلى برنامج في إحدى لغات البرمجة. إحدى الفروق الأساسية بين الخوارزمية و البرنامج هو أن البرنامج لا بد فيه من الانضباط الدقيق بقواعد لغة البرمجة المعنية. أقل خطأ في البرنامج يؤدي إلى رفض البرنامج من قبل المترجم و عدم تنفيذه. و عليه فإن تعلّم قواعد لغات البرمجة يعتبر متطلباً أساسياً لحل المسائل بالحاسوب.

في هذه الوحدة نبدأ التعرض لقواعد البرمجة بلغة C++ ونركز في البداية على الهيكل العام لبرامج C++ ومبادئ البرمجة. البرامج التي نعرض لها في هذا الفصل في غاية البساطة ولا تؤدي عملاً معتبراً، إنما تهدف فقط إلى توضيح بعض المفاهيم الأساسية. لن نستطيع أن نكتب برامج جادة إلا بعد الوحدة الثالثة.

1.1 البرنامج الأول: برنامج التحية

جرت العادة أن تبدأ كتب البرمجة ببرنامج بسيط يلقي التحية على المستخدم ويكسر الحاجز الأساسي في التعامل مع مترجم اللغة. البرنامج البسيط أدناه يقوم بطباعة عبارة السلام على الشاشة. نعطي البرنامج أولاً ثم نقوم بتحليله واستخلاص دلالاته:

مثال (1)

```
#include <iostream.h>
) int main(
{ cout <<"Assalamu Alaikum Wa Rahmatu Allah\n";
return 0;
}
```

تحليل البرنامج:

أولاً: يبدأ البرنامج بمنطقة افتتاحية تحتوي حالياً على سطر واحد هو:

#include <iostream.h>

كما هو ملاحظ أن هذا السطر يبدأ بعلامة (#). السطور التي تبدأ بهذه العلامة تعتبر موجهات للمترجم، تطلب منه القيام بعمل ما قبل البدء في البرنامج (Preprocessor Directives). الموجهات لابد أن تبدأ في العمود الأول من يسار الصفحة. موجه "الإحتواء": #include يطلب من المترجم إضافة ملف آخر إلى البرنامج قبل البدء في الترجمة. الملف المطلوب إضافته في هذه الحالة هو iostream.h، وهو ملف يحتوي على التعريفات النموذجية لبعض دوال إدخال وطباعة البيانات. لاحظ أن إسم الملف مكتوب بين معقوفتين < >، وهذه قاعدة عامة في موجه الإحتواء #include.

المنطقة الإفتتاحية في برامج C++ قد تحتوي على موجهات أخرى أو تعريفات نعرض لها في حينها. إحدى خواص المنطقة الإفتتاحية أنها عامة ويمكن الوصول إلى عناصرها من كل مناطق البرنامج، وهذه نقطة ستظهر أهميتها فيما بعد.

ثانياً: بعد المنطقة الإفتتاحية يأتي عنوان البرنامج و الذي يتكون من سطر واحد هو:

(int main)

الكلمة الأولى في العنوان: int إعلامٌ بأن هذا البرنامج ينبغي أن "يرجع" نتيجة واحدة عند انتهائه، وهذه النتيجة ينبغي أن تكون عدداً صحيحاً (integer). في معظم الأحيان تتعامل C++ مع البرنامج كأنه "دالة" (function) تقوم بعمل معين، بناءً على بعض المعطيات وترجع ناتجاً من نوعٍ ما. الناتج المقصود في هذه الحالة يكون غالباً إشارة إلى نجاح البرنامج أو فشله في أداء مهمته. في بعض الأحيان يمكن التنازل عن هذا الشرط، وذلك باستخدام كلمة void،

(void main())

كلمة main هي إحدى الكلمات الخاصة في لغة C++ و تستخدم إسماً للبرنامج الرئيس (الدالة الرئيسة)، ذلك أن البرامج في لغة C++ قد تتكون من عدة دوال متجاورة (C++ لاتسمح بتداخل الدوال). كلمة main تميز البرنامج الأساس عن غيره من الدوال المجاورة. هناك كلمات خاصة كثيرة في C++ لايسمح باستخدامها إلا للغرض الذي

خصصت لـه. كلمة main مشفوعة بقوسين () تَدُّل على أن البرنامج main ماهو إلا دالة مثل الدوال الأخرى في سي++. أسماء الدوال في C++ تكون دائماً متبوعة بقوسين تتخللها بعض العناصر أحياناً.

ثالثاً: يأتي صلب البرنامج (program body) بعد العنوان والذي يكون دائماً محصوراً بين الحاصرتين: {}. البرنامج في هذه الحالة يتكون من جملتين فقط:

```
cout <<"Assalamu Alaikum Wa Rahmatu Allah\n";
return 0;
```

كل جملة في البرنامج تنتهي بالفاصلة المنقوطة ; وهذه قاعدة عامة في C++.

رابعاً: تستخدم الجملة الأولى الأمر الجاهز cout لطباعة رسالة السلام على الشاشة. الأمر cout يتطلب استخدام علامة "التمرير إلى الخارج" "<<" (المركبة من المعقوفتين) قبل العنصر المراد طباعته. لايهم إن كانت علامة التمرير ملصقة مع ماحولها أم لا، لكن لابد أن تكون المعقوفتان ملتصقتين مع بعضهما البعض. علامة التمرير ينبغي أن تدرج قبل كل عنصر مطبوع (أي أنها تتكرر بعدد العناصر المراد طباعتها). العنصر المراد طباعته في هذه الحالة عبارة عن سلسلة حرفية (string). تكون السلاسل الحرفية دائماً محصورة بين علامتي التنصيص " " في C++.

خامساً: يلاحظ أن السلسلة الحرفية في أمر cout أعلاه تنتهي بالرمزين "\n". الشرطة المائلة \ (slash) تستخدم كعلامة استثناء في أوامر الطباعة. إذا ظهرت علامة الإستثناء قبل رمزٍ ما في أوامر الطباعة، فهذا يدل على أن هذا الرمز ينبغي أن يعامل معاملة خاصة و يستثنى من الطباعة المباشرة. في المثال أعلاه تدل علامة الإستثناء على أن حرف n ينبغي ألا يطبع بصورته الظاهرة، وإنما يصار إلى معناه الخاص. المعنى الخاص في هذه الحالة هو الإنتقال إلى سطر جديد. علامة الإستثناء مع الرمز الذي يتلوها تعرف بالأزواج الإستثنائية (Escape sequences) في مصطلح C++. هناك الكثير من الأزواج الإستثنائية في لغة C++ نلخص بعضها في الجدول أدناه.

الجدول (1)

بعض الأزواج الإستثنائية في لغة C++

معناه	الزوج الإستثنائي
انتقل إلى سطر جديد	\n
انتقل أفقياً بما يعادل مسافة فقرة tab	\t
إرجع إلى بداية السطر	\r
إصدر صوتاً باستخدام جرس النظام	\a
طباعة علامة الشرطة المائلة	\\
طباعة علامة التنصيص	\"

سادساً: الأمر cout ليس هو الأمر الوحيد لإظهار البيانات في C++، ولكنه الأسهل استخداماً. هناك أوامر أخرى تقليدية يمكن التعرف عليها فيما بعد.

سابعاً: الأمر الثاني في البرنامج: return 0; يرجع القيمة 0 إلى بيئة التنفيذ، إشارةً إلى أن البرنامج نجح في مهمته. هذه الجملة استخدمت لأننا قررنا في بداية البرنامج أن الدالة main ينبغي أن ترجع قيمة صحيحة عند انتهائها، وذلك بتعريفنا لها ك:int main() . إذا استخدمنا void في التعريف، فلاداعي لاستخدام return. التقليد العام في C++ هو أن يرجع البرنامج القيمة 0 إذا نجح في مهمته، وأي قيمة أخرى إذا فشل.

أسئلة تقويم ذاتي



1. اكتب برنامجاً يطبع اسمك في ثلاثة سطور: الإسم الأول في السطر الأول، الإسم الثاني في السطر الثاني و الإسم الثالث في السطر الثالث.
2. بيِّن الخطأ في البرنامج التالي:

```
int Main()
{ Cout <<'This is a simple example program';
Return 0
}
```

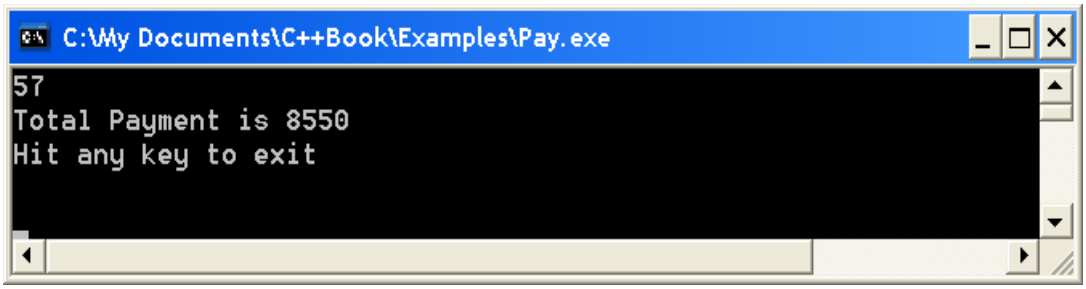
2.1 البرنامج الثاني

مثال (2)

ننتقل عزيزي الدارس الآن إلى مثال بسيط آخر بلغة C++، نضيف من خلاله بعض المعلومات الجديدة. المثال أدناه يوضح كيفية تعريف المتغيرات و الثوابت في C++، بالإضافة إلى أمر إدخال البيانات cin. المثال المذكور عبارة عن برنامج بسيط يحسب الأجر الذي يتقاضاه عاملٌ ما أسبوعياً بعد أن يقرأ عدد الساعات التي عملها العامل خلال الأسبوع HoursWorked ويستخدم في عملية الحساب الأجر في الساعة الواحدة RatePerHour

```
#include <iostream.h>
int main()
{
const float RatePerHour=150.0;
float HoursWorked,Payment;
cin >> HoursWorked;
Payment=HoursWorked * RatePerHour;
cout << "Total Payment is "<< Payment<<endl;
return 0;
}
```

الشاشة أدناه عينة من تنفيذ البرنامج



ملاحظات على البرنامج االثاني:

1. يبدأ البرنامج منطقة الإفتتاحية كسابقه، ثم عنوان البرنامج.
2. كلمة const في السطر الأول من البرنامج تستخدم لتعريف الثوابت في C++. الثابت عبارة عن قيمة لاتتغير داخل البرنامج. المقصود بتعريف الثوابت هنا هو "تسميتها" و إعطاؤها قيمة. عملية "تعريف" أو "تسمية" الثوابت تقليد جميل تنهجه C++ والكثير من لغات البرمجة. الفائدة التي نجنيها من تسمية الثوابت هي سهولة تعديل قيمتها في مكان واحد في البرنامج دون أن يضطر المبرمج إلى البحث عنها في ثنايا البرنامج. مثلاً، هب أن الثابت RatePerHour تغيرت قيمته من 150 إلى 200 في المثال أعلاه. في هذه الحالة ماعلينا إلا أن نغير قيمة الثابت في السطر الأول من البرنامج. أما إذا لم نعرف الثابت، فإننا سنتجشم البحث عن استخدامات الثابت داخل البرنامج. كلمة const هي إحدى الكلمات الخاصة في لغة C++.
3. كلمة float في السطر الثالث من البرنامج تستخدم لتعريف المتغيرات الحقيقية في لغة C++. أي متغير في C++ لابد من تعريفه قبل استخدامه. الفائدة من تعريف المتغيرات هو معرفة حجم الذاكرة المطلوبة لتمثيل المتغير. مثلاً أي متغير حقيقي يحتاج إلى أربع بايتات. تعريف المتغيرات يمكن أن يتم في أي منطقة داخل البرنامج. C++ تسمح باستخدام أنواع أخرى من المتغيرات منها البسيط ومنها المركب. كيفية تعريف المتغيرات في C++ سنعرض لها في الفقرات أدناه.

أسئلة تقويم ذاتي



1. بيِّن الخطأ في الجمل التالية:

```
cin <<quantity;
cin <<x,y,z;
cout <<My GPA is << Gpa;
cout <<m+n, p, q;
```

2. اكتب برنامجاً يقرأ ثلاثة أعداد a,b,c، ثم يحسب متوسطها ويطبع الناتج.

2. التعريف بقواعد لغة C++

يتكون أي برنامج في لغة C++ من عدة جمل، كل جملة تتكون من عدة مكونات نحوية . هناك نوعان من المكونات النحوية: مكونات طرفية (Terminals) و مكونات لاطرفية (Non-Terminals). المكونات الطرفية هي العناصر الأولية التي تتكون منها اللغة مثل الحروف الأبجدية، الأرقام، الأقواس ...الخ. المكونات اللا طرفية هي المكونات التي يمكن تعريفها كسلسلة مركبة من مكونات أخرى. تركيب الجمل في C++ تحكمه قواعد نحوية مفصلة لابد لكل مبرمج من معرفتها و فهمها. من أجل ذلك سنتعرض لمعظم هذه القواعد مستخدمين النهج التالي في تمثيلها للدارس .

الرموز اللا طرفية تدرج بين معكوفتين مثل : <program title> .

الرموز الطرفية تكتب كما هي، مثل الأقواس ( ) ، الفاصلة المنقوطة ; ... الخ.

الكلمات الخاصة تكتب بصورة مغمقة: مثل main float الخ.

الأقواس المربعة [ ] تدل على أن هذا الجزء من القاعدة اختياري ويمكن إدراجه أو حذفه أحيانا، حسبما يتطلب المقام.

مثلاً القاعدة المتعلقة بالشكل العام لجملة عنوان البرنامج تكتب كما يلي:

القاعدة (1)

الشكل العام لجملة "عنوان البرنامج":

<program header>::=[<type>] main ([<parameters>])

هذه القاعدة تقول أن جملة عنوان البرنامج تبدأ بالكلمة الخاصة main مسبوقة بعنصر اختياري [<type>] يحدد نوع القيمة الراجعة من البرنامج. كلمة main متبوعة بقوسين، بينهما مجموعة عناصر اختيارية. هذه القاعدة عادةً تتبعها قواعد أخرى تفسيرية توضح ما الذي نعنيه بالعناصر اللاطرفية <Type> و <Paramters>. عموماً أي عنصر لاطرفي يظهر في قاعدة لابد من تفسيره بقاعدة لاحقة (إن لم يكن تم تعريفه من قبل).

3. الهيكل العام لبرامج C++

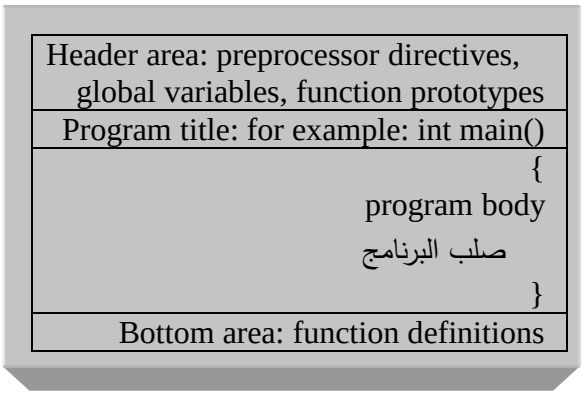
يتكون أي برنامج في لغة C++، أياً كان حجمه أو تعقيده، من الأجزاء التالية:

المنطقة الإفتتاحية (header area).

عنوان البرنامج.

صلب البرنامج (program body)

ذيل البرنامج.



عنوان البرنامج:

القاعدة العامة لجملة عنوان البرنامج تقدمت أعلاه (القاعدة 1). الذي نود أن نضيفه هنا هو أن نظام C++ يسمح بتمرير قائمة عناصر للبرنامج. في هذه الحالة لابد من إعطاء عدد العناصر ثم قائمة بأسمائها. الشكل العام للعنوان في هذه الحالة يكون بالصورة التالية:

int main(int argc, char* argv[])

هذه الصورة تستخدم عند تنفيذ البرنامج من لوحة الأوامر (command line). في هذه الحالة argc هي عدد العناصر على سطر الأوامر بما في ذلك اسم البرنامج، و []argv عبارة عن قائمة حرفية تحتوي على كل العناصر على سطر الأوامر (بما في ذلك البرنامج نفسه).

ليس من الضروري أن تفهم هذا حالياً. المهم أن تفهم أن هناك صورة أخرى لعنوان البرنامج. الصورة تتضح بكاملها بعد مناقشة المصفوفات و الأوامر في وحدات لاحقة إن شاء الله.

4. المتغيرات البسيطة في لغة C++ و كيفية تعريفها

المتغيرات الجبرية تعتبر عنصراً هاماً من عناصر البرمجة في C++ و قل أن يخلو منها برنامج. C++ لا تسمح باستخدام المتغيرات إلا بعد تعريفها و الإعلان عنها كما أسلفنا. أي متغير لم يتم تعريفه لا يسمح باستخدامه و يؤدي إلى خطأ في مرحلة الترجمة. لتعريف متغير معين لابد من إعطاء نوعه و اسمه. C++ تسمح بالعديد من أنواع المتغيرات البسيطة و المركبة. في هذه المرحلة نركز على أنواع المتغيرات البسيطة و نؤجل النظر في أمر المتغيرات المركبة إلى وحدات قادمة إن شاء الله.

هناك أربعة من أنواع المتغيرات البسيطة في C++ هي:

الجدول (2)

الأنواع البسيطة في لغة C++

مجاله	نوع المتغير في C++
الأعداد الصحيحة integer numbers (لا تحتوى على كسور)	int
الأعداد الحقيقية real numbers (قد تحتوى على جانب كسري)	float
القيم المنطقية Boolean values (true or false)	bool
الرموز والحروف character values	char

كما هو معروف من الجبر أن الأعداد الصحيحة لا تحتوي على كسور بينما تحتوي الأعداد الحقيقية على جانب كسري بعد الفاصلة العشرية. تأخذ المتغيرات التي تنتمي إلى النوع المنطقي bool إحدى القيمتين: true (صواب) أو false (خطأ). المتغيرات التي تنتمي إلى النوع char يمكن أن تكون قيمتها أي رمز من الرموز المعرفة في جدول رموز النظام (ASCII table).

يرجى ملاحظة أن C++ تتعامل مع عدة أحجام من الأعداد الصحيحة و الحقيقية. فمثلاً النوع long يستخدم لتعريف الأعداد الصحيحة التي يتطلب تمثيلها أربع بايتات في الذاكرة. يستخدم النوع double لتعريف الأعداد الحقيقية الطويلة التي يتطلب تمثيلها ثمان بايتات في الذاكرة. الجدول أدناه يعطي بعض الأنواع الإضافية في لغة C++:

الجدول (3)

بعض الأنواع الإضافية في لغة C++

النوع	معناه
short	عدد صحيح طوله 2 بايت
long	عدد صحيح طوله 4 بايت
unsigned short	مثل short ولكنه لايترك خانة للإشارة
unsigned long	مثل long ولكن ليس به خانة للإشارة
double	عدد حقيقي طوله 8 بايت

لتعريف متغير معين في لغة C++ ما علينا إلا أن نذكر نوعه ثم اسمه. مثلاً:

```
int StudentID;
float Salary;
char Grade;
```

القاعدة العامة لتعريف المتغيرات في غاية البساطة

القاعدة (2) :

<var-type 1> <var-list 1>; <var-type 2> <var-list 2> ;
.
<var-type i> <var-list> ;
<var-list> ➔ <var1>[, <var2>, …]

في هذه القاعدة بَيِّن إن المتغيرات تعرّف بإعطاء نوع المتغير متبوعاً بقائمة بأسماء المتغيرات التي تنتمي إلى هذا النوع، متبوعةً بالفاصلة المنقوطة. القاعدة تبين أيضاً أنه بالإمكان تعدد جمل تعريف المتغيرات في البرنامج (ليس بالضرورة في صدر البرنامج). المثال البسيط أدناه يوضح مانرمي إليه:

مثال (3)

```
int  x,y;
float  z;
cin >> x >>y>>z;
float result;
result = z*(x+y);
```

5. تسمية المتغيرات و الثوابت في C++

لغة C++ تعطي قدراً كبيراً من الحرية في اختيار أسماء المتغيرات و الثوابت، فمثلاً ليس ثمة حدٌ أقصى لعدد الرموز التي يتكون منها اسم المتغير أو الثابت. غير أن أسماء المتغيرات و الثوابت ينبغي أن تخضع للقواعد الآتية:

1) الرمز الأول من اسم المتغير لابد أن يكون حرفا أبجدياً (إنجليزيا بالطبع، إلا إذا عرّبنا C++).
2) بعد الرمز الأول يمكن استخدام أي عدد من الحروف الأبجدية و الأرقام في تسمية المتغير.
3) لا يجوز استخدام الفراغات في تسمية المتغيرات.
4) لا يجوز استخدام أي رموز غير الحروف و الأرقام في تسمية المتغيرات.
5) يمكن استخدام الشرطة التحتية underscore في تسمية المتغيرات كما هو الحال في Std_name.
6) C++ تميز بين الحروف الإنجليزية الكبيرة و الصغيرة فمثلاً Mean و mean متغيران مختلفان.

أمثلة لأسماء متغيرات مقبولة: Course5, Name, StudentName, x, Q3

أمثلة لأسماء متغيرات غير مقبولة: 9PQ Price$ p%

إدراج الملاحظات في برامج C++

يسمح للمبرمج بإدراج ملاحظات و تعليقات في برامجه وذلك لشرح بعض الخطوات أو توثيق البرنامج بحيث يسهل على الدارس فهمه و متابعة خطواته. هناك طريقتان لإدراج الملاحظات في برامج C++:

1- تستخدم الشرطتان المائلتان لإدراج الملاحظات التي تنتهي في نفس السطر. مثلاً:

```
// قطعة من برنامج لتوضيح امكانية تعريف المتغيرات في أي منطقة
 //declare x and y   int  x,y;
float  z;   //declare z
cin >> x >>y>>z;    //input x,y and z
float result;    //now declare result
result = z*(x+y);   //compute result
cout <<result;    // print the result
```

"الملاحظة" الأولى كتبت بالعربية في سطر مستقل. الملاحظات الأخرى أدرجت في سطر واحد مع الأوامر. كل ملاحظة تبدأ بعد الشرطتين المائلتين ولكنها تنتهي في النفس السطر.

1- إذا كانت الملاحظة تتطلب أكثر من سطر، فيمكن استخدام الطريقة الأخرى وهي استخدام شرطة مائلة مع النجمة في بداية و نهاية الملاحظة كما هو موضَّع في المثال أدناه:

مثال (4)

```
/* This is a very simple program that computes the employee payments, given
                                the Hours worked and the Rate Per Hour */
                                #include <iostream.h>
                                int main()
                                {
                                const float RatePerHour=150.0; //the rate per hour
                                float HoursWorked,Payment;
                                cin >> HoursWorked;   //get the number of hours worked
                                Payment=HoursWorked * RatePerHour;    //compute the payment
                                cout << "Total Payment is "<< Payment<<endl;   //print the result
                                return 0;
```

2- يتجاهل المترجم الملاحظات والتعليقات ولا يعتبرها جزءاً من البرنامج. لاحظ أننا استخدمنا اللغة العربية والإنجليزية في التعليقات التي أضفناها للبرنامج أعلاه، والكل جائز.

إن إدراج الملاحظات في البرنامج تقليد برمجيٌ حميد ينبغي الحرص عليه وممارسته. تساعد الملاحظات في فهم البرنامج و إزالة اللبس و الغموض، خاصةً إذا كان البرنامج جزءاً من نظام كبير يشارك في تطويره أكثر من شخص.

6. كيف يحصل البرنامج على البيانات

كما رأينا في الأمثلة أعلاه، يحصل البرنامج على البيانات باستخدام جملة cin. هذه الجملة يمكن استخدامها لقراءة عدد من العناصر مرةً واحدة و إن اختلفت أنواعها. مثلاً يمكن استخدام cin لقراءة قيمة متغير صحيح، ثم آخر حرفي، ثم آخر حقيقي ما هو موضح أدناه:

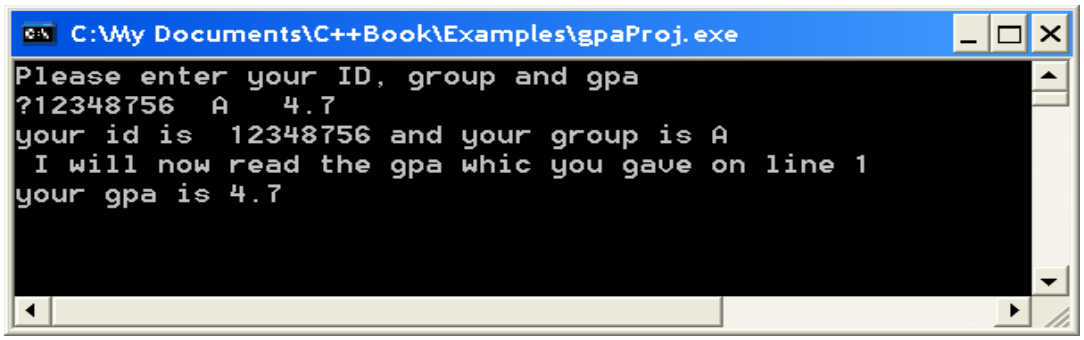
```
int id;    float gpa; char group;
cin >>id>>group>>gpa;
```

الأمر cin هو أحد أوامر "التمرير" (stream I/O) التي تنشئ مساراً افتراضياً بينها وبين جهاز الإدخال (لوحة المفاتيح في هذه الحالة)، ثم تأخذ البيانات عبر هذا الممر. كل مايوضع في هذا الممر تتلقفه cin. الأجمل من ذلك أن cin تتذكر موقعها في الممر وتعود إليه بعد انقطاع لتقرأ البيانات من حيث انتهى بها المقام. المثال أدناه يوضح ما نعني:

مثال (5)

```
#include <iostream.h>
int main(int argc, char* argv[])
{ int id;
char group;
float gpa;
//ask the user to give his id, group and GPA
cout <<"Please enter your ID, group and gpa\n?";
//now read the id and group only
cin >> id >>group;
// print out what you read
cout <<"your id is  "<<id<<" and your group is "<<group<<"\n";
//now read the gpa
cout <<" I will now read the gpa whic you gave on line 1 \n";
cin >>gpa;
cout <<"your gpa is "<<gpa;
return 0;
}
```

يطلب البرنامج أعلاه من المستخدم إدخال الرقم id، رمز المجموعةgroup والمعدل gpa، يقرأ الرقم و الرمز أولاً، يقوم ببعض الأعمال، ثم يعود ليقرأ المعدل. السيناريو بكامله معطى في شاشة تنفيذ البرنامج أدناه:



الشكل العام لجملة cin البسيطة نلخصه في القاعدة التالية:

القاعدة (3)

```
cin  >>var1 [>>var2 ……];
```

7. كيف يطبع البرنامج البيانات:

التحكم في شكل النتائج

يتم إظهار النتائج على الشاشة وطباعتها باستخدام جملة cout كما هو واضح من الأمثلة أعلاه. الشكل العام لأمر cout هو

```
cout <<element-1[<<element-2 ……<<element-n……];
```

حيث element-n يمكن أن يكون عنصراً قابلاً للطباعة كالمنغيرات و الثوابت، أو عاملاً تنظيمياً يستخدم في تنظيم النتائج وتشكيلها. هناك الكثير من الأوامر التي تستخدم في تشكيل النتائج المطبوعة في سي++، نذكر منها على سبيل المثال مايلي:

1- الدالة setw(n) تستخدم للتحكم في عرض المسافة التي تطبع فيها قيمة المتغير الذي يليها في جملة cout. مثلاً الجملة: cout <<setw(10)<<x; تطبع المتغير x في حقل عرضه عشر مسافات.

2- الدالة setioflags() تستخدم للتحكم في طباعة الأعداد بالصورة المناسبة. البرنامج أدناه يوضح بعض استخدامات هذه الدالة.

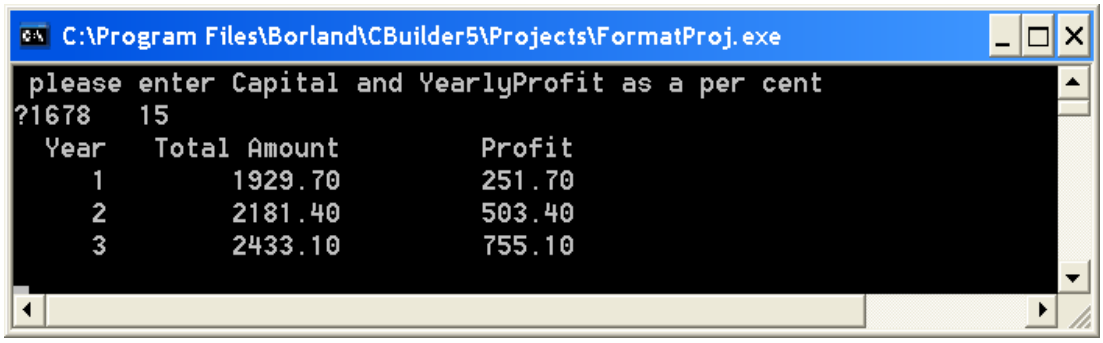
3- الدالة setprecision تستخدم للتحكم في عدد الخانات العشرية (عدد الأرقام بعد العلامة العشرية) عند طباعة الأعداد الحقيقية. استخدام هذه الدالة يوضحه البرناج أدناه.

المثال التالي يوضح كيف يمكن تشكيل الناتج باستخدام الدوال أعلاه:

مثال (6) 

```
#include <iostream.h>
#include <iomanip.h>
int main(int argc, char* argv[])
{ float Capital, YearlyProfit, TotalAmount, Profit;
  int Year;
  cout <<" please enter Capital and YearlyProfit as a per cent \n?";
  cin >>Capital >>YearlyProfit;
  //Print row headers
  cout <<setw(6)<<"Year"<<setw(15)<<"Total
Amount"<<setw(15)<<"Profit"<<endl;
  //set floating point number format
  cout << setiosflags(ios::fixed | ios::showpoint)<<setprecision(2);
  //compute profit and amount for year 1
  Year=1;
  Profit = Capital * Year*YearlyProfit/100;
  TotalAmount = Capital + Profit;
  //print the results
  cout <<setw(6)<<Year<<setw(15)<<TotalAmount
      <<setw(15)<<Profit<<endl;
  //Compute the profit for Year 2
  Year = 2;
  Profit = Capital * Year*YearlyProfit/100;
  TotalAmount = Capital + Profit;
  //print the results for Year 2
  cout <<setw(6)<<Year<<setw(15)<<TotalAmount
      <<setw(15)<<Profit<<endl;
   //Compute the profit for Year 3
  Year = 3;
  Profit = Capital * Year*YearlyProfit/100;
  TotalAmount = Capital + Profit;
  //print the results for Year 2
      cout <<setw(6)<<Year<<setw(15)<<TotalAmount
          <<setw(15)<<Profit<<endl;
      getche();
      return 0;
    }
```

عينة من تنفيذ البرنامج:



أسئلة تقويم ذاتي



اكتب برنامجاً يطبع الجدول الأسبوعي لمحاضرات طالب كمايلي:

Day	lect1	Lect2	Lect3	Lect4	Lect5
Sat	Phys	Math1	Eng	Chem	Bio

8. العمليات و الدوال الحسابية البسيطة في لغة C++

ننتقل الآن خطوة إلى الأمام وننظر إلى العمليات و الدوال الحسابية في Pascal. Pascal لغة جبرية تسمح بالتمثيل المباشر للتعابير الجبرية البسيطة و المعقدة باستخدام العمليات و الدوال البسيطة المبينة في الجدول أدناه:

الجدول (2)

الرمز	العملية
+	الجمع (ثنائية)
-	الطرح (ثنائية مثل x-y)، سالب (أحادية مثل –x)
*	الضرب (ثنائية مثل x*y)
/	القسمة الحقيقية (ثنائية مثل x/y)
%	باقي القسمة الصحيحة (ثنائية مثل M % N)

9. أسبقية العمليات الحسابية

يمكن استخدام العمليات أعلاه بالإضافة إلى الأقواس في تكوين تعبيرات جبرية كاملة مثل:

x+y/z r*(p+q) x+y-(p+q)/(w+z)*p

عند صياغة هذه التعبيرات الحسابية ينبغي مراعاة أسبقية تنفيذ العمليات والتي يمكن تلخيصها في القواعد البسيطة التالية:

1- العمليات المحصورة بين الأقواس ( ) تنفذ أولاً.

2- عمليات * / % تأتى في المرتبة الثانية وكلها في مستوى واحد.

3- عمليات + - تأتى في المرتبة الأخيرة، والإثنان في مستوى واحد من الأسبقية.

4- عند استواء الأسبقية، تنفذ العمليات التي على اليسار أولاً.

فمثلاً عند تقويم التعبير:

r*(p+q) + z/(y-x)

يبدأ الحاسوب بتنفيذ العمليات المحصورة بين الأقواس، مبتدئاً من جهة اليسار p+q ثم y-x.

بعد ذلك ينفذ عملية الضرب r*(p+q).

بعد ذلك ينفذ عملية القسمة z/(y-x).

ختاماً ينفذ عملية الجمع r*(p+q) + z/(y-x).

- لاحظ أننا إذا أزلنا الأقواس فإن الناتج من تقويم التعبير الجديد r*p+q + z/y-x يختلف

من الناتج من تقويم التعبير الأول. في هذه الحالة تنفذ العمليات حسب الترتيب التالي:

r*p
z/y
r*p+q
r*p+q+z/y
r*p+q+z/y-x

لعل الدارس قد اعتاد على استخدام العمليات الجبرية العادية مثل الضرب و القسمة ...الخ، ولكن قد يتساءل ما الجدوى من استخدام عملية باقي القسمة الصحيحة %. الجواب على هذا السؤال أن هذه العملية لها تطبيق واسع في مجالات علوم الحاسوب و في بعض فروع

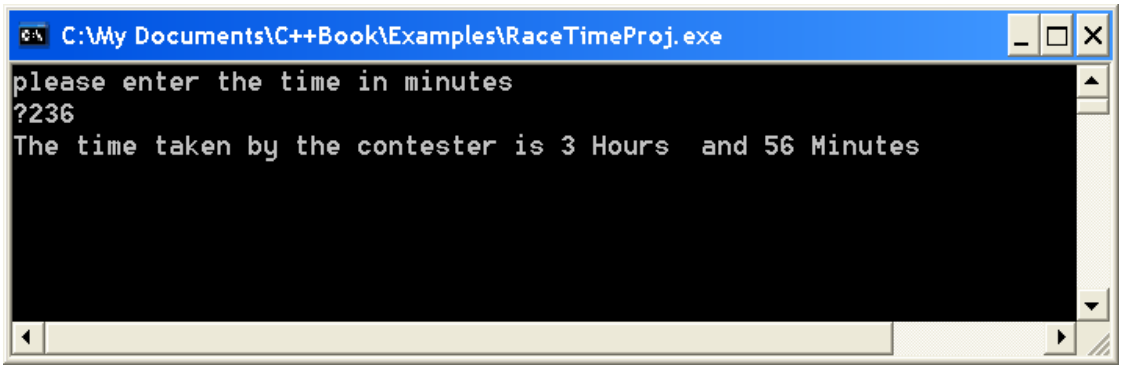
العلوم الرياضية. المثال البسيط التالي قد يقنعك عزيزي الدارس بجدوى هذه العملية على مستوى الاستخدام العادى:

مثال (7)

اكتب برنامجاً يقرأ الزمن الذي استغرقه أحد متسابقي الدراجات في سباق طويل المدى بالدقائق ويطبعه بصورة ساعات ودقائق.

```
#include <iostream.h>
int main(int argc, char* argv[])
{// program that computes the RaceTime in Hours and Minutes
int TimeInMinutes;
int Hours, Minutes;
cout <<"please enter the time in minutes\n?";
cin >>TimeInMinutes;
Hours = TimeInMinutes / 60;
Minutes = TimeInMinutes % 60;
cout <<"The time taken by the contester is " << Hours
<< " Hours  and "<< Minutes<< " Minutes";
return 0;
}
```

عينة من تنفيذ البرنامج



10. الدوال الجاهزة في لغة C++

تعطي C++ الكثير من الدوال الجاهزة التي يمكن استخدامها للكثير من الأغراض. هذه الدوال تأتي في ملفات مكتبية libraries خاصة. إذا أردنا استخدام هذه الدوال فلابد من إضافة الملفات الإفتتاحية الخاصة بها وذلك باستخدام الموجه #include الذي رأيناه في الأمثلة أعلاه. الجدول أدناه يعطي أمثلة على هذه الدوال، والمثال الذي يليه يشرح كيفية استخدام بعض هذه الدوال.

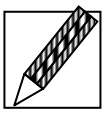
جدول (5): بعض الدوال البسيطة في C++

الدالة	أثرها
abs(x)	القيمة المطلقة للمتغير x
arctan(x)	الزاوية التي ظلها x (عكس ظل الزاوية)
cos(x)	جيب تمام الزاوية x
exp(x)	E مرفوعة للقوة x (e=2.178…)
log(x)	اللوغاريثم الطبيعى ل x
pow(x,y)	يرفع x للقوة y
sin(x)	جيب الزاوية x
sqr(x)	تربيع x
sqrt(x)	الجذر التربيعى ل x
ceil(x)c	تقرب x إلى أصغر عدد صحيح أكبر من x
floor(x)	أكبر عدد صحيح أصغر من x

11. المتغيرات وذاكرة الحاسوب

لعله من المفيد دائماً أن نتذكر أن أسماء المتغيرات في C++ و غيرها من لغات البرمجة ترمز إلى أماكن في ذاكرة الحاسوب. فعندما نعرِّف متغيراً صحيحاً مثلاً فإننا إنما نحجز بذلك مكاناً في ذاكرة الحاسوب يتناسب مع حجم القيمة المرتقبة لهذا المتغير. و عندما نسند قيمة معينة إلى متغيرٍ ما فإننا إنما نضع القيمة في المكان الذي يرمز له اسم المتغير.

تدريب (1)



1- هل يجوز تعريف المتغيرات في أي مكان في برامج C++ ؟ جرّب بنفسك.
2- هل يمكن طباعة الثوابت العددية مباشرةً باستخدام جملة cout، مثلاً
cout <<7; ؟
3- هل يجوز تعريف المتغير الواحد أكثر من مرّة في البرنامج ؟

12. إسناد القيم إلى المتغيرات:

جملة الإسناد Assignment Statement

في C++ تستخدم علامة الإسناد " = " لإسناد قيم إلى المتغيرات. ولعلك لاحظت أننا استخدمنا هذه العلامة في المثال الأول عندما حسبنا قيمة المتغير Payment:

```
Payment = HoursWorked * RatePerHour;
```

يقوم الحاسوب في الجملة أعلاه بحساب قيمة التعبير على يمين علامة الإسناد ثم يسند القيمة الناتجة إلى المتغير على يسار علامة الإسناد (أي يضع القيمة الناتجة في ذاكرة الحاسوب في المكان الذي يرمز له اسم المتغير على يسار علامة الإسناد). الجملة أعلاه مثال لجملة الإسناد في C++. لاحظ أن علامة الإسناد تختلف من علامة التساوي في C++ (علامة التساوي سنعرض لها في الفصل القادم إن شاء الله، وتكون دائماً مزدوجة "==")
الشكل العام لعبارة الإسناد في C++ هو:

قاعدة (5)

>اسم متغير< > =تعبير;<

حيث ال<تعبير> على يمين علامة الإسناد يمكن أن يكون ثابتاً، اسم متغير أو تعبيراً كاملاً تستخدم فيه العمليات الجبرية والدوال.

تختلف جملة الإسناد في معناها ومغزاها من المعادلات الجبرية التي عهدناها، فهي ليست معادلة ولكن عملية إسناد. (لذا استخدمت علامة الإسناد بدلاً من علامة التساوي). واحدة من الفروق الأساسية بين المعادلات و جملة الإسناد هو أنه لا يجوز استخدام الثوابت و لا التعبيرات الرياضية على الجانب الأيسر من علامة الإسناد. الفرق الآخر أن عملية الإسناد تتم على مرحلتين: المرحلة الأولى يتم فيها تقويم التعبير الواقع في الجانب الأيمن من علامة الإسناد. في المرحلة الثانية توضع هذه القيمة في المكان الذي يرمز له اسم المتغير في ذاكرة الحاسوب.

13. جملة الإحلال الذاتي

نتيجة للفرق الجوهري بين المعادلات و جملة الإسناد، فإنه يجوز أن يظهر اسم المتغير المسند إليه في التعبير الأيمن (التعبير المسند)، فمثلاً يجوز أن نكتب:

x:=x+5;

و تأويل ذلك أن القيمة الجديدة للمتغير المسند إليه x (على يسار علامة الإسناد) تصبح القيمة السابقة للمتغير x (على يمين علامة الإسناد) مضافاً إليها 5. يستخدم هذا النوع من الجمل كثيراً في برامج الحاسوب، وسنطلق عليه جملة "الإحلال الذاتي"، ذلك أن القيمة الجديدة تحل محل القيمة القديمة للمتغير في ذاكرة الحاسوب.

14. عوامل الإضافة و التنقيص المختصرة في لغة C++

Increment and Decrement Operators

C++ تعطي عاملين خاصين يمكن استخدامهما للزيادة الأحادية أو النقصان الأحادي (زيادة واحد أو نقصان واحد) لقيم المتغيرات. هذان العاملان هما: ++ و --. فمثلاً، بدلاً من أن نكتب:

x = x + 1; يمكن أن نستخدم x++; أو ++x ;

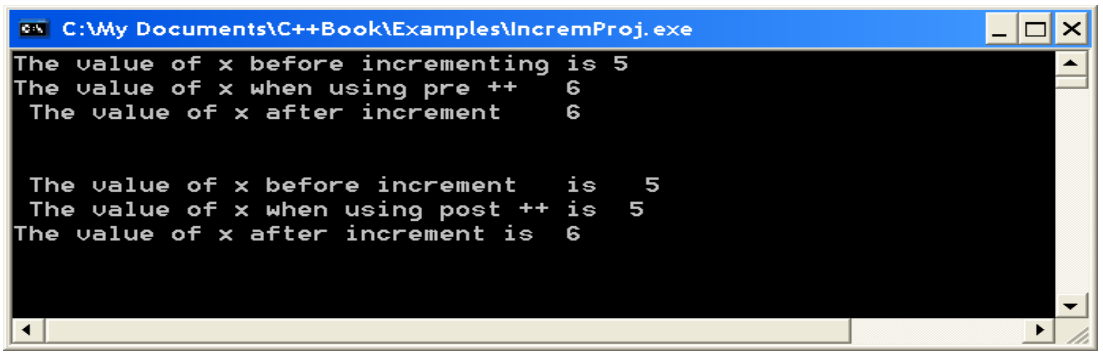
وبدلاً من:

x = x – 1; يمكن أن نستخدم x--; أو --x ;

نرجو أن تلاحظ إستخدام عامل الإضافة (النقصان) قبل المتغير أو بعده، كما هو الحال في x++ و ++x. السؤال الذي يطرح نفسه: هل هناك فرق بين الإستخدامين، وهل يؤثر ذلك على الناتج؟ الإجابة: نعم، في بعض الأحيان، إذا تمت العملية كجزء من جملة أخرى. المثال التالي يوضح الفرق بين الإستخدامين:

مثال (8)

```
#include <iostream.h>
int main(int argc, char* argv[])
{ int x;
  x = 5;
cout <<"The value of x before incrementing is " <<x <<endl;
cout <<"The value of x when using pre ++   " << ++x <<endl;
cout << " The value of x after increment    "<< x<<endl<<endl<<endl;
  x= 5;
cout <<" The value of x before increment   is   " << x<<endl;
cout << " The value of x when using post ++ is  " << x++<<endl;
cout << "The value of x after increment is  " << x<<endl;
return 0;
}
```



كما يبدو من الشاشة أعلاه، فإن استخدام معامل الزيادة القبلي ++x يؤدي إلى إجراء عملية الزيادة قبل طباعة القيمة. أما في حالة استخدام معامل الزيادة البَعدي فإن الزيادة تتم بعد طباعة القيمة.

أسئلة تقويم ذاتي

وضح ماهو الناتج من الأوامر التالية:

1- y = 10; x = 3; y = x++ + 7; cout << y<<" "<<x;

2- y = 10; x=3; y = ++x + 7; cout <<y<<" "<<x;

15. عوامل الإحلال الذاتي المختصرة

ذكرنا في فقرة سابقة أن جملة الإحلال الذاتي تعتبر حالة خاصة لجملة الإسناد يمكن استخدامها لتغيير قيمة متغير بالإضافة، النقصان أو غيرها من العمليات. فمثلاً، الجملة:

x = x + 7; تضيف 7 إل قيمة x وتضع الناتج في x.

والجملة:

x = x * 3; تضرب قيمة x في 3 و تضع الناتج في x.

C++ تعطي عوامل مختصرة يمكن استخدامها في هذه الحالات. فمثلاً، بدلاً من الجملة:

x = x + 7; يمكن استخدام x += 7;

المعامل الجديد هو =+.

الجملة x = x * 3; يمكن إعادة صياغتها باستخدام المعامل =* لتصبح x *=3;.

الجدول أدناه يعطي تلخيصاً لمعاملات الإحلال الذاتي في C++ و مغزى كلٍ منها:

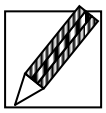
جدول (6)

عوامل الإحلال الذاتي

افرض أن القيمة المبدئية ل x هي 9 في كل حالة

العامل	مثال	المغزى	القيمة
+=	x+=5	x =x+5	14
-=	x-=3	x=x-3	6
*=	x*=2	x=x*2	18
/=	x/=3	x=x/3	3
%=	x%=7	x=x%7	2

تدريب (2)



1. بافتراض أنّ X,Y متغيرات حقيقية، N متغير صحيح، ch متغير ألفبائي، found متغير منطقي، بين ما هو الصحيح و ما هو الخطأ من جمل الإسناد التالية:

```
N = found ;                    N + 1 = 7 ;
X = N+ 3 ;                     ch= ' B + 1' ;
Y= X / N ;                     ch = ch + 5 ;
N = X * Y/7 ;                  found = ' true ' ;
ch = A ;
```

2. بافتراض أنّ X,Y,Z متغيرات حقيقية

X = 2 , Y = 3 , Z =4

M,N,P متغيرات صحيحة، NumCh متغير ألفبائي:

M = 8 , N = 5 ,

NumCh = '2'

أوجد قيم الجمل الآتية :

```
R = ( X + Y ) * 2 ;
R= ( X + Y / Z ) * 2 ;
R = X + Y / Z * 2 ;
R= M / N + 5 ;
P = M DIV N + 5 ;
P = M /N ;
P = SQR ( M ) / SQR ( N ) ;
Symbol = NumCh ;
Symbol = ' 4 ' ;
X = NumCh ;
P = round ( SQr ( M mod n ) / Z ) ;
M = NumCh +1 ;
```

3. بافتراض أنّ M، متغير صحيح ، اكتب جمل إسناد تفعل الآتي :
   - تزيد قيمة M بضعف قيمته.
   - تزيد قيمة M ب 10.
   - تضيف الرقم الأيمن من M إلى M.
   - تضيف إلى M الجزء الصحيح من قيمة المتغير الحقيقي X

4. بافتراض أنّ id متغير صحيح، Symbol متغير ألفبائي، Score متغير حقيقي، اكتب برنامجاً يقرأ المتغيرات أعلاه حسب الترتيب الآتي معطاة على سطرٍ واحد: id Symbol Scoreويطبعها أيضاً على سطرٍ واحد بحيث لا يستغرق id أكثر من أربعة أرقام ويقرِّب Score إلى أقرب خانتين عشريتين.

5. اكتب برنامجاً يقرأ عدد صحيحاً مكوَّناً من ثلاثة أرقام ثمّ يطبع رقم الآحاد ورقم العشرات ثمّ رقم المئات كل في سطرٍ منفصل.

الخلاصة

عزيزى الدارس،

إشتملت هذه الوحدة على التعريف بقواعد لغة C++ و الهيكل العام لبرامج لغة C++ وعمل تدريبات على إدخال برنامج التنمية.

كما احتوت الوحدة علىتدريبات عملية على طباعة البرامج و إدخال البيانات عبر لوحة المفاتيح وطباعة النتائج بعد إجراء بعض العمليات الحسابية و استخدام الدوال عليها وتعرفنا ايضاً اسناد القيم الى المتغيرات و الاحلال الذاتي وعوامل الاضافة و التنقيص المختصرة في لغة C++ بالاضافة لعوامل الاحلال الذاتي المختصرة في لغة C++ و تعتبر هذه البرمجيات نموذج للهيكل العام للبرامج في مختلف اللغات مع ملاحظة التغيير في قواعد اللغة المعينة التى تختلف من لغة إلى أخرى.

لمحة مسبقة عن الوحدة التالية

الوحدة التالية في هذا المقرر تحتوي على المتغيرات المنطقية وعمليات المقارنة وقواعد العمليات الحسابية و الشرطية و أساليب التفريغ و الاختبار و التكرار متخذين في ذلك لغة C++ وعليه نرجو منك عزيزي الدارس أن تكثر من عمل التدريبات على الحاسوب لإكتساب مهارات البرمجة.

مسرد المصطلحات

الأزواج الاستثنائية Escape sequences

هو استخدام علامة الاستثناء (\) مع رمز ما فى اوامر الطباعة . مثال (\n) لطباعة سطر جديد

مكونات طرفية Terminal:

المقصود منها العناصر الأولية التي تتكون منها اللغة مثل الحروف الأبجدية، الأرقام، الأقواس .... الخ.

مكونات لا طرفية Non-Terminals:

المقصود بها المكونات التى يمكن تعرفيها كسلسلة مركبة من مكونات اخرى من تركيب الجمل

جملة الاسناد Assignment statement

المقصود منها ان يضع الحاسوب القيمة الناتجة فى ذاكرة الحاسوب فى المكان الذى يرمز له اسم المتغير على يسار علامة الاسناد وذلك باستخدام علامة الاسناد جملة الاحلال . وهى ان تحل القيمة الجديدة محل القيمة القديمة للمتغير فى ذاكرة الحاسوب .

المصادر والمراجع

1) Deictl, H. & Deitel, P., **C++ How to Program.** Third Edition, Prentice Hall: 2001.
2) Savitch, W. **Problem Solving With C++: The Object of Programming**. Third Edition, Addison Wesley: 1999.
3) Friedman, F. and Koffman, E. **Problem Solving, Abstraction, and Design Using C++** (2 nd Edition). Addison Wesley: 1996.
4) Lippman, S. lajoie, J. **C++ Primer,** (3rd Edition). Addison Wesley: 1998.
5) Horstmann, C. **Computing Concepts With C++ Essentials.** John Wiley: 1997.
6) Stroustrup, B. **The C++ Programming Language.** (3 rd edition), Addison Wesley: 1990.
7) Ellis, M. and Bjarnc Stroupstrup, **The Annotated Reference Manual.** Addison Wesley: 1990.
8) Tcale, S. C++ IOS treams Handbook. Addison Wesley: 1993.
9) Meyers, S. **Effective C++: 50 Specific Ways to Improve Your Programs and Designs** (2 nd edition). Addison Wesley: 1996.
10) Meyers, S. **More Effective C++: 35 New Ways to Improve Your Programs and Designs.** Addison Wesley. 1996.
11) Schildt, H**. C++ From the Ground up.** Mc Graw Hill. 2003.
12) Savitch, W. **Absolute C++.** Addison Wesley: 2002.
13) Cogswell, J. **C++ all – in – one Desk Refcrence for Dummies.** 2002.


الوحدة الثالثة
التحكم في مسار البرنامج : التفريع والتكرار

محتويات الوحدة

الموضوع	الصفحة
المقدمة	**63**
تمهيد	63
أهداف الوحدة	64
1. التعبيرات المنطقية وعمليات المقارنة في C++	**64**
2. الجمل المركبة في C++	**66**
3. أساليب التفريغ و الاختيار في C++	**67**
1.3 جملة if البسيطة	67
2.3 جملة if-else	68
3.3 جملة if-else-if	69
4.3 جملة Switch التفريغية	71
4. التعابير الصحيحة والتعابير شبه الصحيحة	**72**
5. أساليب التكرار في لغة C++	**76**
1.5 التكرار باستخدام جملة For	76
2.5 الخروج من التكرار قبل تمامه	83
3.5 التكرار بإستخدام جملة While	85
4.5 التكرار بإستخدام جملة Do While	88
5.5 التكرارات المتداخلة	90
الخلاصة	**96**
لمحة مسبقة عن الوحدة التالية	**96**
مسرد المصطلحات	**97**
المصادر والمراجع	**98**

المقدمة

تمهيد

عزيزي الدارس،

مرحباً بك في الوحدة الثالثة (التحكم في مسار البرنامج) من مقرر أساليب البرمجة.

تعتبر البرامج التى تعرضنا لها مسبقاً (أحادية الاتجاه) بمعنى أن أوامرها تنفذ بصورة متتالية حسب ترتيبها ابتداءً من الأمر الأخير، دون تفريغ أو تكرار.

هذا النوع من البرامج لايفيد كثيراً في التطبيقات العملية لأنه يصعب تعميمه على عدد كبير من الحالات. مثلاً إذا أردنا أن نحسب الراتب الأسبوعي لأكثر من موظف في مثال – فالخيار الوحيد أمامنا في برنامج (1) هو أن نعيد تنفيذ البرنامج لكل موظف (لكل حالة)، وهي بلا شك عملية عقيمة و قليلة الجدوى. الأمثل من ذلك أن نزود البرنامج نفسه بأوامر و أدوات تمكنه من **تكرار** بعض الأوامر تلقائيا.

أما إذا أردنا أن نكتب برنامجاً يحسب عدد الطلاب الناجحين و عدد الطلاب الراسبين في اختبارٍ ما، فإن الأمر يحتاج بالإضافة إلى التكرار (أكثر من طالب)، إلى مقدرة على التمييز بين الراسب و الناجح. لكي يميز الحاسوب بين الراسب و الناجح لابد من إجراء "**مقارنة**" بين درجات الطلاب و درجة النجاح المتعارف عليها. باستخدام هذه المقارنة يستطيع الحاسوب التمييز بين الطلاب الذين استوفوا "شرط" النجاح و الطلاب الذين لم يستوفوا هذا الشرط. بناءً على ذلك لابد من تزويد البرنامج المعني **بأوامر أو أدوات شرطية** تمكنه من اتخاذ الإجراء اللازم عند استيفاء الشرط أو عدم استيفائه.

عزيزي الدارس، في هذه الوحدة نعرض بشيءٍ من التفصيل **أوامر التكرار و التفريع** في لغة C++ ونوضح كيفية استعمالها مستخدمين مجموعة من الأمثلة. نعرض لأساليب التفريع أولاً، ثم أساليب التكرار، ولكن قبل هذا و ذاك لابد من التعرض لأساليب وعمليات المقارنة في C++ وكيفية تكوين التعبيرات المنطقية والقواعد التي تحكم أسبقية العمليات الشرطية والحسابية.

أهداف الوحدة



عزيزي الدارس،

بعد دراستك لهذه الوحدة ينبغي أن تكون قادراً على أن:

- **تميز** بين التعبيرات المنطقية وعمليات المقارنة.
- **تضع** العمليات الشرطية والحسابية حسب قواعد الأسبقية.
- **تحلل** الجمل المركبة.
- **تكتسب** المهارات اللازمة لاستخدام أساليب التفريع والاختبار بتوظيف جملة If و switch في لغة C++
- **تعرف** طرق التكرار المختلفة For, while, do while

1. التعبيرات المنطقية وعمليات المقارنة في C++

Conditional expressions

الجمل الشرطية التي سنعرض لها في هذه الوحدة تتطلب استخدام تعابير شرطية أو منطقية تكون قيمتها النهائية إما "صواب" (true) أو "خطأ" (false). هذه التعابير إما أن تكون بسيطة تنطوي على عملية مقارنة واحدة فقط أو مركبة تنطوي على أكثر من مقارنة، يربط بينها أحد عوامل الربط المنطقية مثل and أو or. التعابير الشرطية يتم تركيبها باستخدام **عوامل المقارنة** والعوامل **المنطقية**. الجدول أدناه يبين هذه العوامل مع أمثلة لاستخدام كلٍ منها:

جدول (1)

العامل	معناه	مثال على الاستخدام
>	أكبر من	x>y
<	أقل من	x<y
==	يساوى	y==X
>=	أكبر أو يساوى	x>=y
<=	اقل أو يساوى	x<=y
<>	لا يساوى	x<>y
!	عكس	!x
&&	و	(a>b) && (b>c)
||	أو	(a>b) || (x<=y)

☜**ملاحظة هامة:** لاحظ الفرق بين عامل الإسناد = و عامل التساوي = = في C++.
الكثير من المبرمجين يخطئون في استخدام عامل الإسناد بدلاً من التساوي.

أسبقية العمليات الشرطية والحسابية:

الجدول أدناه يوضح أسبقية العمليات الشرطية والحسابية إذا اجتمعت في تعبير واحد:

جدول (2)

الأسبقية	العمليات
المستوى الأعلى	!
المستوى الثاني	*, /, %, &&
المستوى الثالث	+, - , ||
المستوى الأدنى	<, <=, >, >=, ==, <>

2. الجمل المركبة في C++

التعامل مع الجمل المركبة في البرامج و الخوارزميات شيءٌ طبيعي ينشأ من طبيعة المسائل التي نتعامل معها. فمثلاً كثيراً ما تكون لدينا الرغبة في تكرار مجموعة من الجمل أو تنفيذها إذا استُوفيَ شرطٌ ما. يركب هذا النوع من الجمل في C++ باستخدام الحاصرتين {}.

مثلاً الجملة المركبة التالية تهدف إلى تبديل قيمني x,y:

```
{
temp = x;
x = y;
y = temp;
}
```

الجمل المركبة سيكثر استخدامها في الأمثلة أدناه.

3. أساليب التفريع و الاختيار في C++

1.3 جملة if البسيطة

الشكل العام لجملة if البسيطة هو

قاعدة (1)

```
if ( <condition> ) شرطif (<
<statement>  جملة;<
```

حيث ال <شرط > إما أن يكون بسيطاً أو مركباً، و ال<جملة> إما أن تكون بسيطة أو مركبة.

لاحظ أن الشرط يجب أن يكون محصوراً بين قوسين. أمثلة على الاستخدام جملة If:

```
cout << "a is greater than b";   )a>b(if
if  (a>b)
  {
     temp=a;
     a=b;
     b=temp;
  }
```

لاحظ استخدام الجملة المركبة في المثال أعلاه.

```
if ((a>b) && (b>c))
   cout << "b is the middle number";
```

```
if  (!b)  cout <<  "b is false";
```

2.3 جملة if-else

الشكل العام لهذه الجملة هو:

القاعدة (2)

>) ifشرط (<

> جملة1;<

else

> جملة2;<

أسئلة تقويم ذاتي



1- اكتب جملة IF تسند القيمة true إلى المتغير المنطقي LeapYear إذا كان المتغير Year يقبل القسمة على أربعة وإلا فتسند القيمة false إلى المتغير LeapYear

أمثلة على الاستخدام جملة If-else

```
if (x>=0)
cout << "x is not negative";
else
cout << "x is negative";
if (x> 0)
{
pcount+1; = pcount
psum+x; = psum
}
else
{
ncount:=ncount+1;
nsum:=nsum+x;
}
```

3.3 جملة ... if-else-if

الشكل العام لهذه الجملة هو:

قاعدة(3)

if)>شرط<(> جملة1 ;<

else if)>شرط2<(> جملة2;<

else if) >شرط3<(> جملة3;<

....

....

> **else**جملة أخيرة;<

مثالٌ على استخدامها:

```
if (mark<60)  Grade:='F';
else if (mark<70) Grade:='D';
else if (mark<80)  Grade :='C';
else if (mark<90)  Grade:='B';
else Grade:='A';
```

القطعة من البرنامج أعلاه تحسب التقدير الذي حصل عليه الطالب في مادة معينة بناءً على الدرجة التي حصل عليها في المادة.

مثال (1)

شركة تعطي تخفيض لزبائنها بناءً على حجم مشترياتهم من الشركة. إذا كان حجم المشتريات اقل من 500 دينار، يكون التخفيض 5%، أما إذا كانت المشتريات أكثر من 500 دينار و أقل من 1000 دينار فإن التخفيض يكون 5%. بالنسبة للمشتريات من 1000 إلى 5000يكون التخفيض 7%، و 10% بالنسبة للمشتريات التي تزيد على 5000 دينار. اكتب برنامجاً يقرأ حجم المشتريات لزبون معين و يطبع سعر البيع للزبون بعد التخفيض.

تحليل المسألة

المعطيـات: المعطـى الـرئيس فـي هـذه المسـألة هـو حجـم المشـتريات بالـدينار للزبـون المعنـي: AmountPurchased

المطلوب: المطلوب هنا طباعة سعر البيع للزبون بعد التخفيض.

المتغيرات الوسيطة: نحتاج لمتغير وسيط واحد لحساب التخفيض Reduction لخصمه من ثمن الشراء الأصلي.

الخوارزمية

- في البداية نحسب التخفيض الذي يستحقه المشتري بناءً على حجم مشترياته:

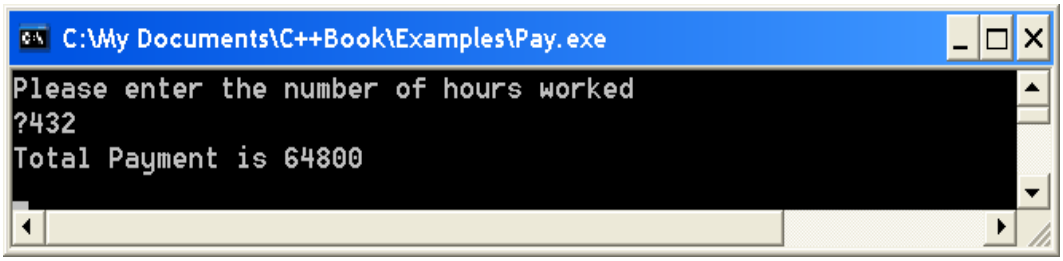
إذا كان حجم المشتريات أقل من 500 فإن التخفيض يكون 2×ثمن الشراء/100

أما إذا كان حجم المشتريات بين 500و 1000 فإن التخفيض يكون 3× ثمن الشراء/100، بعد ذلك نحسب ثمن الشراء الجديد: ثمن الشراء الجديد =ثمن الشراء الأصلي – التخفيض.

البرنامج

```
#include <iostream.h>
int main()
{ float  AmountPurchased, Reduction;
  cout <<"Please enter the amount purchased:  ";
  cin >> AmountPurchased;
   if  (AmountPurchased < 500)
      Reduction =2*AmountPurchased/100;
   else if (AmountPurchased < 1000)
      Reduction = 3* AmountPurchased/100;
   else if (AmountPurchased <5000)
      Reduction = 5* AmountPurchased/100;
   else if (AmountPurchased < 10000)
      Reduction = 7*AmountPurchased/100;
   else Reduction=10* AmountPurchased/100;
      AmountPurchased = AmountPurchased - Reduction;
   cout << "Prrice after reduction is = " << AmountPurchased <<endl;
   cout << "The Reduction is = " << Reduction<<endl;
   return 0;
```

عينة من تنفيذ البرنامج:



3.4 جملة switch التفريعية

إذا كان التفريع يترتب على القيمة الفعلية لتعبير صحيح (قيمته النهائية تنتمي إلى الأعداد الصحيحة)، أو تعبير شبه صحيح (يمكن تحويل قيمته النهائية إلى عدد صحيح)، فإنه بالإمكان في هذه الحالة استخدام جملة switch التفريعية، وهي تشبه جملة if المعممة إلى حد كبير إلا أنها تشترط أن يكون التعبير المستخدم من النوع الصحيح.

أسئلة تقويم ذاتي



1- بيِّن ماهو الخطأ في الجملة الآتية:

a- if (a=b) cout << "a and b are equal";

b- if (x <=0) then
cout << " x is not positive"
else cout<<" x is positive"

c- if (r>p && p> m) cout <<"r is greater m";

d- if (grade = 'A' | (grade = 'B') && (behavior = "excellent"))
cout <<"Prize";

2- اكتب برنامجاً يقرأ عددين x,y ثم يطبع الجملة: " العددان متفقان في الإشارة"، إذا كان العددان موجبين أو سالبين، و يطبع الجملة "العددان مختلفان في الإشارة" إذا كان الأمر غير ذلك (أحدهما موجب و الآخر سالب).

4. التعابير الصحيحة و التعابير شبه الصحيحة

التعابير الصحيحة هي التي تشتمل على متغيرات أو ثوابت صحيحة وتكون قيمتها النهائية صحيحة. التعابير شبه الصحيحة هي تعابير لاتشتمل على متغيرات أو ثوابت صحيحة مباشرةً، ولكنها تتكون من متغيرات أو ثوابت يمكن تحويلها إلى الأعداد الصحيحة. مثال ذلك:

1- التعابير الألفبائية char expressions .

2- التعابير المبنية على أنواع معددة enumerated types.

النوع 2 سنتعرض له في الوحدات القادمة إن شاء الله.

تدريب (1)



اكتب الناتج من تنفيذ هذا الجزء من البرنامج :

```
M = 4 ;
X= - 1.0 ;
If (M > 0)
If  (X  > 0)
cout << " First " ;
else
cout << "Second " ;
cout << " Third " ;
```

الشكل العام لجملة switch هو:

قاعدة (4)

```
switch (<تعبير>)
{
case <حالة1>:
        <جملة1>;
        break;
        ...
        …
case <حالة ن>:
        <جملة ن>;
        break;
default:
        <معالجة استثنائية>
}
```

الـ<تعبير> في هذه الحالة لابد أن يكون صحيحاً كما أسلفنا. كل حالة من الحالات 1 إلى ن عبارة عن حالة من حالات التعبير (أي قيمة محتملة للتعبير) المذكور في switch() والتي ينبغي أن تكون ثابتاً صحيحاً. إذا تعددت الحالات في جزءٍ واحد من أجزاء عبارة التحويل switch فلابد من تكرار كلمة case مع كل حالة. مثال ذلك:

```
switch (grade)
{ case ‘A’:
  case ‘B’:
      cout <<”you made it”<<endl;
      break;
  case ‘C’:
  case ‘D’:
  break;
      cout << “You almost made it, try again”<<endl;
  default:
       cout << “ You have a long way to go”<<endl;
}
```

لاحظ أننا نستخدم الأمر break بعد كل حالة من الحالات. القصد من استخدام break هنا هو تحاشي تنفيذ الحالات اللاحقة التي تأتي بعد الحالة المستوفاة. مثلاً إذا كانت قيمة المتغير grade في المثال أعلاه 'A'، **ولم نستخدم** break بين الحالات، فإن البرنامج سيطبع الجمل الثلاثة:

You made it
You almost made it, try again
You have a long way to go

بينما كان يجب أن يطبع الجملة الأولى فقط. هذه الظاهرة تسمى ظاهرة "السقوط الرأسي".

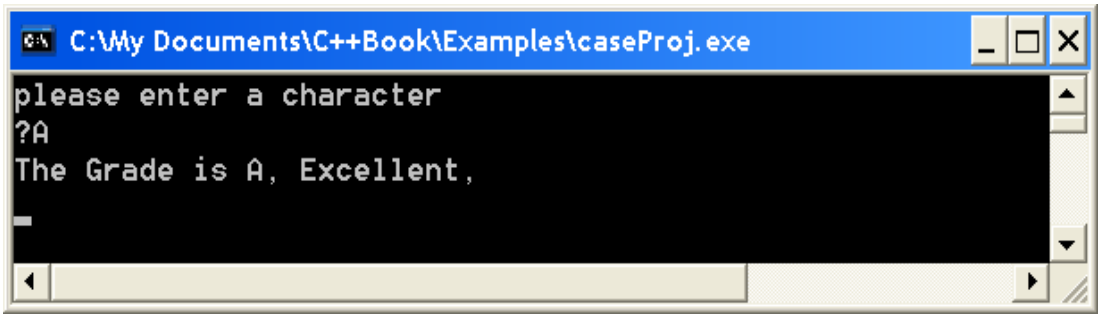
يرجى ملاحظة أن الجزء الإستثنائي default من جملة switch اختياري ويمكن حذفه. في هذه الحالة ينتقل التحكم إلى الجملة التي تلي جملة switch في البرنامج.



مثال (2)

```
#include <iostream.h>
int main(int argc, char* argv[])
{
char ch;
cout << "please enter a character\n?";
cin >> ch;
switch (ch)
{case 'A':
case 'a':
cout << "The Grade is A, Excellent, " <<endl;
break;
case 'B':
case 'b':
cout << "The Grade is B, Very Good …" <<endl;
break;
case 'C':
case 'c':
cout <<"You just made it, Try to improve"<<endl;
break;
default:
cout << "You are very weak, Try again";
break;  //optional
}
return 0;
}
```

عينة من تنفيذ البرنامج



☜ الهدف من هذا المثال هو قراءة حرف من لوحة المفاتيح ثم طبع رسالة على الشاشة بناءً على قيمة الحرف المدخل. لاحظ أن التعبير في هذه الحالة شبه صحيح (ألفبائي).

أسئلة تقويم ذاتي



1- بين ما هو الخطأ في الجمل التالية:

a- switch (a+b)

{ case 1, 7, 12:
cout <<"you win";
break;
case 3, 5,9:
break;
cout <<"you lost";
default:
cout << "try again";
}

b- switch (price)

case > 1000: cout <<"expensive";
case 500: cout <<"reasonable";
case <500: cout <<"cheap";

5. أساليب التكرار في لغة C++

عزيزي الدارس،

التكرار هو أحد العمليات التي يتقنها الحاسوب و يتفوق فيها على الإنسان. الحاسوب بإمكانه تكرار العديد من العمليات عدداً كبيراً من المرات دون كللٍ أو مللٍ أو خطأ. الأهم من ذلك أنه يفعل ذلك بسرعةٍ فائقة يصعب على الإنسان مضاهاتها. و لعل الدارس يذكر أن سرعة الحاسوب تقاس بملايين العمليات في الثانية الواحدة.

تهتم C++ كغيرها من لغات البرمجة بعملية التكرار في البرامج و توفر عدة أساليب لتمكين المبرمج من الاستفادة من هذا الأسلوب الفعال من أساليب البرمجة. أساليب التكرار في C++ تشمل:

1) التكرار باستخدام جملة for وتستخدم إذا كان عدد مرات التكرار معلوماً.
2) التكرار باستخدام جملة while، و تستخدم إذا كان عدد مرات التكرار غير معلوم وكان إنهاء التكرار ينبني على شرط معروف. جملة while تنظر في شرط التكرار قبل كل دورة من دورات التكرار: إذا استوفي الشرط أكملت الدورة و إلا فإن التكرار ينتهي.
3) التكرار باستخدام جملة do while، و هي تشبه جملة while في أن عدد دورات التكرار غير معلوم سلفاً و ينبني على استيفاء شرط معين، إلا أن الشرط هنا ينظر فيه بعد نهاية كل دورة من دورات التكرار بدلاً من بدايتها.

1.5 التكرار باستخدام جملة for

كما أسلفنا يستخدم هذا الأسلوب إذا كان عدد دورات التكرار معلوماً. الشكل العام لجملة for هو:

قاعدة (5)

for)>]]نوع>[<متغير التحكم> = <قيمة مبدئية>] ;[<شرط الإنتهاء ;[<
>]خطوة الإنتقال([<
> منطقة التكرار<

<متغير التحكم> هنا هو المتغير الذي يستخدم في التحكم في عدد دورات التكرار. جملة for تبدأ بالنظر في <القيمة المبدئية> لهذا المتغير، ثم تنظر في شرط الإنتهاء لترى ما إذا كانت القيمة المبدئية مستوفية للشرط أم لا؟ فإن كانت القيمة المبدئية مستوفية للشرط، بدأت التكرار و إلا انتقل مؤشر التنفيذ إلى الأمر الذي يلي منطقة التكرار. بعد كل دورة من دورات التكرار تستخدم جملة for <خطوة الإنتقال> في تعديل قيمة متغير التحكم ثم تكرر النظر في <شرط الإنتهاء> لترى إذا ماكانت القيمة الجديدة مستوفية للشرط أم لا. ما إذا كانت القيمة الجديدة مستوفية للشرط استمر التكرار و إلا توقف التكرار وانتقل مؤشر التنفيذ إلى الأمر الذي يلي منطقة التكرار.

<منطقة التكرار> هنا قد تكون جملة واحدة بسيطة أو جملة مركبة محصورة بين الحاصرتين {}. مثال على تكرار الجمل البسيطة:

```
for (int k=1; k<=10; k++)
cout << k <<”    “<< sqr(k);
```

في هذا المثال تستخدم جملة for في طباعة مربعات الأعداد من واحد إلى عشرة.متغير التحكم هنا هو k، وقيمته المبدئية هي 1. شرط الإنتهاء هو k<=10 وخطوة الإنتقال هي 1 (k=k+1) . لاحظ استخدام متغير التحكم كعنصر لدالة sqr. ++C، و إن كانت تسمح باستخدام متغير التحكم كعنصر مدخل للدوال و الروتينات أو في الجانب الأيمن من عبارات الإسناد إلا أنها تحظر تغيير قيمة هذا المتغير داخل منطقة التكرار. من هذا المنطلق لا يسمح باستخدام متغير التحكم في الجانب الأيسر من عبارات الإسناد أو كعنصر متغير في الدوال و الروتينات.

☜ لاحظ أننا عرفنا المتغير k داخل جملة for ، وليس بالضرورة أن يتم التعريف بهذه الصورة، ولكنه خيارٌ متاح. فمثلاً، بالإمكان تعريف k خارج الجملة بالصورة التالية:

```
int k;
for (k=1; k<=10; k++)
    cout << k <<”    “<< sqr(k);
```

أما تكرار الجملة المركبة فلا يختلف كثيراً عن تكرار الجملة البسيطة، إلا أننا في هذه الحالة نستطيع تكرار أكثر من جملة بسيطة محصورة بين الحاصرتين. مثال ذلك:

```
float kilos, miles;
for ( int k=1; k<15; k=k+1)
{
  cin >> miles;
  kilos=miles*8/5;
  cout << miles <<"        "<<kilos;
}
```

هذا الجزء من البرنامج يقرأ مسافات معطاة بالميل ثم يحولها إلى كيلومترات. العملية تتكرر 15 مرة، في كل مرة يقرأ البرنامج مسافة بالميل، يحولها إلى كيلومترات ثم يطبع الناتج. إذا لم نستخدم الحاصرتين فإن البرنامج يكرر الجملة البسيطة التي تلي جملة for فقط. أي أن البرنامج في هذه الحالة (إذا لم نستخدم الحاصرتين) سيقرأ 15 مسافة فعلاً و لكنه لن يحول إلا القيمة الأخيرة إلى كيلومترات. تأكد من ذلك بتنفيذ هذا الجزيء من البرنامج.

لعلك لاحظت أننا استخدمنا الأقواس المربعة [] حول مكونات جملة for، وهذا يدل على أن هذه المكونات اختيارية (اي أنه بالإمكان حذف كل هذه المكونات أو بعضها). مثلاً، البرنامج الجزئي أدناه يتكرر طالما أن قيمة j أقل من 10:

```
 j=1;
 for ( ; j<10 ;)
 { cout <<1/j <<endl;
   j = j +2;
 }
```

☜ لاحظ أننا في هذه الحالة حذفنا جملة القيمة المبدئية و خطوة الإنتقال من جملة for ووضعناها في أماكن أخرى. البرنامج الجزئي أدناه يتكرر يصورة لانهائية، إلا أن نتحكم فيه من الخارج:

```
 for ( ; ;)
{ cout << " I am infinite"<<endl;
  cout <<" you are in trouble, why did you start me in the first
place"<<endl;
  cout <<" to stop me press CTRL + C"<<endl;
}
```

في حالة حذف أحد مكونات جملة for فلابد من حفظ مكانه باستخدام الفاصلة المنقوطة التي تليه. المكون الأخير يحفظ مكانه بالقوس الذي يليه.

أمثلة تطبيقية

مثال (3)

اكتب برنامجاً يحسب الراتب الأسبوعي لمجموعة من الموظفين، علماً بأن الأجر الذي يتقاضاه الموظف في الساعة الواحدة هو 150 دينار وعدد الساعات التي عملها كل موظف خلال الأسبوع معطى لكل موظف مع رقمه الوظيفي. العدد الكلى للموظفين يتغير من فترةٍ إلى أخرى، و لكنه دائماً يعطى في بداية القائمة.

تحليل المسألة

- المطلوب في هذه المسألة هو طباعة جدول بالرواتب الأسبوعية لعدد من العمال. تطبع هذه الرواتب في شكل جدول من عمودين: العمود الأول يمثل الأرقام الوظيفية(EmployeeNo) للعمال(معطاة كأعداد صحيحة) بينما يمثل العمود الثاني الرواتب الأسبوعية WeeklyPayment (مطلوب حسابها كأعداد حقيقية) للعمال.
- المعطيات: يحتاج البرنامج أولاً العدد الكلى للعمال الذين يراد حساب الرواتب لهم NumOfEmployees (وهذا يعرّف في البرنامج كمتغير صحيح). بعد ذلك يحتاج البرنامج إلى رقم كل عامل (عدد صحيح) وعدد الساعات التي عملها كل عامل خلال الأسبوع NumOfHours (عدد حقيقي).
- المتغيرات الوسيطة والثوابت: من الواضح أن البرنامج سيكرر بعض العمليات، و عليه لابد من تعريف متغير للتحكم k (متغير صحيح). يحتاج البرنامج كذلك لمعرفة الأجر في الساعة RatePerHour، وقد أعطي كثابت في و صف المسألة.

تصميم الخوارزمية

الخوارزمية التي تستخدم في حل المسألة في غاية البساطة و يمكن تلخيصها كما يلى:

كرر لكل الموظفين:

1- اقرأ رقم الموظف و عدد الساعات التي عملها.

2- احسب الراتب الأسبوعي حسب العلاقة

WeeklyPayment:=NumOfHours*RatePerHour

3- اطبع رقم الموظف و الراتب الأسبوعي.



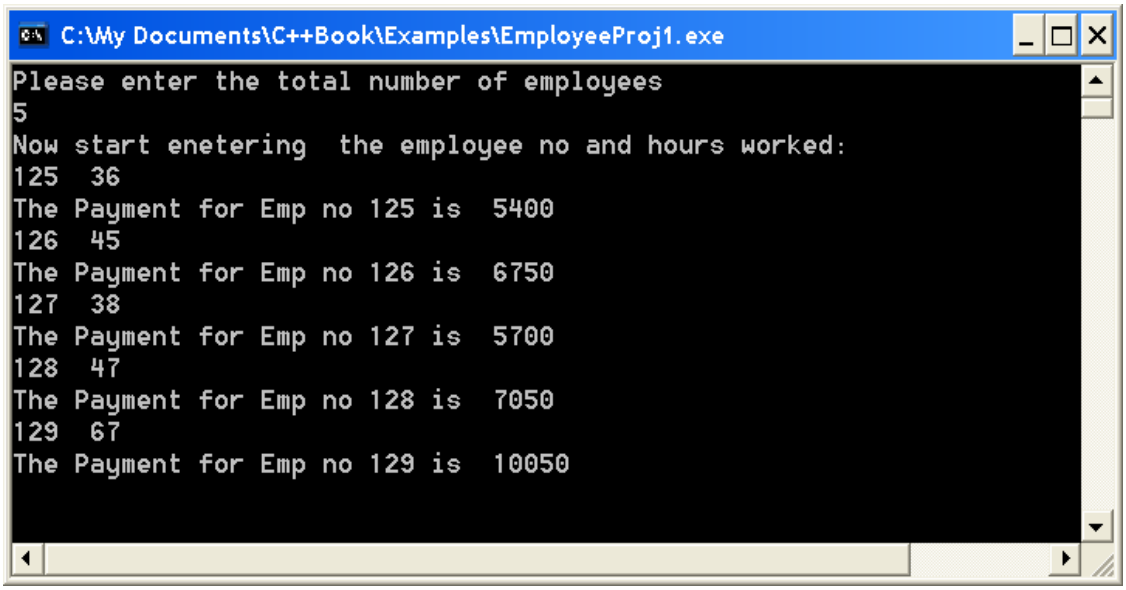
البرنامج

بناءً على ما تقدم يمكن كتابة البرنامج كما يلي:

```
#include <iostream.h>
int main(int argc, char* argv[])
{ const RatePerHour=150;
int EmployeeNo, k,NumOfEmployees;
float WeeklyPayment, NumOfHours;
cout << "Please enter the total number of employees"<<endl;
cin >> NumOfEmployees;
cout << "Now start enetering  the employee no and hours worked:"<<endl;
for (k=1; k<=NumOfEmployees; k=k+1)
{
cin >>EmployeeNo>> NumOfHours;
WeeklyPayment = NumOfHours * RatePerHour;
cout << "The Payment for Emp no " <<EmployeeNo
<< " is  " << WeeklyPayment<<endl;
}
getch();
return 0;
}
```

```
program ComputePayments(input, output);
uses WinCrt;
CONST
RatePerHour=150;
VAR
EmployeeNo, k,NumOfEmployees: integer;
WeeklyPayment, NumOfHours: Real;
Begin
Writeln('Please enter the total number of employees:');
Readln(NumOfEmployees);
writeln('Now start enetering  the employee no and hours worked:');
For k:=1 to NumOfEmployees do
Begin
Readln(EmployeeNo, NumOfHours);
WeeklyPayment := NumOfHours * RatePerHour;
Writeln('The Payment for Emp no ',EmployeeNo,' is ', WeeklyPayment:6:2);
End;
End.
```

عينة من تنفيذ البرنامج:



مثال (4)

اكتب برنامجاً يقرأ الدرجات التي حصل عليها الطلاب في اختبارٍ ما، ثم يحسب متوسط هذه الدرجات. عدد الطلاب يعطى في بداية القائمة.

تحليل المسألة

- المطلوب في هذه المسألة هو حساب المتوسط الحسابي AverageMark (عدد حقيقي) لمجموعة من الدرجات Mark التي حصل عليها الطلاب في اختبارٍ ما وطباعة هذه القيمة.
- لكي نحسب المتوسط لابد من حساب مجموع الدرجات MarkSum (عدد حقيقي) أولاً.
- العدد الكلي للطلاب N يعطى في بداية القائمة.
- نحتاج لمتغير وسيط للتحكم في التكرار.

تصميم الخوارزمية

في حل هذه المسألة نستخدم أسلوب الجمع التراكمي في حساب مجموع الدرجات. أسلوب الجمع التراكمي يستغل جملة الإحلال الذاتي التي تعرضنا لها في الفصل الثالث:

```
Readln(Mark);
MarkSum:=MarkSum + Mark
```

وتعني أن القيمة الجديدة لـMarkSum تصبح القيمة السابقة مضافاً إليها قيمة Mark الجديدة. بتكرار الجملتين أعلاه نستطيع أن نحسب المجموع الكلي للدرجات. بناءً على ذلك يمكن كتابة الخوارزمية على النحو التالي:

1- إقرأ العدد الكلي للدرجات N.

2- صفِّر قيمة المتغير MarkSum.

3- من 1 إلى N كرر الآتي:

- إقرأ قيمة Mark.
- أضف قيمة Mark إلى ما هو موجود في المتغير MarkSum.

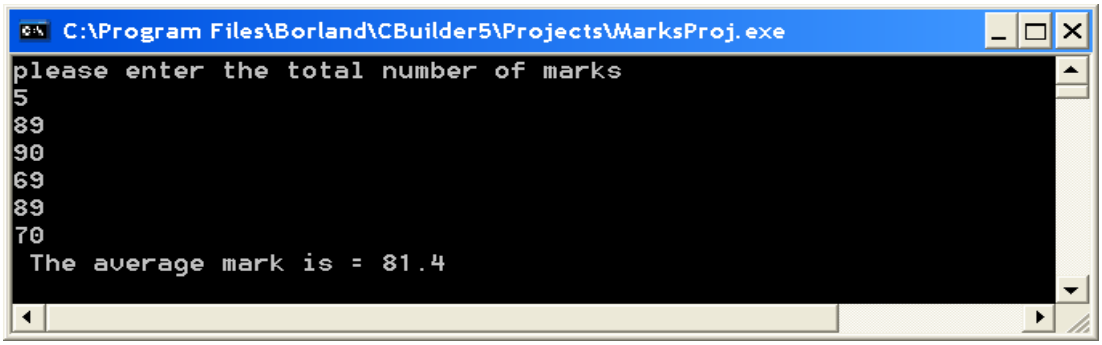
4- أحسب المتوسط AverageMark := MarkSum / N.

5- اطبع المتوسط.

البرنامج

```
#include <iostream.h>
int main(int argc, char* argv[])
{  float Mark, MarkSum, MarkAverage;
int i, n;
MarkSum=0.0;
cout << "please enter the total number of marks"<<endl;
cin >> n;
for (i=1; i<=n; i++)
{ cin >>Mark;
MarkSum = MarkSum + Mark;
}
MarkAverage = MarkSum / n;
cout << " The average mark is = "<< MarkAverage;
getche();
return 0;
}
```

عينة من تنفيذ البرنامج:



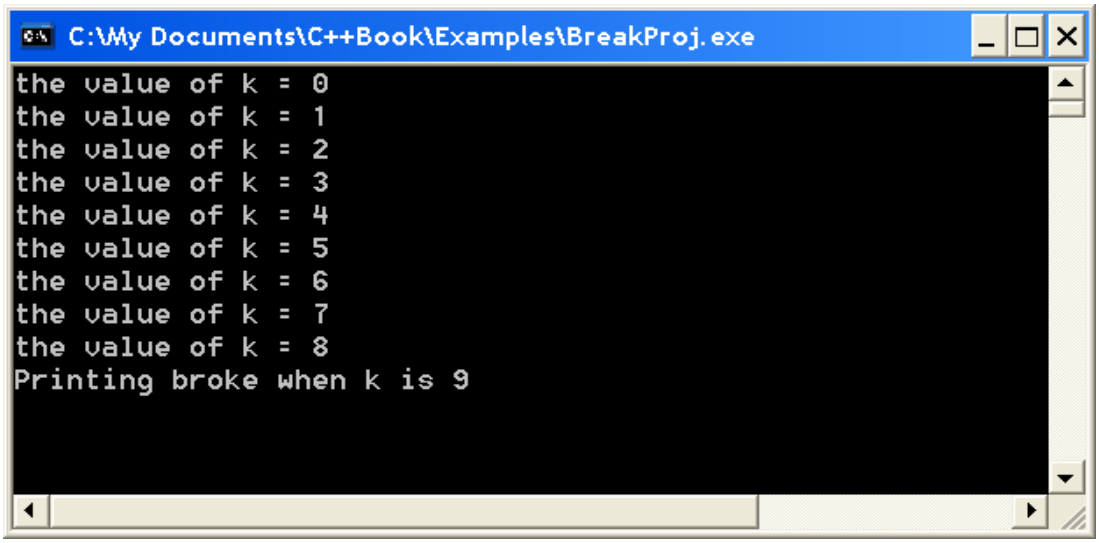
2.5 الخروج من التكرار قبل تمامه وتجاوز بعض الخطوات

هناك بعض التطبيقات التي تتطلب إيقاف التكرار إذا استوفي شرطٌ ما، كما أن هناك حالات تتطلب تجاوز بعض الخطوات في سلسلة التكرار. C++ تعطي أمرين مفيدين يساعدان في ذلك: break الذي يستخدم في إيقاف التكرار و الخروج منه قبل إتمامه (و قد رأينا استخدام beak من قبل لتحاشي ظاهرة السقوط الراسي في جملة switch)، و continue الذي يستخدم في تجاوز بعض الخطوات. المثالان أدناه يوضحان استخدام هذين الأمرين:

مثال (5)

برنامج يتوقف عن طباعة النتائج عندما تصبح قيمة k=9:

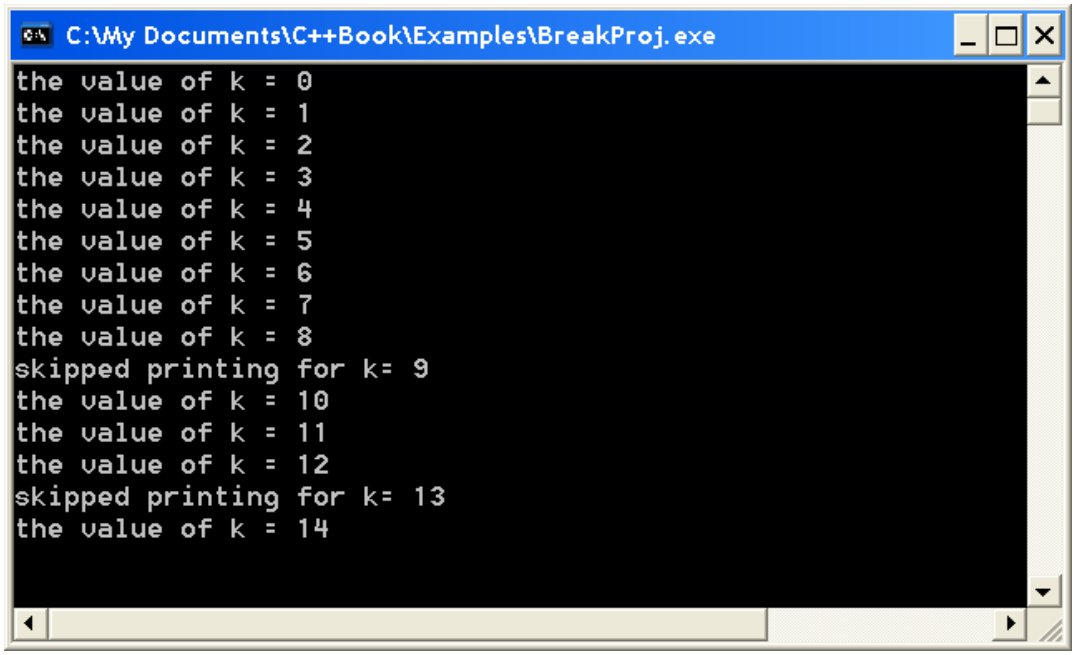
```
#include <iostream.h>
int main(int argc, char* argv[])
{ int k;
for (k=0; k<15; k++)
{ if (k==9) break;
cout <<"the value of k = "<<k<<endl;
}
cout <<"Printing broke when k is "<<k<<endl;
return 0;
}
```



مثال (6)

برنامج يتجاوز الخطوة 9 و الخطوة 13 عند طباعة الأعداد من 0 إلى 14:

```
#include <iostream.h>
int main(int argc, char* argv[])
{ int k;
for (k=0; k<15; k++)
{ if ((k==9)|| (k==13))
{  cout << "skipped printing for k= "<<k<<endl;
continue;
}
cout <<"the value of k = "<<k<<endl;
}
return 0;
}
```



أسئلة تقويم ذاتي



– اشرح باللغة العادية مالذي ترمي إليه القطعة التالية من البرنامج

و هو الناتج المتوقع في كل حالات المتغيرين Hon و prize :

```
if   (Hon == true)
if   (prize == true)
        Good =  true
              else
        Good = false
else if  (Hon = false)
        Good = false;
```

3.5 التكرار باستخدام جملة while

لعلك لاحظت أن جملة for تتطلب أن يكون عدد مرات التكرار معلوماً قبل البدء في التكرار، وهذا أمرٌ قد لا يتسنى في كثير من التطبيقات. C++ تعطي وسيلة أخرى للتكرار إذا لم يكن عدد مرات التكرار معلوماً في البداية. هذه الوسيلة هي جملة while.

جملة while تتطلب فقط أن يكون شرط التوقف معلوماً في البداية وتكون مفيدة في حالة البيانات التي تنتهي بعلامة خاصة sentinel ولا يعرف عددها في البداية. الشكل العام لجملة while هو:

```
while (<شرط التوقف>)
{
<منطقة التكرار>
}
```

المثال التالي يوضح كيفية استخدام جملة while:

مثال(7)

اكتب برنامجاً يقرأ الدرجات التي حصل عليها طلاب مادة البرمجة في الإختبار النهائي ثم يحسب متوسط هذه الدرجات ويطبعه. عدد الطلاب في شعبة البرمجة غير معروف ويتغير من فترة لأخرى، إلا أن قائمة الدرجات المعطاة تنتهي برقم سالب يدل على نهاية البيانات.

الخوارزمية

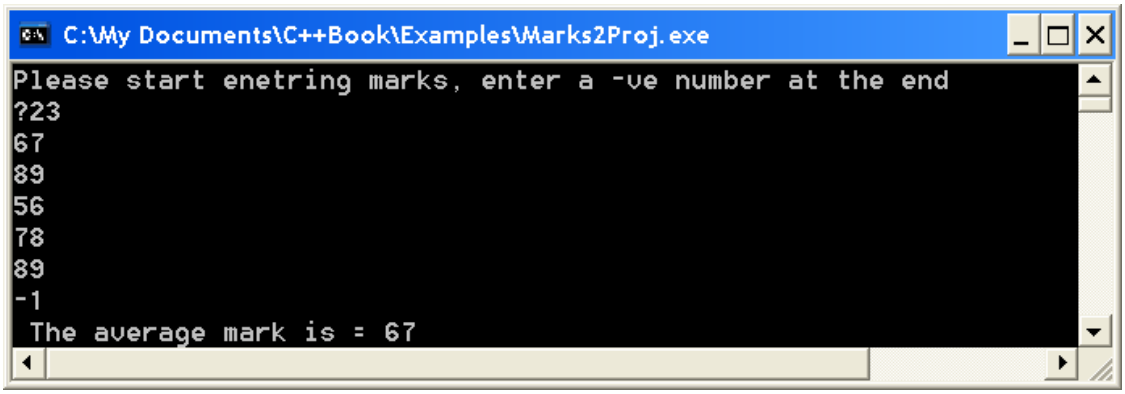
متوسط الدرجات هو المجموع الكلي للدرجات مقسوماً على عدد الدرجات. عدد الدرجات هنا غير معلوم وعلينا أن نحسبه من واقع البيانات. لهذا الغرض نستحدث متغيراً اسمه count. تتوقف عملية قراءة البيانات إذا وجدنا عدداً سالباً. يمكن تصميم الخوارزمية على النحو التالي:

1- صفِّر العداد count ومتغير الجمع MarkSum.
2- ابدأ بقراءة القيمة الأولى من الدرجات Mark.
3- طالما Mark>0 كرر الآتي
   1) أضف 1 إلى العداد
   2) أضف Mark إلى MarkSum.
   3) اقرأ قيمة جديدة من الدرجات Mark.
4- أحسب المتوسط MarkAverage = MarkSum / count.
5- اطبع الناتج.

البرنامج

```
#include <iostream.h>
int main()
{  float Mark, MarkSum, MarkAverage;
int i,count;
MarkSum=0;
count=0;
    cout << "Please start enetring marks, enter a -ve number at the
end\n?";
cin >> Mark; //read first mark
while (Mark>0)
{ count=count + 1;
MarkSum = MarkSum + Mark;
cin >>Mark;
}
MarkAverage = MarkSum / count;
cout << " The average mark is = " <<  MarkAverage<<endl;
getche();
return 0;
}
```

عينة من تنفيذ البرنامج:



ملاحظات على البرنامج و الخوارزمية أعلاه

لاحظ أن البيانات المدخلة إلى البرنامج أعلاه تتكون من الدرجات الفعلية متبوعة في النهاية بقيمة سالبة تُعلن انتهاء قائمة الدرجات. هذه القيمة السالبة تسمى الحارس أو الحاجز

sentinel في مصطلح المبرمجين. من المهم جداً ألا يخطئ البرنامج فيدخل الحاجز في المجموع. لكيلا يحدث ذلك ينبغي أن ترتب الأوامر بحيث ينتهي التكرار فور ظهور هذه القيمة وقبل أن تضاف للمجموع. من هذا المنطلق جاء ترتيب الأوامر داخل منطقة التكرار على النحو التالي:

```
count=count + 1;
MarkSum = MarkSum + Mark;
cin >> Mark;
```

هذا الترتيب يضمن ألا يحسب الحاجز في العداد ولا يضاف إلى المجموع. ولكي يبدأ التكرار لابد من قراءة القيمة الأولى من الدرجات قبل بداية التكرار لتكون بمثابة الإذن لدخول منطقة التكرار.

4.5 التكرار باستخدام جملة do while

جملة while تتطلب النظر في شرط التكرار قبل البدء في التكرار. في المثال السابق اضطررنا لقراءة القيمة الأولى قبل الدخول في التكرار، إذ لم يكن بالإمكان معرفة صحة الشرط إذا لم نفعل ذلك. إذا لم يكن بالإمكان النظر في الشرط قبل البدء في التكرار، فثمة حلٌ آخر في لغة C++ وهو جملة اللجوء إلى استخدام do while. حسب القاعدة التالية:

قاعدة (7)

الشكل العام لجملة do while :

```
do
{       منطقة التكرار
} while ( شرط التوقف);
```

شرط التوقف هنا عبارة عن تعبير منطقي بسيط أو مركب.

مثال على استخدمها:

```
x:=0.0;
do
{   cout << x <<"        "<< exp(x)<<endl;
x=x + 0.5;
}
while (x<=10);
```

البرنامج الجزئي أعلاه يطبع قيم exp(x) للمتغير x من صفر إلى 10 مفرقةً بفاصل 0.5 في شكل جدول مبسط.

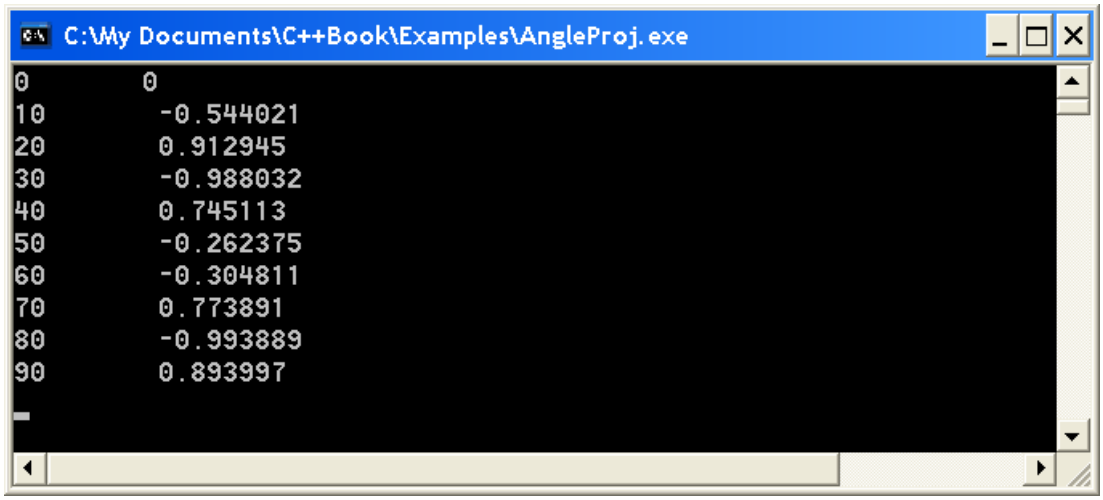
مثال (8)

اكتب برنامجاً يطبع جدولاً لجيب الزوايا من 0 إلى 90، كل 10 درجات.

البرنامج

```
#include <iostream.h>
int main(int argc, char* argv[])
{ float angle;
  angle=0;
  do
    {cout << angle <<"       "<<sin(angle)<<endl;
     angle = angle + 10;
    }
   while ( angle<=90);
   getche();
 return 0;
}
```

عينة من تنفيذ البرنامج



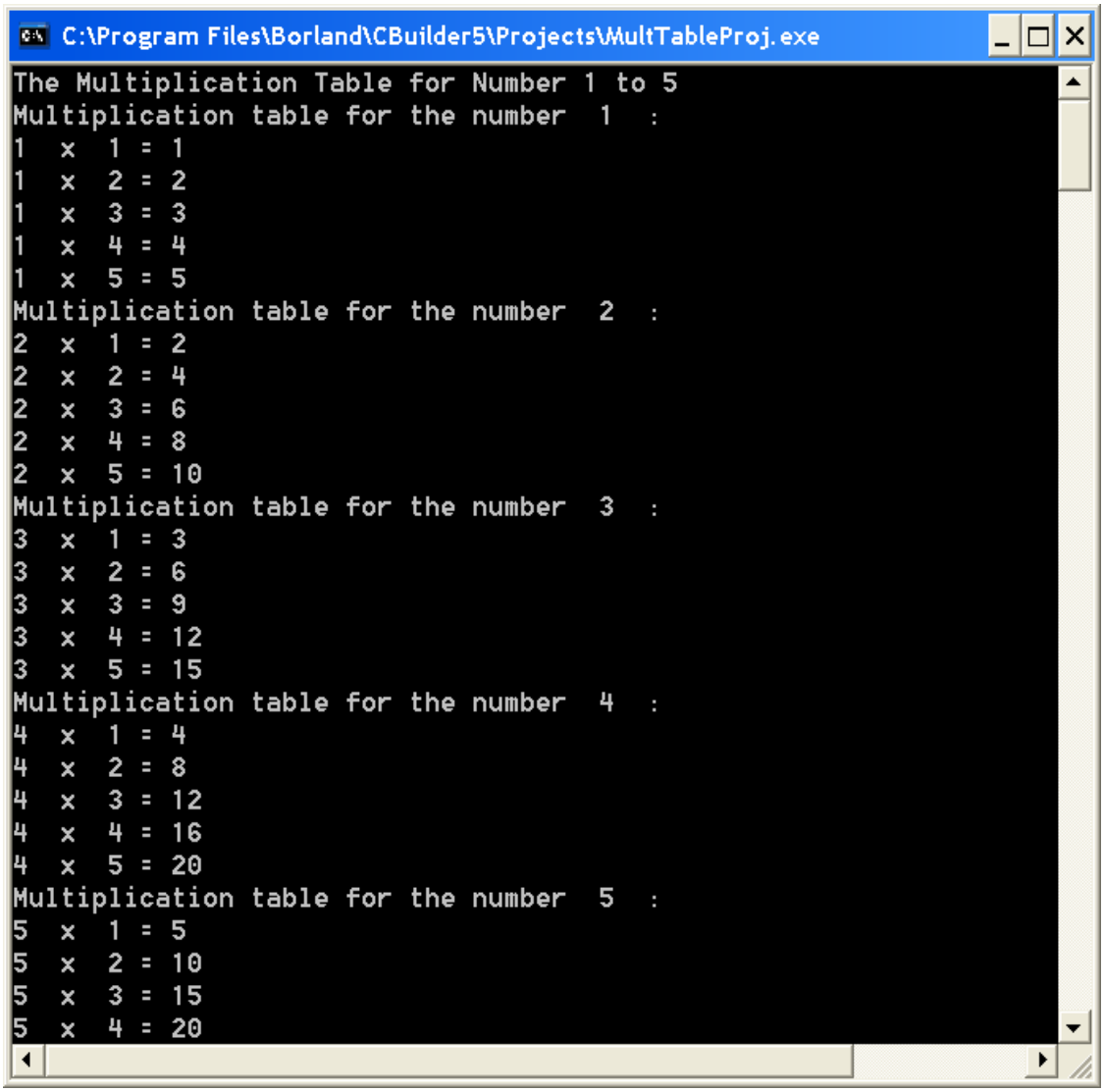
5.5 التكرارات المتداخلة

تسمح لغة C++ بتداخل التكرارات بحيث يتم تكرار منطقة معينة في البرنامج بناءً على عدة جمل تكرار متداخلة. مثلاً جملتا التكرار أدناه تطبعان جدول الضرب للأعداد من 1 إلى 5. جدول الضرب لكل عدد يشمل حاصل ضرب العدد في الأعداد حتى 9.

مثال (9):

```
#include <iostream.h>
int main(int argc, char* argv[])
{ int i,j;
cout <<"The Multiplication Table for Number 1 to 5"<<endl;
for(i=1; i<=5; i++)
{ cout <<"Multiplication table for the number  "<< i << "  :"<<endl;
for (j=1; j<=5; j++)
cout <<i <<"  x  " <<j <<" = "<<i*j<<endl;
}
return 0;
}
```

عينة من تنفيذ البرنامج



مثال (10)

اكتب برنامجاً يقرأ الدرجات التي حصل عليها الطلاب في خمسة مواد من ملف ويطبع جدولاً بالتقديرات التي حصل عليها كل طالب في هذه الخمس مواد علماً بأن التقديرات تحسب حسب الجدول الآتي:

00 – 59 → F
60 – 69 → D
70 – 79 → C
80 – 89 → B
90 – 100 → A

أفرض أن العدد الكلي للطلاب غير معلوم وأن القائمة تنتهي بقيمة سالبة.

تحليل المسألة

سبق أن تعرضنا لهذه المسألة وتم حلها لطالب واحد. المطلوب هنا تعميمها لعدد من الطلاب وعدد من المواد.

المطلوب: حساب التقدير Grade الذي حصل عليه كل طالب في كل مادة من المواد الخمس.

المعطيات: درجات الطلاب Mark في كل مادة من المواد الخمس.

المتغيرات الوسيطة: نحتاج لمتغير وسيط للتحكم في التكرار الداخلي (هناك تكراران متداخلان أحدهما معلوم العدد والآخر يتوقف على وجود حاجز في نهاية البيانات). نحتاج لمتغير id يستخدم لقراءة أرقام الطلاب.

تصميم الخوارزمية

بما أن عدد الطلاب الكلي غير معروف فإن توقف التكرار يعتمد على وجود حاجز في نهاية البيانات. نفترض أن هذا الحاجز يمثل قيمة شاذة لرقم الطالب id. يتوقف التكرار إذا أدخلنا قيمة سالبة للرقم id. بناءً على ذلك تصمم الخوارزمية كما يلي:

1- إقرأ رقم الطالب الأول من الملف.
2- طالما رقم الطالب id>0 كرر الآتي:
3- اطبع رقم الطالب في سطر جديد.
لكل مادة من المواد الخمس كرر الآتي:
إقرأ الدرجة التي حصل عليها الطالب في المادة Mark

إذا كانت Mark<60 فإن التقدير يكون F

وإلا إذا كانت Mark<70 فإن التقدير يكون D

وإلا إذا كانت Mark<80 فإن التقدير يكون C

وإلا إذا كانت Mark<90 فإن التقدير يكون B

وإلا فإن التقدير يكون A

اطبع التقدير في نفس السطر الذي طبعت فيه رقم الطالب.

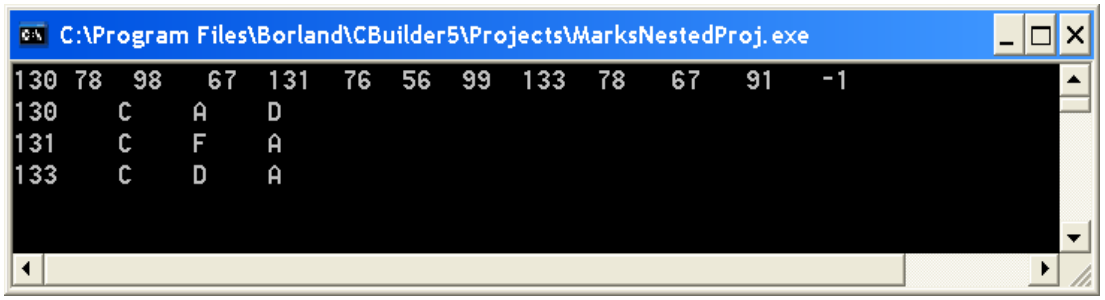
انتقل للقراءة من سطر جديد في الملف باستخدام جملة readln الفارغة.

إقرأ رقم طالب جديد id.

البرنامج

```
#include <iostream.h>
int main(int argc, char* argv[])
{ char Grade;
  int  mark, id,k;
  cin >> id;
  while (id > 0)
  {  cout << id;
     for (k=1; k<=3; k++)
     { cin >> mark;
       if (mark < 60) Grade='F';
       else if (mark<70) Grade = 'D';
       else if (mark<80)   Grade = 'C';
       else if (mark<90)   Grade = 'B';
       else Grade = 'A';
       cout << "    "<< Grade;
     }
     cout << endl;
     cin >> id;
   }
   getche();
    return 0;
}
```

عينة من تنفيذ البرنامج



تدريبات

1- شركة الكهرباء المحلية تحاسب زبائنها حسب حجم الاستهلاك ، كلما زاد الاستهلاك زادت التكلفة . قيمة الفاتورة النهائية تحسب حسب الجدول التالي :

من 1- 3000	0.05 للوات
من 3001- 6000	0.1 للوات
من 6000 - 10000	0.15 للوات
أكثر من 10000	0.2 للوات

اكتب برنامجاً يقرأ استهلاك زبون معين ثمّ يحسب قيمة الفاتورة له ويطبعها.

2- اطبع الناتج من تنفيذ هذا الجزء من البرنامج :

```
k= 5 ;
I:= - 4 ;
while  (I < = k)
{  I+= 2 ;
k- = 1 ;
cout <<  I + k <<endl;
}
```

اكتب برنامجاً يقرأ مجموعة من الأعداد الموجبة ، ويحسب عدد عناصر المجموعة، ثمّ يحسب المتوسط الحسابي و الانحراف المعياري. الانحراف المعياري يمكن حسابه حسب العلاقة:

std = Sqrt(sumsq - Sqr(sum)/N)/(N-1)

حيث sumsq هو مجموع مربعات الأعداد، sum هو مجموع الأعداد، N هو العدد الكلي للقراءات.

3- اكتب برنامجاً يقرأ جدولاً بالمسافات التي قطعت واستهلاك البنزين لكل مسافة وذلك لعدد من السيارات و احسب عدد الكيلومترات لكل جالون من البنزين لكل سيارة. أفرض أن المسافات معطاة بالكيلومتر و الاستهلاك بالليتر. أفرض أن عدد السيارات غير معروف وأن البيانات تنتهي بعدد سالب.

$$1+\frac{1}{x}+\frac{1}{x^3}+\frac{1}{x^5}+.....$$

4- اكتب برنامجاً يحسب مجموع المتسلسلة:

لأقرب 0.0001. يتوقف البرنامج إذا كان الطرف الجديد لا يضيف أكثر من 0.0001 للمجموع.

الخلاصة

عزيزي الدارس،

اشتملت هذه الوحدة على التعبيرات المنطقية و القواعد التى يجب اتباعها في تنفيذ العمليات الحسابية و الشرطية، كما تعرضنا الى الجمل المركبة و أساليب التفريغ و الاختبار و التكرار متخذين لغة C++ كنموذج للغات البرمجة.

وتضمنت الوحدة بعض التدريبات العملية التى يجد فيها الدارس كل القواعد التى درسها في هذه الوحدة ولذا يجب عليك عزيزي الدارس ادخالها في جهاز الحاسوب ومراجعتها.

لمحة مسبقة عن الوحدة التالية

عزيزي الدارس؛

إن الوحدة التالية التي بعنوان (الملفات المتتالية البسيطة) سنتعرف فيها على أنواع الملفات المستخدمة في لغة C++، بالاضافة إلى كيفية تصميم برنامج يقوم بقراءة البيانات من هذه الملفات باستخدام لغة C++.

مسرد المصطلحات

الجمل الشرطية Conditional Expressions

هي الجمل التي يستخدم فيها تعابير شرطية أو منطقية تكون قيمتها النهائية إما "صواب: أو "خطأ".

التعابير الألفبائية Char Expressions

هي التعابير التي تحوي حروفاً وأرقاماً وأقواساً.

التعابير المبنية على أنواع معددة Enumeratul types

هي التعابير التي تحوي أعداداً لقيمة السالبة تسمى الحارس أو الحاجز في مصطلح المبرمجين sentinal.

المصادر والمراجع

1- Deictl, H. & Deitel, P., **C++ How to Program.** Third Edition, Prentice Hall: 2001.
2- Savitch, W. **Problem Solving With C++: The Object of Programming**. Third Edition, Addison Wesley: 1999.
3- Friedman, F. and Koffman, E. **Problem Solving, Abstraction, and Design Using C++** (2 nd Edition). Addison Wesley: 1996.
4- Lippman, S. lajoie, J. **C++ Primer,** (3rd Edition). Addison Wesley: 1998.
5- Horstmann, C. **Computing Concepts With C++ Essentials.** John Wiley: 1997.
6- Stroustrup, B. **The C++ Programming Language.** (3 rd edition), Addison Wesley: 1990.
7- Ellis, M. and Bjarnc Stroupstrup, **The Annotated Reference Manual.** Addison Wesley: 1990.
8- Tcale, S. C++ IOS treams Handbook. Addison Wesley: 1993.
9- Meyers, S. **Effective C++: 50 Specific Ways to Improve Your Programs and Designs** (2 nd edition). Addison Wesley: 1996.
10- Meyers, S. **More Effective C++: 35 New Ways to Improve Your Programs and Designs.** Addison Wesley. 1996.
11- Schildt, H**. C++ From the Ground up.** Mc Graw Hill. 2003.
12- Savitch, W. **Absolute C++.** Addison Wesley: 2002.
13- Cogswell, J. **C++ all – in – one Desk Refcrence for Dummies.** 2002.


الوحدة الرابعة
الملفات المتتالية البسيطة

محتويات الوحدة

الموضوع	الصفحة
المقدمة	101
تمهيد	101
أهداف الوحدة	101
1. أنواع ملفات البيانات في لغة C++	**102**
2. أطوار فتح الملفات	**106**
الخلاصة	**109**
لمحة مسبقة عن الوحدة التالية	**109**
مسرد المصطلحات	**109**
المراجع	110

المقدمة

تمهيد

عزيزي الدارس؛

مرحباً بك في الوحدة الرابعة وهي بعنوان (الملفات المتتالية البسيطة).

الأمثلة التي تعرضنا لها في الوحدات السابقة كانت تأخذ البيانات من لوحة المفاتيح مباشرةً. هذا قد يكون مجدياً وممكناً إذا كان حجم البيانات المقروءة صغيراً و عددها قليل. الأمر يختلف كثيراً إذا كان حجم البيانات كبيراً. فمن الصعب جداً أن نطلب من موظف في مؤسسة إدخال البيانات المتعلقة بألف موظف مباشرةً من لوحة المفاتيح لكي نحسب الرواتب الشهرية! وماذا يحدث لو حدث خطأٌ في المحاولة الأولى؟ هل نطلب منه إعادة إدخال البيانات مرة أخرى؟

الحل في هذه الحالة، ومثلها من الحالات، هو أن يتم إدخال البيانات و تحضيرها مسبقاً في ملف منفصل، ثم تصميم البرنامج المعني بحيث يقرأ هذه البيانات من هذا الملف. نوع الملف المستخدم يعتمد على طبيعة التطبيق ونوع البيانات.

أهداف الوحدة

عزيزي الدارس؛

بعد فراغك من دراسة هذه الوحدة يتوقع أن تكون قادراً على أن:

- تفرق بين أنواع الملفات.
- تصمم ملف لإدخال البيانات.
- تحفظ البيانات في الملفات المتتالية بإستخدام لغة C++.
- تعدل فى البيانات داخل الملفات المتتالية بإستخدام لغة C++.

1. أنواع ملفات البيانات في لغة C++

لغة C++ لها أكثر من طريقة للتعامل مع الملفات وتتعامل مع نوعين رئيسين من الملفات:

1- الملفات المتتالية، والتي يتم تخزين المعلومات فيها بصورة متتالية. عملية الوصول للمعلومة في هذه الحالة تتطلب المرور بكل ماهو قبلها و تستغرق وقتاً طويلاً نسبياً.

2- الملفات العشوائية، والتي يمكن تخزين المعلومات فيها بصورة مفهرسة بناءً على مفتاح رقمي معين. الوصول إلى المعلومة في هذه الحالة يتم بصورة مباشرة دون الحاجة للمرور بما هو قبلها.

في هذا الفصل نتعرف على الملفات المتتالية في C++، و نؤجل النظر في الملفات العشوائية إلى وحدة أخرى.

الملفات المتتالية و نظام الإدخال بالتمرير Stream I/O

لعلك عزيزي الدارس تذكر أننا عرضنا في الوحدة الثانية نظام الإدخال و الإخراج بالتمرير، وذكرنا أن الأمرين cin و cout ماهما إلا **ممران افتراضيان** بين الأجهزة الطرفية و الحاسوب. عمليات الإدخال و الإخراج ماهي إلا تمرير للمعلومات المطلوبة عبر هذين الممرين الإفتراضيين.

التعامل مع الملفات المتتالية في C++ عبارة عن تعميم لنظام التمرير الذي عرضنا في الوحدة الثانية. الملفات في C++ يتم تعريفها **كممرات إفتراضية** للإدخال أو الإخراج. C++ تعطي نوعين من الممرات لتعريف الملفات:

1- ifstream، والذي يستخدم لتعريف ملفات الإدخال (القراءة).

2- ofstream، والذي يستخدم لتعريف ملفات الإخراج (الكتابة).

التعامل مع هذين النوعين يتطلب إدراج الملف الإفتتاحي fstream.h في المنطقة الإفتتاحية من البرنامج.

الأمثلة أدناه توضح كيف يتم تعريف الملفات و كيف يتم التعامل معها.

ملفات الإخراج Output Files(الكتابة) و كيفية التعامل معها:

المثال البسيط أدناه يوضح كيف يتم تعريف ملف OutFile وكيف يمكن تخزين معلومات عليه. المعلومات يتم أخذها من لوحة المفاتيح مباشرةً.

مثال (1):

اكتب برنامجاً يقرأ الأسماء و المعدلات التراكمية لعشرة طلاب ويخزنها في ملف اسمه StudentInfo.txt.

البرنامج:

```
#include <fstream.h>
#include <iostream.h>
int main()
{
 // Define OutFile using ofstream and open it
 ofstream OutFile( "StudentsInfo.txt", ios::out );

  if ( !OutFile ) {  // overloaded ! operator
    cerr << "File could not be opened" << endl;
    exit( 1 );    // prototype in cstdlib
  }
  cout << "Enter the student names, and GPA's.\n"
```

```
      << "Enter end-of-file to end input.\n? ";
   int ID;
   char name[ 30 ];
   float GPA;
   while ( cin >> ID >> name >> GPA ) {
<< ' ' << GPA << '\n';      OutFile << ID << ' ' << name
      cout << "? ";
   }
   return 0;  // ofstream destructor closes file
}
```

تحليل البرنامج

يبدأ البرنامج في السطر الثاني بتعريف الملف OutFile كممر للإخراج من نوع ofstream وفتحه :

```
ofstream OutFile( "StudentsInfo.txt", ios::out );
```

☜ لاحظ أن التعريف يبدأ في الجانب الأيسر بكلمة ofstream ، متبوعة بإسم المتغير OutFile، ثم، بين القوسين، الإسم الفعلي للملف StudentsInfo.txt و**طور فتح الملف** (الغرض الذي من أجله فتح الملف) ios::out. استخدام ios::out تدل على أن الملف تم فتحه للكتابة (out). إذا تم فتح ملف للكتابة **فإن محتوياته الحالية تمسح كلياً قبل بدأ الكتابة**. أي أننا لانستطيع الإضافة أو تعديل محتويات ملف في هذا الطور. إذا أردنا الإضافة إلى المحتويات الحالية دون مسحها فعلينا أن نستخدم الطور ios::app. أما إذا أردنا أن ندخل معلومات من ملف فعلينا أن نفتحه في الطور ios::in.

الكتابة في الملف تتم في السطر الخامس من أسفل البرنامج داخل جملة while:

```
while ( cin >> ID >> name >> GPA ) {
      << ' ' << GPA << '\n';          OutFile << ID << ' ' << name
```

☜ لاحظ استخدام جملة cin داخل جملة while. الأمر cin يرجع القيمة صفر إذا أدخل المستخدم علامة نهاية الملف (بالضغط على مفتاح Ctrl متبوعاً بالحرف z بصورة متزامنة).

كما هو واضح من المثال أعلاه، فإن الكتابة إلى الملف OutFile تشبه إلى حد كبير الكتابة إلى الممر cout.

☜**ملاحظة هامة:** لاحظ أن جملة تعريف الملف:

ofstream OutFile("StudentsInfo.txt", ios::out);

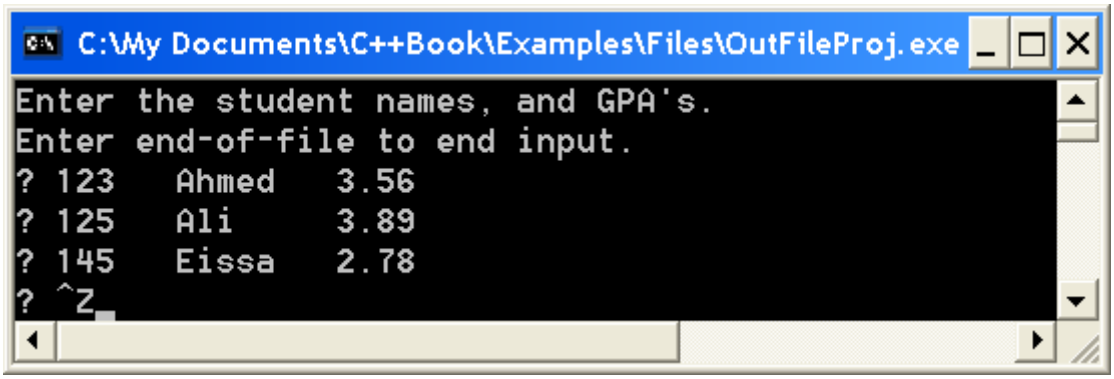
تقوم بفتح الملف تلقائياً و لانحتاج إلى جملة منفصلة لفتح الملف. غير أنه يمكن فصل عملية فتح الملف من التعريف بالصورة التالية:

ofstream OutFile;// تعريف الملف

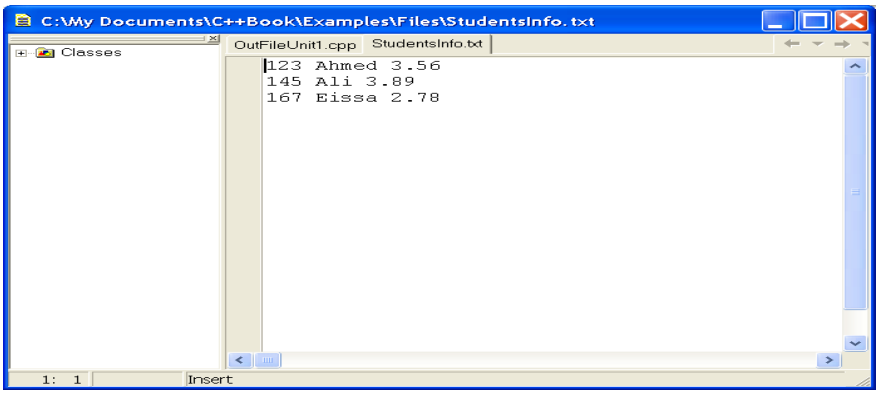
OutFile.open("StudentsInfo.txt", ios::out)// فتح الملف

في هذه المرحلة نفضل استخدام الطريقة الأولى. الطريقة الثانية تتطلب إدراك أن المتغير OutFile عبارة عن كائن object من فصيلة ofstream (وهي حقيقة ظللنا نخفيها حتى الآن). الكائنات و الفصائل نتعرض لها في جزء آخر من هذا المقرر.

عينة من تنفيذ البرنامج:



محتويات الملف بعد الكتابة إليه:



2. أطوار فتح الملفات

فتح ملف للإدخال ios::in

فتح ملف للكتابة عليه ios::out

فتح ملف للإضافة إليه ios::app

هناك أطوار أخرى ثانوية يمكن التعرض لها حين تدعو الحاجة.

☜ملاحظة هامة:

إذا تم فتح ملف للكتابة في الطور ios::out فإن المحتويات الحالية يتم مسحها بالكلية.

إذا أردنا أن نضيف معلومات إلى ملف، فينبغي فتحه في الطور ios::app.

أسئلة تقويم ذاتي



اكتب برنامجاً يضيف أسماء خمسة طلاب جدد و المعلومات المتعلقة بهم (الرقم والمعدل) إلى الملف StudentsInfo.txt الذي تم إنشاؤه في المثال السابق. لاحظ أن الملف يحتوي على معلومات سابقة و ينبغي الحذر من مسحها. وعليه ينبغي أن يفتح الملف في طور الإضافة ios::app.

ملفات الإدخال: القراءة من ملف

في الفقرات السابقة تعرضنا لكيفية إنشاء ملف وفتحه للكتابة. في المثال التالي نعرض لكيفية القراءة من ملف.

مثال (2)

أكتب برنامجاً يفتح الملف StudentsInfo.txt الذي أنشئ في المثال و يطبع محتوياته على الشاشة.

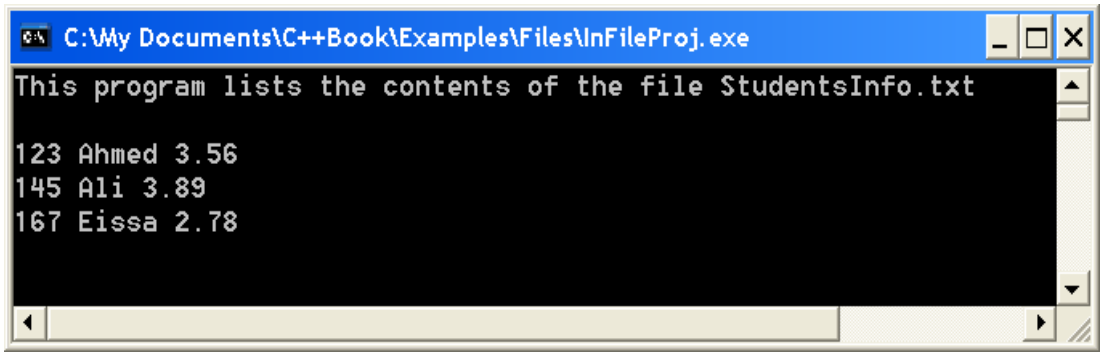
البرنامج:

في هذه الحالة نغير جملة تعريف الملف، بحيث نستخدم الفصيلة (النوع) ifstream بدلاً من ofstream و الطور ios::in بدلاً من الطور ios::out :

```
##include <fstream.h>
#include <iostream.h>
int main()
{
 // Define InFile using ifstream and open it
 ifstream InFile( "StudentsInfo.txt", ios::in );

  if ( !InFile ) {  // overloaded ! operator
    cerr << "File could not be opened" << endl;
    exit( 1 );    // prototype in cstdlib
  }
  cout << "This program lists the contents of "
       << "the file StudentsInfo.txt\n\n";
  int ID;
  char name[ 30 ];
  float GPA;
  while ( InFile >> ID >> name >> GPA ) {
    cout << ID << ' ' << name << ' ' << GPA << '\n';
   }
   cin >> ID;
  return 0;  // ofstream destructor closes file
}
```

عينة من تنفيذ البرنامج:



☜ملاحظات على البرنامج:

لاحظ التغييرات التي حدثت على جملة تعريف الملف:

ifstream InFile("StudentsInfo.txt", ios::in);

هذه المرة غيرنا إسم المتغير إلى InFile بدلاً من OutFile. أسماء الملفات لنا مطلق الحرية في اختيارها. مثلاً كان بالإمكان تسمية الملف StudentFile أو أي إسم آخر يوحي بنوع المحتوى.

القراءة من الملف تتم هذه المرة داخل جملة ال while:

```
while ( InFile >> ID >> name >> GPA ) {
cout << ID << ' ' << name << ' ' << GPA << '\n';
}
```

استخدام الممر الإفتراضي InFile يشبه إلى حد كبير استخدام الممر cin.

أسئلة تقويم ذاتي



اكتب برنامجاً ينقل محتويات الملف StudentsInfo.txt الذي استخدم في الأمثلة أعلاه إلى ملف آخر اسمه Std2Info.txt.

الخلاصة

عزيزي الدارس،

في هذه الوحدة تعرفنا على أنواع الملفات التى تم فيها حفظ البيانات بإستخدام لغة C++، كما تعرفنا على كيفية استخدام لغة C++ في فتح ملف و إدخال البيانات فيه وكيفية عرض البيانات من داخل الملف على الشاشة دون عمل تغيير عليها.

حاول عزيزي الدارس أن تدخل الأمثلة في الحاسوب و التأكد من صحتها.

لمحة مسبقة عن الوحدة التالية

عزيزي الدارس،

في الوحدة التالية سنتعرف المصفوفات وكيفية برمجتها بإستخدام لغة C++ وسنعرض لك عدداً من الأمثلة نسميها الخوارزمية لتسهل فهمها ثم البرامج المطبق لذلك.

مسرد المصطلحات

الملفات التالية

هي الملفات التي يتم تخزين المعلومات فيها بصورة متتالية.

الملفات العشوائية

هي الملفات التي يتم فيها تخزين المعلومات بصورة مفهرسة بمفتاح رقمي معين ولذا يمكن الوصول للمعلومة بصورة عشوائية.

المصادر والمراجع

1) Deictl, H. & Deitel, P., **C++ How to Program.** Third Edition, Prentice Hall: 2001.
2) Savitch, W. **Problem Solving With C++: The Object of Programming**. Third Edition, Addison Wesley: 1999.
3) Friedman, F. and Koffman, E. **Problem Solving, Abstraction, and Design Using C++** (2 nd Edition). Addison Wesley: 1996.
4) Lippman, S. lajoie, J. **C++ Primer,** (3rd Edition). Addison Wesley: 1998.
5) Horstmann, C. **Computing Concepts With C++ Essentials.** John Wiley: 1997.
6) Stroustrup, B. **The C++ Programming Language.** (3 rd edition), Addison Wesley: 1990.
7) Ellis, M. and Bjarnc Stroupstrup, **The Annotated Reference Manual.** Addison Wesley: 1990.
8) Tcale, S. C++ IOS treams Handbook. Addison Wesley: 1993.
9) Meyers, S. **Effective C++: 50 Specific Ways to Improve Your Programs and Designs** (2 nd edition). Addison Wesley: 1996.
10) Meyers, S. **More Effective C++: 35 New Ways to Improve Your Programs and Designs.** Addison Wesley. 1996.
11) Schildt, H**. C++ From the Ground up.** Mc Graw Hill. 2003.
12) Savitch, W. **Absolute C++.** Addison Wesley: 2002.
13) Cogswell, J. **C++ all – in – one Desk Refcrence for Dummies.** 2002.


الوحدة الخامسة
المصفوفات

محتويات الوحدة

الموضوع	الصفحة
المقدمة	**113**
تمهيد	113
اهداف الوحدة	113
1. معالجة البيانات بالمصفوفات	**114**
1.1 المصفوفات	114
1.2 تعريف المصفوات في البرنامج	**115**
2. المصفوفات ذات البعدين	**122**
الخلاصة	**126**
لمحة مسبقة عن الوحدة التالية	**126**
مسرد المصطلحات	**126**
المصادر والمراجع	127

المقدمة

تمهيد

عزيزي الدارس؛

مرحباً بك في الوحدة الخامسة بعنوان (المصفوفات).

ما زلنا حتى الآن نتعامل مع متغيرات بسيطة لا تحتاج إلى أكثر من مكان واحد في ذاكرة الحاسوب. حتى عندما تعاملنا مع قوائم من المعطيات تضم أكثر من عنصر استطعنا أن نتحاشى تخزين القوائم بكاملها و تعاملنا معها من خلال متغير بسيط واحد. فمثلاً عند حساب المجموع لقائمة من المعطيات في الوحدة السابقة كنا نقرأ القائمة عنصراً تلو الآخر. نعالج العنصر الحالي ثم نحل محله عنصراً آخر فنعالجه. بهذه الطريقة استطعنا أن نتحاشى تخزين القائمة بكاملها وتأتي هذه الوحدة لنتعلم من خلالها كيف نصمم مصفوفة ونخزنها ونجري عمليات حسابية عليها ثم نطبع الناتج بصورة جميلة ومنسقة.

أهداف الوحدة



عزيزي الدارس،

بعد فراغك من دراسة هذه الوحدة نتوقع منك أن تكون قادراً على أن:

- **تصمم** مصفوفة بها قواعد بيانات.
- **تستخدم** المصفوفات داخل البرنامج.
- **تحل** مسائل حسابية بعد كتابة الخوارزمية وتصميم برنامج لها باستخدام لغة C++.
- **تتقن** كتابة برنامج لحل المسائل البسيطة بإستخدام لغة C++.

1. معالجة البيانات بالمصفوفات

كان أسلوب التعامل مع القوائم ممكناً لأن المعالجة كانت تتم في مرحلة واحدة وتنطوي على عمليات بسيطة لا تتطلب تخزين القائمة بكاملها. ولكن عند ما تتطلب المعالجة عدة مراحل في البرنامج فإن الصورة تتغير بعض الشيء. خذ مثلاً المسألة التالية:

مثال (1):
اكتب برنامجاً يقرأ مجموعة من درجات الحرارة، و يحسب الدرجة **القصوى**، الدرجة **الدنيا** و **متوسط** درجات الحرارة، ثم يحسب عدد درجات الحرارة التي تزيد قيمتها على المتوسط و عدد درجات الحرارة التي تقل قيمتها عن المتوسط.

يتضح من هذه المسألة أن المعالجة تتم على مرحلتين:

- في المرحلة الأولى نحسب الدرجة القصوى، الدرجة الدنيا و المتوسط.
- في المرحلة الثانية نحسب عدد الدرجات التي تزيد قيمتها على المتوسط وعدد درجات الحرارة التي تقل قيمتها عن المتوسط.

المرحلة الثانية تتطلب بقاء قيم درجات الحرارة في الذاكرة حتى يتسنى لنا مقارنتها بالمتوسط أو إعادة قراءتها مرةً أخرى. إعادة قراءة البيانات يؤثر سلباً على أداء البرنامج، ذلك أن التعامل مع أجهزة الإدخال و الإخراج أبطأ مئات المرات من التعامل مع ذاكرة الحاسوب. و عليه فإن الخيار الأول هو الأمثل.

تخزين عدة قيم لمتغير واحد في ذاكرة الحاسوب يتطلب استخدام المصفوفات.

1.1 المصفوفات Matrices

المصفوفة عبارة عن متغير مركب يتكون من عدة عناصر كلها من جنسٍ واحد، تشترك في الاسم و تختلف في الموقع داخل المصفوفة. العناصر المكونة للمصفوفة قد تكون بسيطة أو مركبة أيضاً. لكي نتعامل مع عنصر معين في المصفوفة نستخدم اسم المصفوفة مقروناً بمتغير أو ثابت ترقيمي يدل على ترتيب العنصر أو موقعه داخل المصفوفة.

2.1 تعريف المصفوفات في البرنامج

المصفوفات كغيرها من المتغيرات ينبغي تعريفها في البرنامج قبل استخدامها. يتم تعريف المصفوفة بذكر نوعها ثم اسمها متبوعاً ببيان حجمها. مثلاً الجملة الآتية تعرِّف مصفوفة x نوعها float وحجمها 10:

```
float x[10];
```

حسب التعريف أعلاه، المصفوفة x تحتوي على عشرة عناصر ابتداءً بالعنصر x[0] وانتهاءً بالعنصر x[9]، و هذا يعني أن مداها يمتد من 0 إلى 9. مدى المصفوفة هي مجموعة المواقع التي تحتلها العناصر داخل المصفوفة و تبدأ دائماً بالصفر و تنتهي بـJ (حجم المصفوفة – 1) في C++ حجم المصفوفة لابد أن يكون ثابتاً مباشراً أو معرّفاً.

الشكل العام لجملة تعريف المصفوفات أحادية البعد هو:

قاعدة (1):

; [حجم المصفوفة] < اسم المصفوفة> <نوع المصفوفة>

أمثلة على تعريف المصفوفات:

```
int mark[30];
float y[100];
char name[40];
```

بالطبع يمكن تعريف أكثر من مصفوفة من نفس النوع في جملة واحدة كما هو الحال في الأمثلة أدناه:

```
float  t[20], r[30], p[70];
int a[8], b[7], k[5];
```

1) القيم الإبتدائية للمصفوفات (ابتداء المصفوفات)

الكثير من التطبيقات يتطلب إعطاء قيمة مبدئية للمصفوفات قبل استخدامها. قد تكون القيمة المبدئية صفراً أو أي قيمة أخرى، وقد تكون القيمة المبدئية واحدة لكل عناصر المصفوفة (مثلاً إسناد القيمة صفر لكل عنصر في المصفوفة) وقد تختلف من عنصر إلى آخر.

C++ تسمح بإسناد قيم ابتدائية للمصفوفات عند تعريفها. الأمثلة البسيطة التالية توضح كيف يتم ذلك:

2) استخدام المصفوفات داخل البرنامج

يمكن التعامل مع أي عنصر من عناصر المصفوفة إما بذكر رقم العنصر مباشرةً (في شكل ثابت) أو باستخدام متغير ترقيمي (index) ، محصور بين القوسين المربعين[ ] ، يدل على موقع العنصر داخل المصفوفة. مثلاً يمكن قراءة العنصر الأول من المصفوفة mark أعلاه باستخدام الجملة:

```
cin >> mark[1];
```

كذلك يمكن قراءة العنصر الثاني باستخدام الجملة التالية:

```
cin >> mark[2];
```

أما إذا أردنا قراءة المصفوفة كلها فمن الأمثل استخدام متغير فهرسي كما في المثال التالي:

```
for (k= 0; k<10; k++)
    cin >> mark[k];
```

متغير التحكم k في المثال أعلاه يستخدم كمتغير ترقيمي index لتحديد موقع العنصر المراد قراءته.

عودة إلى مثال (1)

نعود الآن إلى المثال (1) أعلاه و نبدأ بتحليل المسألة.

تحليل المسألة:

المطلوب: المطلوب في هذه المسألة حساب وطباعة درجة الحرارة القصوى MaxTemp ودرجة الحرارة الدنيا MinTemp و متوسط درجة الحرارة MeanTemp. ثم بعد ذلك حساب وطباعة عدد درجات الحرارة فوق المتوسط AboveAv وعدد درجات الحرارة دون المتوسط BelowAv.

المعطيات: قائمة بدرجات الحرارة TempReadings مسبوقة بالعدد الكلي للقراءات N (عدد صحيح).

المتغيرات الوسيطة: نحتاج لمتغير للتحكم k، مثلاً ومتغير لحساب مجموع درجات الحرارة TempSum.

تصميم الخوارزمية

- الجزء الأول من المسألة تم معالجته في الفصل الماضي. هذا الجزء ينتهي بحساب الدرجة القصوى، الدرجة الدنيا و المتوسط.
- في الجزء الثاني نحسب عدد درجات الحرارة فوق المتوسط و عدد درجات الحرارة دون المتوسط:

صفِّر العدادات BelowAv=0, AboveAv=0

لكل درجات الحرارة من 1 إلى N

إذا كانت TempReading>MeanTemp إذاً

AboveAv=AboveAv+1

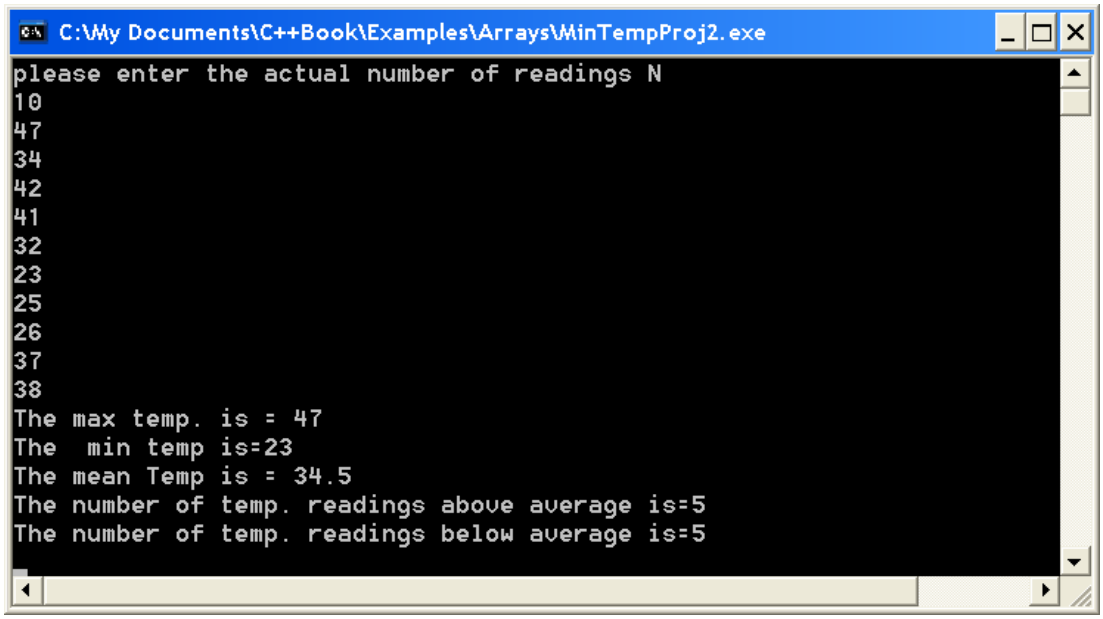
وإلا إذا كانت TempReading<MeanTemp إذاً

BelowAve=BelowAv+1

البرنامج

```
#include <iostream.h>
int main(int argc, char* argv[])
{
int k, N, AboveAv, BelowAv;
float MeanTemp, MaxTemp, MinTemp,TempSum;
float TempReading [100];  //no more than 100 readings are expected
//phase1: compute mean, max and min
cout <<"please enter the actual number of readings N"<<endl;
cin >> N;
TempSum=0; // clear the summation variable
MaxTemp=-1000; //assume a value below the range of temps
MinTemp= 1000; //assume a value above the range of temps
for (k=1; k<=N; k++)
{
cin >>TempReading[k];
TempSum=TempSum+TempReading[k];
if (TempReading[k] > MaxTemp)
MaxTemp=TempReading[k];
if (TempReading[k] < MinTemp)
MinTemp=TempReading[k];
}
MeanTemp = TempSum/N;
//phase 2: now compute AboveAv and BelowAv
AboveAv=0;   BelowAv=0;
for (k=1; k<=N; k++)
if (TempReading[k] > MeanTemp)
AboveAv = AboveAv +1;
else if (TempReading[k] < MeanTemp)
BelowAv = BelowAv + 1;
// Now print the results
cout <<"The max temp. is = "<< MaxTemp <<endl;
cout <<"The  min temp is=" << MinTemp <<endl;
cout <<"The mean Temp is = "<< MeanTemp <<endl;
cout <<"The number of temp. readings above average is=" <<
AboveAv<<endl;
cout <<"The number of temp. readings below average is=" <<
BelowAv<<endl;
return 0;
}
```

عينة من تنفيذ البرنامج



مثال (2)

اكتب برنامجاً يقرأ مجموعة من الأعداد الحقيقية ثم يطبعها بعكس الترتيب الذي أدخلت به. أفرض أن حجم المجموعة N.

تحليل المسألة

يتضح من طبيعة هذه المسألة أن حلها يتم في مرحلتين: قراءة البيانات ثم طباعتها بصورة عكسية، و عليه فإن المقام يتطلب استخدام مصفوفة لتمثيل البيانات.

المعطيات: حجم المجموعة N متبوعاً بعناصر المجموعة x1,x2, ...xN.

المطلوب: طباعة عناصر المصفوفة x بعكس الترتيب الذي قرئت به.

المتغيرات الوسيطة: نحتاج لمتغير تحكم k،

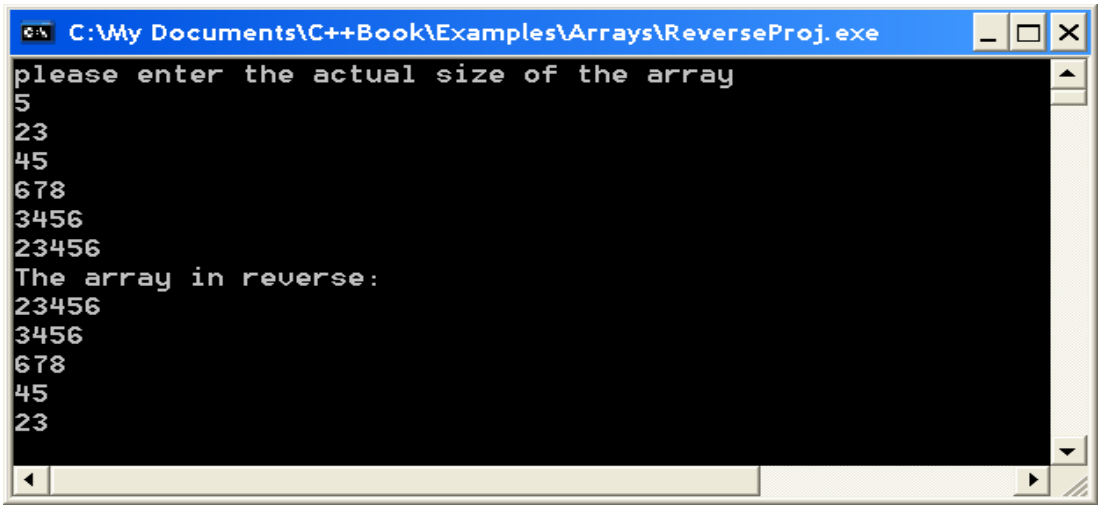
تصميم الخوارزمية:

الخوارزمية في هذه الحالة ليست معقدة و تتكون من تكرارين متتالين: الأول للقراءة و الثاني للطباعة.

البرنامج:

```
#include <iostream.h>
int main(int argc, char* argv[])
{  const int size=10;
   float x[size];
   int k,n;
   cout << "please enter the actual size of the array" <<endl;
   cin >> n;
   for (k=1; k<=n; k++)
     cin >> x[k];
   cout << "The array in reverse: "<<endl;
   for (k=n; k>=1; k--)
     cout << x[k]<<endl;
   return 0;
 }
```

عينة من تنفيذ البرنامج:



مثال (3)

اكتب برنامجاً يقرأ قطعة نص مخزنة في ملف ثم يحسب تكرار كل حرف من الحروف الأبجدية الصغيرة من a إلى z في هذه القطعة (يشمل التكرار الحروف الصغيرة و الكبيرة).

تحليل المسألة

المعطيات: قطعة نص مخزنة في ملف SourceFile.

المطلوب: حساب و طباعة تكرار كل حرف من الحروف الأبجدية الإنجليزية freqa, freqb, freqc…etc. هنا نحتاج إلى مصفوفة لتمثيل التكرار لكل الحروف الستة و عشرون في اللغة الإنجليزية.

الخوارزمية:

1- صفِّر العدادات (المصفوفة freq).

2- كرر حتى نهاية الملف:

i. اقرأ رمزاً من النص.

ii. إذا كان الرمز واقعاً في المدى بين a و z أضف واحد إلى العداد المقابل.

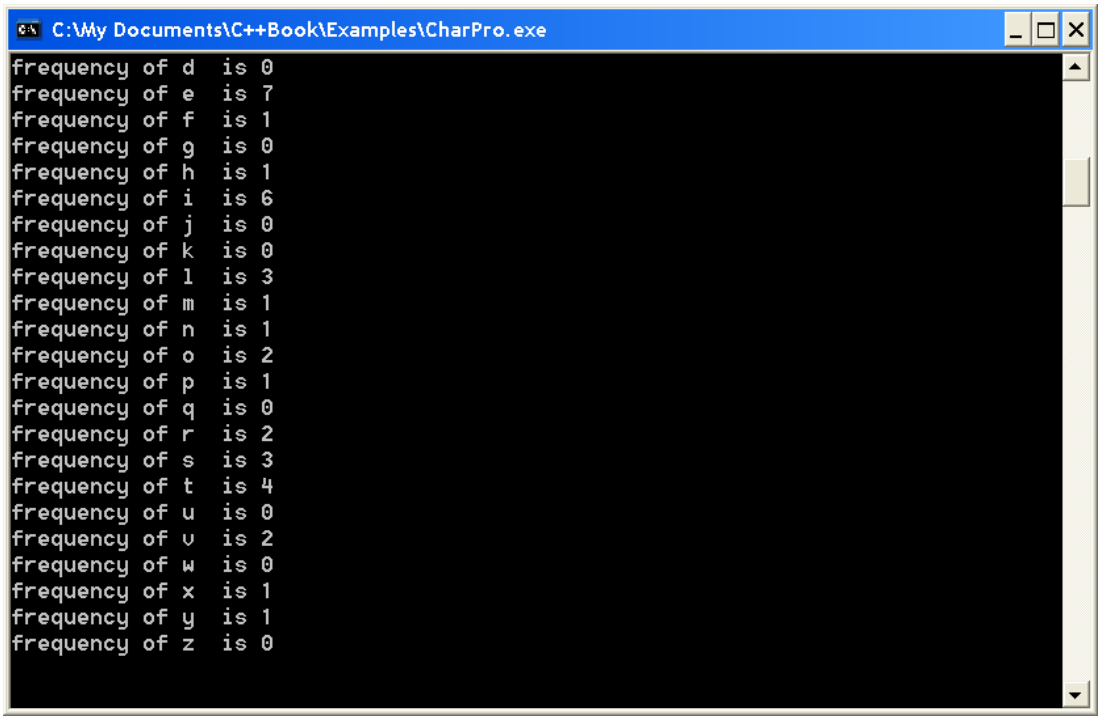
iii. اطبع النتائج لكل الحروف.

البرنامج:

```
#include <fstream.h>
#include <iomanip.h>
#include <conio>
#include <iostream.h>
#pragma argsused
int main(int argc, char* argv[])
{
//Reading characters
 //Create a file object InFile ifstream InFile("Text.txt", ios::in);
char ch;
int CharFrequency[256]={0};  // The frequency array
// Check if the file can be opened
if (!InFile)
{ cerr<<"File Could not be open"<<endl;
exit(1);
}
// Read the characters and count the frequency
while (InFile >>ch) //reading characters from the file
{ CharFrequency[ch]++; //Increment char frequency
cout <<ch<<"  ";  //optionally print out the read character
```

```
}
cout <<endl;
for (ch='A'; ch<='z'; ch++)
  cout <<"frequency of "<<ch<<
        "  is "<<CharFrequency[ch]<<endl;
cout << endl;
getch();
return 0;
}
```

عينة من تنفيذ البرنامج:



2. المصفوفات ذات البعدين

تستخدم المصفوفات ذات البعدين في تمثيل الجداول و ما يقوم مقامها من تراكيب البيانات وتعرّف في C++ بذكر المدى لكلٍ من البعدين بين قوسين مربعين. مثلاً:

```
float table[5][3];
```

في المثال أعلاه المدى الأول 5 يمثل عدد الصفوف في المصفوفة table بينما يمثل المدى الثاني 3 عدد الأعمدة.

مثال (4)

اكتب برنامجاً يقرأ الدرجات التي حصل عليها طلاب فصل معين في عدة مواد ثم يحسب معدل الطالب في هذه المواد حسب المعادلة:

المعدل = (مجموع الدرجات – عدد المواد × 40)/100 إذا كانت الدرجة أكبر من 60

أو المعدل=1 إذا كانت الدرجة أقل من 60.

البيانات المتعلقة بكل طالب تعطى في سطر منفصل و تبدأ برقم الطالب متبوعاً بالدرجات التي حصل عليها. البرنامج ينبغي أن يطبع النتائج في شكل جدول يشمل الدرجات و المعدل. افرض أن عدد الطلاب لا يزيد على 30 و أن عدد المواد لا يزيد على 5.

تحليل المسألة:

المعطيات: العدد الفعلي للطلاب m و العدد الفعلي للمواد n ثم جدول Results يشمل النتائج التي حصل عليها الطلاب مع أرقامهم.

المطلوب: حساب المعدل لكل طالب ثم طباعة جدول يعطى درجات الطلاب مع معدلاتهم.

المتغيرات الوسيطة: مصفوفة Score لتخزين معدلات الطلاب ريثما تطبع ومتغيرات تحكم i,j للتحكم في التكرارات المتداخلة التي ستنشأ.

تصميم الخوارزمية

1- اقرأ عدد الطلاب و عدد المواد.

2- كرر لكل طالب:

i. إقرأ الرقم و الدرجات.

ii. احسب مجموع الدرجات.

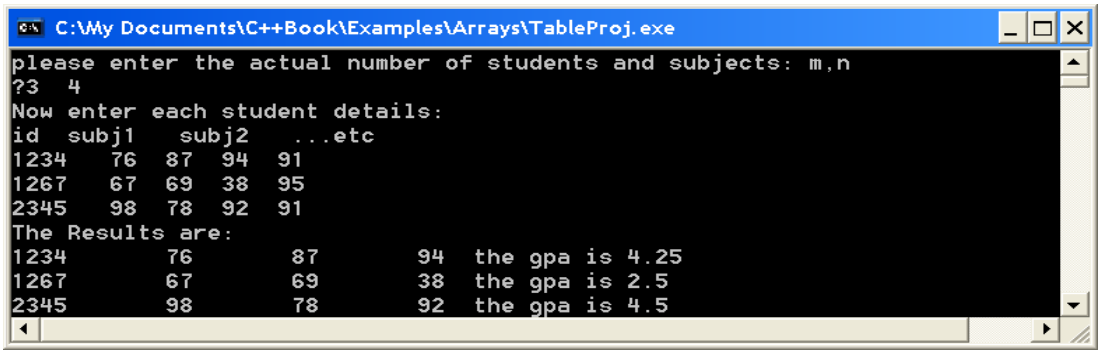
iii. احسب المعدل.

3- اطبع جدول النتائج لكل الطلاب.

البرنامج:

```
#include <iostream.h>
int main(int argc, char* argv[])
{   const int MaxStudents=30, MaxSubjects=5;
    int i,j,m,n;
    float sum;
    int id[MaxStudents];
    float Score[MaxStudents];
    float Result[MaxStudents][MaxSubjects];
    cout <<"please enter the actual number of students and subjects: m,n\n?";
    cin >> m >> n;
cout <<"Now enter each student details:\n"
          <<"id  subj1   subj2   ...etc\n";
for (i=1;i<=m;i++)
    {  cin >> id[i]; // read student id
       Score[i]=0;
       for (j=1;j<=n;j++)
       {
         cin >> Result[i][j];
         if (Result[i][j]>60)
           Score[i]=Score[i]
            + floor((Result[i][j]-40)/10);
         else
           Score[i]=Score[i] +1;
       }
       Score[i]=Score[i]/n;
    }
     //now print the table of results
      cout << “The Results are : \n”;
       for (i=1;i<=m;i++)
       {
         cout << id[i];
         for (j=1;j<=m;j++)
            cout << "       "<< Result[i][j];
         cout <<"  the gpa is "<<Score[i];
         cout <<endl;
       }
    return 0;
}
```

عينة من تنفيذ البرنامج:



أسئلة تقويم ذاتي



1- اكتب برنامجاً يقرأ مصفوفة من الأعداد الحقيقية حجمها N ثم يطبع العناصر المخزنة في المواقع الفردية منها.

2- صمم برنامجاً يقرأ مصفوفة أعداد صحيحة ثم يبدل العنصر الأخير منها مع العنصر الأول. أفرض أن حجم المصفوفة N.

تدريب (1)



1- اكتب برنامجاً يقرأ مصفوفة أعداد حقيقية منها ما هو سالب و منها ما هو موجب ثم يحذف الأعداد السالبة ويطبع رسالة على الشاشة تعطي عدد القيم السالبة التي حذفت وحجم المصفوفة الجديدة.

2- عد برنامجاً يقرأ مصفوفة ذات بعدين عدد صفوفها M وعدد أعمدتها N، ثم يحسب القيمة القصوى في كل صف، ثم يعدِّل عناصر كل صف بقسمتها على القيمة القصوى فيها.

3- نفذ برنامجاً يبحث عن كلمة السر لمستخدم معين بين مجموعة حجمها N من كلمات السر المخزنة في ملف معين ثم يسمح للمستخدم بالاستمرار في تنفيذ البرنامج إذا وجدت كلمة السر أو يمنعه من الاستمرار إذا لم توجد كلمة السر. كلمات السر هنا يمكن تمثيلها كمصفوفة من الكلمات.

4- اكتب برنامجاً يحسب عدد مرات تكرار الأرقام من 0-9 في نص معين.

الخلاصة

عزيزي الدارس؛

لقد تناولنا المصفوفات وكيفية تصميمها داخل البرنامج باستخدام لغة C++، في هذه الوحدة الخامسة من مقرر أساليب البرمجة كما تعلمنا توظيف امكانات الحاسوب في اجراء العمليات الحسابية داخل المصفوفات وطباعة الناتج عن ذلك.

تأكد عزيزي الدارس من تطبيق الأمثلة التي وردت في هذه الوحدة على جهاز الحاسوب لأنها تعتبر تقويماً ذائياً لدى استيعابك لمضمون ومحتويات الوحدة.

لمحة مسبقة عن الوحدة التالية

سنتعرض عزيزي الدارس في الوحدة التالية لأسلوب البرمجة التركيبية والذي يستخدم فيه الدوال بصورة مباشرة كما سنتعرَّف أنواع عناصر الدوال في لغة C++ وكيفية تصميمها وإعداد البرامج لها.

مسرد المصطلحات

المصفوفات Matrices

عبارة عن متغير مركب يتكون من عدة عناصر كلها من جنس واحد تشترك في الاسم تختلف في الموقع داخل المصفوفة.

المصادر والمراجع

1- Deictl, H. & Deitel, P., **C++ How to Program.** Third Edition, Prentice Hall: 2001.
2- Savitch, W. **Problem Solving With C++: The Object of Programming**. Third Edition, Addison Wesley: 1999.
3- Friedman, F. and Koffman, E. **Problem Solving, Abstraction, and Design Using C++** (2 nd Edition). Addison Wesley: 1996.
4- Lippman, S. lajoie, J. **C++ Primer,** (3rd Edition). Addison Wesley: 1998.
5- Horstmann, C. **Computing Concepts With C++ Essentials.** John Wiley: 1997.
6- Stroustrup, B. **The C++ Programming Language.** (3 rd edition), Addison Wesley: 1990.
7- Ellis, M. and Bjarnc Stroupstrup, **The Annotated Reference Manual.** Addison Wesley: 1990.
8- Tcale, S. C++ IOS treams Handbook. Addison Wesley: 1993.
9- Meyers, S. **Effective C++: 50 Specific Ways to Improve Your Programs and Designs** (2 nd edition). Addison Wesley: 1996.
10- Meyers, S. **More Effective C++: 35 New Ways to Improve Your Programs and Designs.** Addison Wesley. 1996.
11- Schildt, H**. C++ From the Ground up.** Mc Graw Hill. 2003.
12- Savitch, W. **Absolute C++.** Addison Wesley: 2002.
13- Cogswell, J. **C++ all – in – one Desk Refcrence for Dummies.** 2002.


الوحدة السادسة
البرمجة التركيبية: الدوال

محتويات الوحدة

الموضوع	الصفحة
مقدمة	**131**
تمهيد	131
أهداف الوحدة	132
1. لغة C++ تدعم نوعين من البرامج الجزئية	**133**
2. مفاهيم أساسية	**137**
3. تعريف الدالة واستدعائها	**137**
4. دوال من غير عناصر	**140**
5. أنواع عناصر الدوال في C++	**141**
1.5 عناصر الادخال	142
2.5 عناصر التلقي	144
6. تمرير المصفوفات كعناصر للاجراءات و الدوال	**150**
الخلاصة	**154**
لمحة مسبقة عن الوحدة التالية	**154**
مسرد المصطلحات	**155**
المصادر والمراجع	**156**

المقدمة

تمهيد

عزيزي الدارس؛

مرحباً بك الى الوحدة السادسة التي تُعنَى بالتصميم التركيب. التصميم التركيبي أسلوبٌ أثبت جدواه في شتى مناحي الحياة ومنها مجال هندسة البرمجيات. فمنذ وقتٍ ليس بالقريب ظل مصممو النظم البرمجية يتبعون هذا الأسلوب بنجاح في بناء أنظمة عالية الفاعلية و الإعتمادية. أسلوب التصميم التركيبي يهدف إلى الوصول إلى حل المسائل المعقدة بتركيب **الحلول الجزئية** لفروع المسألة الأصلية و ينبني على ثلاث **دعامات أساسية**:

- أسلوب "فرِّق تَسُد": قسِّم المسألة المعقدة إلى **مسائل جزئية صغيرة** يسهل حلها ثم جمِّع هذه الحلول لتعطيك حلاً متكاملاً.
- الأسلوب الإنحدارى (من فوق إلى اسفل) في تقسيم المسائل: تقسيم المسألة إلى مسائل فرعية ينبغي أن يتم بصورةٍ تدريجية من فوق إلى اسفل بصورة تتناسب مع طبيعة المسألة.
- تجنب إعادة "اختراع العجلة": حاول قدر الإمكان ألا تبدأ من الصفر و استفد من الحلول الجزئية الجاهزة.

عملية إعادة تركيب الحل النهائي من حلول جزئية يعتمد بصورة قوية على تقنية **البرامج الجزئية** في لغات البرمجة. البرامج الجزئية عبارة عن بريمجات شبه مستقلة تتعامل مع البيئة المحيطة من خلال واجهة (قد تتطلب تمرير بعض العناصر)، ويمكن استخدامها (أقصد البرامج الجزئية) بعد تعريفها في البرامج من غير أن يعرف المبرمج كيف تعمل. المهم هنا "ماذا" تعمل.

اهداف الوحدة



عزيزي الدارس،

بعد فراغك من دراسة هذه الوحدة ينبغي أن تكون قادراً على أن:

- **تميّز** إسلوب التصميم التركيبي و استخدامه في حل المسائل المعقدة.
- **تعرف** الدالة وعناصرها و أنواعها.
- **توظف** الدوال داخل البرنامج بإستخدام لغة C++.
- **تمرر** المصفوفات كعناصر للاجراءات و الدوال.

1. لغة C++ تدعم نوعين من البرامج الجزئية

1- الدوال التي ترجع قيمةً واحدة من نوع معين عند استدعائها typed functions.

2- الدوال التي لا ترجع أي قيم مباشرةً عند استدعائها void functions (هذا النوع قد يرجع قيم بصورة غير مباشرة من خلال عناصر التمرير).

وحتى نكسر الحاجز النفسي ونسهل فهم الدوال، نعطي مثالاً بسيطاً على كيفية تعريف و استخدام كلٍ من النوعين:

مثال (1): الدوال التي ترجع قيمة واحدة typed functions:

اكتب برنامجاً يحسب قيمة y باستخدام العلاقة:

$$y = 3x^3 + 4x$$

لقيم x من 1 إل 10. استخدم دالة تحسب مكعب العدد الذي يمرر لها.

الحل:

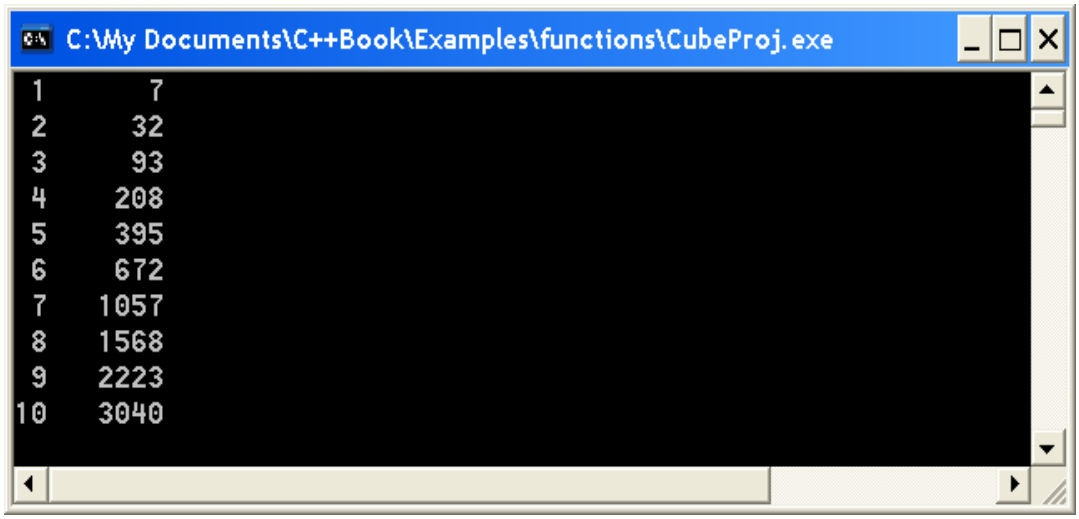
حساب قيمة y يتطلب حساب x3 (مكعب x)، وعليه نعرِّف دالة بسيطة تحسب هذه القيمة ونستخدمها في البرنامج. الدالة أطلقنا عليها إسم cube وتتطلب تمرير عنصر واحد من خلال واجهتها كما هو واضح من البرنامج أدناه:

☜ ملحوظة:

واجهة الدالة هي عنوانها، و الذي يتكون عادةً من نوعها، إسمها و العناصر التي تمرر لها. المثال أدناه يوضح ذلك.

```
#include <iomanip.h>
#include <iostream.h>
float cube(float);
int main(int argc, char* argv[])
{ float y,x;
  for (x=1.0; x<=10; x++)
   { y = 3*cube(x) + 4*x;
     cout <<setw(2)<< x <<setw(7)<< y<<endl;
   }
   cin >> x;
  return 0;
}
float cube(float r)
{ return r*r*r;  }
```

عينة من تنفيذ البرنامج:



تحليل البرنامج:

- لعل الدارس يلاحظ ما أحدثنا من تغييرات تركيبية في هذا البرنامج. في السطر الثالث من صدر البرنامج قمنا بتعريف **واجهة الدالة** cube:

float cube(float);

- بما أن هذه الدالة ترجع في نهايتها قيمة واحدة هي قيمة مكعب العدد الذي يمرر لها، لزم إعطاء نوع القيمة التي ينبغي إرجاعها قبل إسم الدالة على اليسار، float في هذه الحالة.
- الدالة cube تحسب مكعب العنصر الذي يمرر لها بين القوسين (بعد إسم الدالة)، وعليه لابد من إعطاء نوع العنصر الذي يمكن تمريره. لاحظ أننا لم نعط إسم العنصر، وإنما أعطينا نوعه فقط. وهذا مسموح به في C++، إذ الهدف في هذه المرحلة هو توضيح النوع فقط، إلا أنه لابأس بإعطاء إسم العنصر أيضاً، ولن يحتج المترجم على ذلك.
- تعريف واجهة الدالة لابد أن يسبق تعريف الدالة نفسها في C++، والهدف منه التعريف بإسم الدالة، نوع القيمة التي ترجعها، عدد و أنواع العناصر التي ينبغي تمريرها للدالة.

- تعريف واجهة الدالة يكون دائماً خارج البرنامج الرئيس ()main وليس داخله (قبل البرنامج)، وهذه مسألة لها علاقة بمستوى الرؤيا المطلوب والذي سنناقشه في قسمٍ لاحق من هذه الوحدة.
- جاء تعريف الدالة نفسها في أسفل البرنامج (بعد البرنامج ()main وخارجه):

```
float cube(float r)
{ return r*r*r;  }
```

- كما هو ملاحظ، تعريف الدالة يتكون من واجهة الدالة، مضافاً إليه صلب الدالة و الذي يتكون في هذه الحالة من سطر واحد:

```
{ return r*r*r;  }
```

- الدوال التي ترجع قيمة واحدة لابد أن تنتهي بجملة return والتي تتكون من الكلمة الخاصة return متبوعة بالقيمة المرجعة. القيمة المرجعة قد تكون إسم متغير عادي أو تعبير يرجع قيمة واحدة. نوع القيمة المرجعة ينبغي أن يتوافق مع نوع الدالة الذي تم إعطاؤه في تعريف الواجهة.
- كان يمكن كتابة الدالة أعلاه بالصورة التالية:

```
float cube(float r)
{ float s;
 s= r*r*r;
 return s;
 }
```

في هذه الحالة قمنا بتعريف متغير محلي s واستخدمناه كمتغير وسيط، ثم أرجعنا قيمته.

- ليس بالضرورة أن يأتي التعريف الكامل للدالة دائماً بعد البرنامج، ويمكن أن يأتي قبل البرنامج. في هذه الحالة لاداعي لتعريف الواجهة بصورة منفصلة. مثلاً، البرنامج أعلاه كان يمكن كتابته بالصورة التالية:

```
#include <iomanip.h>
#include <iostream.h>
float cube(float r)
{ return r*r*r;  }
int main(int argc, char* argv[])
{ float y,x;
  for (x=1.0; x<=10; x++)
   { y = 3*cube(x) + 4*x;
     cout <<setw(2)<< x <<setw(7)<< y<<endl;
   }
   cin >> x;
  return 0;
}
```

- لغة C++ لاتسمح بتداخل الدوال، أي أنه لايمكن تعريف دالة داخل دالة أخرى. من هذا المنطلق لايمكن تعريف الدالة cube داخل البرنامج الرئيس، لأن البرنامج الرئيس عبارة عن دالة في مفهوم C++.
- الدالة cube تم استدعاؤها في السطر السابع من البرنامج (الصيغة الأولى) كجزء من جملة الإسناد:

y = 3*cube(x) + 4*x;

وهذا هو المعتاد في حالة الدوال التي ترجع قيمة معينة: لابد من استدعائها كجزء من أمر آخر و إلا فقدنا القيمة المرجعة. يمكن استخدام الدالة كجزء من أمر الطباعة cout، مثلاً يمكن طباعة مكعب x باستخدام الأمر cout <<cube(x)<<endl; .

2. مفاهيم أساسية

واجهة الدالة (التعريف الأولي للدالة): واجهة الدالة هي عنوان الدالة و الذي يتركب من الآتي:

>نوع الدالة> <إسم الدالة>]) < قائمة عناصر الدالة;([

<type> <function name> ([Parameter List]);

مثال ذلك تعريف واجهة الدالة cube الذي استخدمناه في المثال أعلاه:

float cube(float);

نوع الدالة: المقصود به نوع القيمة التي ترجعها الدالة، ويعطى دائماً على يسار إسم الدالة. في المثال أعلاه نوع الدالة هو float.

عناصر الدالة: المقصود بعناصر الدالة المتغيرات التي ينبغي تمريرها كمدخلات للدالة كي تؤدي مهمتها. في المثال أعلاه هناك عنصر مدخل واحد، وهو المتغير الذي نود حساب مكعبه. لاحظ أنه ينبغي إعطاء نوع المتغير و إسمه في التعريف الكامل للدالة. عناصر الدالة تعطى بين قوسين على يمين إسم الدالة، ونفصل بينها بالفاصلة العادية. مثال ذلك العنصر r في تعريف الدالة cube: float cube(float r)

والعنصران c و Length في تعريف الدالة PrintLine: void PrintLine(char c, int Length)

الدالة PrintLine سنستخدمها في المثال (2) أدناه.

3. تعريف الدالة واستدعائها

المقصود بتعريف الدالة هو التعريف الكامل لها والذي يتكون من عنوانها ونصها.

والمقصود باستدعاء الدالة استخدام الدالة في البرنامج كأمر منفصل أو كجزء من أمر. فمثلا الدالة cube تم استدعاؤها كجزء من جملة الإسناد في البرنامج أعلاه:

y = 3*cube(x) + 4*x;

مثال (2): الدوال التي لا ترجع قيمة مباشرةً void functions:

اكتب دالة تطبع سطراً من الرموز بناءً على المعلومات التي يعطيها مستخدم الدالة. المعلومات التي ينبغي تمريرها للدالة هي نوع الرمز (نجمة، شرطة ...الخ)، و طول السطر (عدد الرموز). مثلاً إذا كان نوع الرمز هو النجمة وعدد الرموز 20، فإن الدالة تطبع سطراً مكوناً من عشرين نجمة على النحو التالي:

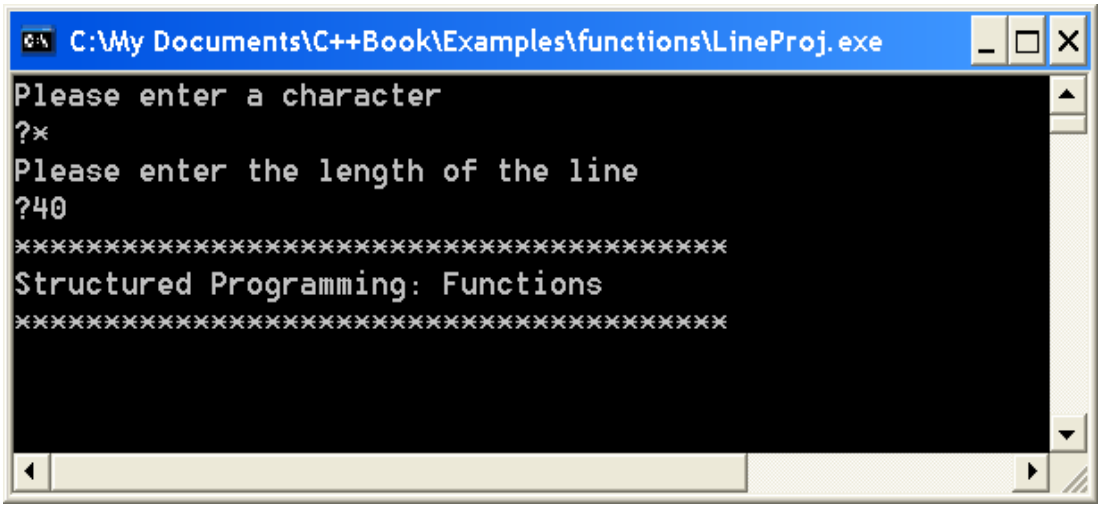
****

اكتب برنامجاً يستخدم هذه الدالة لطباعة عنوان هذا الفصل بين سطرين:

البرنامج:

```
#include <iostream.h>
void PrintLine(char, int);
int main(int argc, char* argv[])
{char ch; // character to be used in printing the line
 int Length;  // Length of line
 cout <<"Please enter a character\n?";
 cin >> ch;
 cout <<"Please enter the length of the line\n?";
 cin >>Length;
 PrintLine(ch, Length); //Print first line
 cout <<"Structured Programming: Functions"<<endl;
 PrintLine(ch, Length);
 cin >>ch;
 return 0;
}
void PrintLine (char c, int L)
{ for (int k=0; k<L; k++)
    cout <<c;
  cout <<endl;
}
```

عينة من تنفيذ البرنامج:



ملاحظات على البرنامج:

- كما فعلنا في المثال السابق، قمنا بتعريف واجهة الدالة PrintLine في السطر الثاني من البرنامج:

void PrintLine(char, int);

- بما أن الدالة PrintLine لا يتوقع منها أن ترجِّع قيمةً بعينها، فقد استخدمنا كلمة void في تعريفها، وهذه قاعدة عامة.
- الدالة PrintLine تتطلب تمرير عنصرين: الأول من نوع char و الثاني من نوع int، وهذا واضحٌ من تعريف الواجهة.
- التعريف الكامل للدالة جاء بعد نهاية البرنامج:

```
void PrintLine (char c, int L)
{ for (int k=0; k<L; k++)
  cout <<c;
  cout <<endl;
}
```

- ليست ثمة جديد في منطق الدالة، غير أنها لاتنطوي على جملة return إذ ليس هناك ما ترجعه.

4. دوال من غير عناصر

تمرير العناصر للدوال ليس أمراً إجبارياً، وإنما يستخدم إذا دعت الحاجة و حسبما يقتضيه المقام. مثلاً الدالة PrintLine أعلاه يمكن استخدامها من غير عناصر إذا ثبتنا نوع الرمز و طول السطر. في هذه الحالة يمكن كتابة الدالة كما يلي:

```
void PrintLine()
{ for (int k=0; k<50; k++)
      cout <<"*";
   cout <<endl;
}
```

لاحظ أننا في هذه الحالة استخدمنا قيمة ثابتة للرمز هي النجمة * ، كما استخدمنا قيمة ثابتة لطول السطر هي 50. وعليه أصبح ليس من الضرورة تمرير عناصر للدالة. إلا أنه لايخفى على الدارس أن الصورة الأولى من الدالة أكثر مرونة. المهم أن تعلم أن تمرير العناصر للدوال ليس إجبارياً وقد تواجه الكثير من الدوال التي لاتتطلب تمرير عناصر.

أسئلة تقويم ذاتي



1- ماالذي نعنيه ب "واجهة الدالة"، ومم تتكون هذه الواجهة؟
2- ماهي أنواع الدوال التي تدعمها C++؟
3- ما الذي نعنيه ب "تمرير العناصر للدالة" و كيف يتم ذلك؟
4- أين يتم تعريف واجهة الدالة؟
5- هل يمكن تعريف واجهة الدالة داخل البرنامج الرئيس main()؟
6- أين يتم التعريف الكامل للدالة؟
7- هل تسمح C++ بتداخل الدوال (تعريف دوال داخل دوال أخرى)؟
8- اكتب دالة تحسب مربع أي عدد صحيح يمرر لها. اكتب برنامجاً يستخدم هذه الدالة لحساب مربع الأعداد من 1 إلى 10 ويطبع النتيجة في شكل جدول

قاعدة (1):

الشكل العام لتعريف الدوال:

<type> <name> (<parameter list>)
<function body>

>نوع الدالة> <إسم الدالة>) <قائمة العناصر(<

اسم الدالة يختاره المبرمج حسب القواعد العامة لتسمية المتغيرات في C++. قائمة العناصر تشمل مجموعة من المتغيرات المعرفة و تنقسم إلى نوعين كما سنفصل في الفقرات أدناه. صلب الدالة يعتبر صورةً مصغرة من البرنامج الكبير و قد يحتوى على تعريفات للمتغيرات و الثوابت المحلية.

أين يتم تعريف الدوال

يتم تعريف الدوال على مرحلتين : نبدأ بتعريف واجهة الدالة في المنطقة الإفتتاحية قبل بداية البرنامج، ثم نعطي التعريف الكامل للدالة في أسفل البرنامج.
لايمكن تعريف دالة داخل دالة أخرى.

4. أنواع عناصر الدوال في C++

C++ كغيرها من لغات البرمجة تميز بين نوعين من العناصر::

- عناصر تستخدم في إدخال (إعطاء) قيم إلى الدالة ولاتصلح لتلقي قيم من الدالة (أي أنها لاتتأثر بمايدور داخل الدالة)، وتعرف عند المبرمجين بعناصر القيمة value parameters. في النص أدناه سنطلق على هذا النوع "عناصر الإدخال" Value Parameters.
- عناصر تتأثر بما يدور داخل الدالة و يمكن استخدامها في تلقي قيم من الدالة، كما يمكن استخدامها في تمرير قيم إلى الدالة. هذا النوع يعرف عند المبرمجين بالعناصر

المتغيرة variable parameters أو العناصر المرجعية reference parameters، وسنطلق عليها عناصر التلقي Variable Parameters في النص أدناه.

1.5 عناصر الإدخال: عناصر ذات اتجاه واحد

كل العناصر التي استخدمناها حتى الآن عبارة عن عناصر إدخال. مثلاً الدالة PritLine:

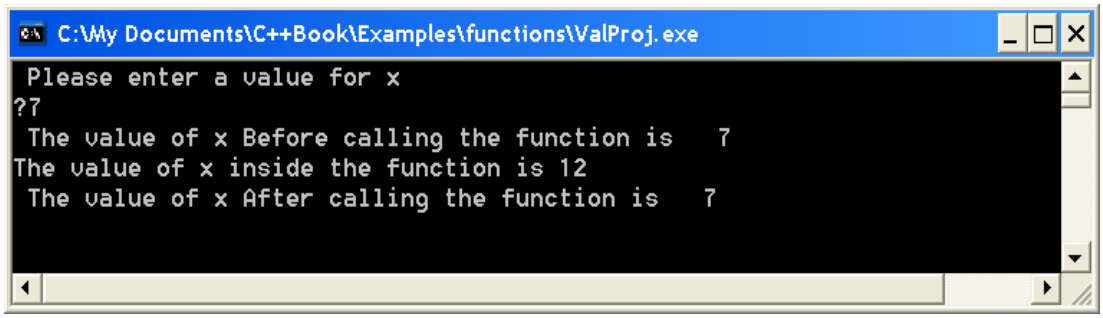
```
void PrintLine (char c, int L)
{ for (int k=0; k<L; k++)
cout <<c;
cout <<endl;
}
```

تستخدم العنصرين c و L وكلاهما عنصرا إدخال يمكن استخدامهما في تمرير قيم إلى الدالة، و لايمكن استخدامهما في تلقي قيم من الدالة ولاتتأثر بما يدور داخل الدالة. المثال التالي يوضح كيف أن عناصر القيمة لاتتأثر بما يدور داخل الدالة، ولايمكن استخدامها في تمرير قيم من الدالة إلى البرنامج:

مثال(3)

```
#include <iostream.h>
void Modify( int );
int main( )
{ int x;
cout << " Please enter a value for x\n?";
cin >> x;
cout <<" The value of x Before calling the function is "<<x<<endl;
Modify(x);
cout <<" The value of x After calling the function is  "<<x<<endl;
cin >> x;
return 0;
}
void Modify(int r)
{ r = r + 5;
cout << "The value of x inside the function is "<<r<<endl;
}
```

عينة من تنفيذ البرنامج:



كما هو ملاحظ من تنفيذ البرنامج أعلاه أن قيمة x في البرنامج لم تتغير بالرغم أن الدالة أضافت 5 إلى القيمة الأصلية. السبب في ذلك أنه في حالة التمرير بالقيمة، فإن النظام لايمرر x نفسها إلى الدالة عند استدعائها، و **إنما ينشئ نسخة من x ويمررها إلى الدالة**. الدالة تقوم بإجراء التعديل على النسخة الممررة. عند انتهاء الدالة تمسح النسخة من الذاكرة.

2.5 عناصر التلقي (العناصر المتغيرة) (عناصر ذات اتجاهين)

رأينا في مثال أعلاه أن عناصر الإدخال لا يمكن استخدامها في إرجاع قيم أو نتائج من الدالة إلى البرنامج. وذكرنا أن البرنامج يعطي الدالة نسخة من العنصر الممرر و لايعطيها العنصر نفسه.

إذا أردنا أن نمرر قيمة من الدالة إلى البرنامج من خلال العناصر الممررة، فلابد من تعريف هذه العناصر كعناصر متغيرة variable parameters أو مرجعية reference parameters، كما تسمى أحياناً. العناصر المتغيرة تعرف بإضافة الرمز & قبل اسم المتغير في التعريف. إذا أردنا أن يتم التعديل على المتغير x بالصورة المطلوبة في المثال أعلاه، فلابد من تعريف العنصر الممرر للدالة r كعنصر مرجعي &r:

```
void Modify(int &r)
{ r = r + 5;
cout << "The value of x inside the function is "<<r<<endl;
}
```

كذلك تعريف واجهة الدالة في أعلى البرنامج ينبغي أن يصبح كما يلي:

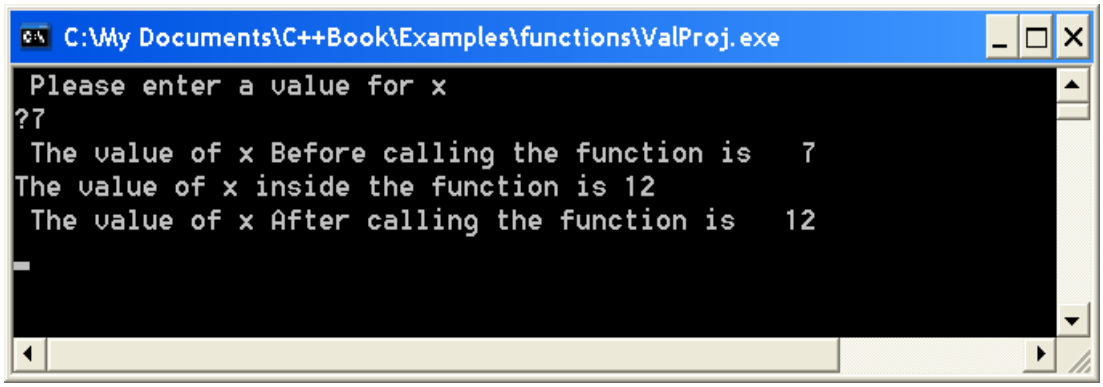
void Modify(**int &r**);

البرنامج الكامل بعد التعديل:

مثال (4)

```
#include <iostream.h>
void Modify( int &);
int main( )
{ int x;
cout << " Please enter a value for x\n?";
cin >> x;
cout <<" The value of x Before calling the function is "<<x<<endl;
Modify(x);
cout <<" The value of x After calling the function is  "<<x<<endl;
cin >> x;
return 0;
}
void Modify(int &r)
{ r = r + 5;
cout << "The value of x inside the function is "<<r<<endl;
}
```

عينة من تنفيذ البرنامج بعد التعديلات:



كما هو ملاحظ من شاشة التنفيذ، فإن التعديل تم كما هو مطلوب في هذه الحالة و أصبحت قيمة x 12 كما هو متوقع. السبب في ذلك أنه تم تمرير x نفسها للدالة في هذه الحالة، و ليس نسخة منها.

إضافة الرمز & قبل اسم المتغير على النحو الذي رأيناه في المثال أعلاه يجعل x عنصراً ذا اتجاهين أو على الأقل قابلاً لتمرير نتائج من الدالة إلى البرنامج الأب.



مثال (5)

يوضح استخدام العناصر المتغيرة في مسألةٍ أكثر جدوى من المثال أعلاه. الهدف من هذا المثال هو إجراء عملية تبديل بين قيمتي متغيرين x,y. نبدأ أولاً بكتابة الدالة ثم نشرح كيفية استدعائها في برنامج:

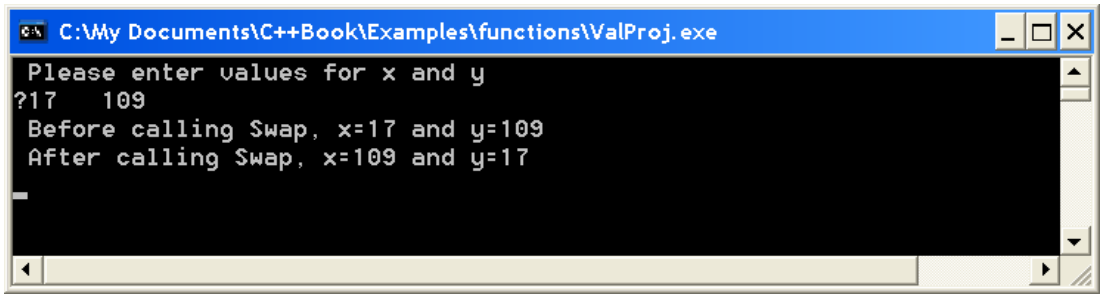
```
void  Swap(float &a, float &b)
{ float temp;
   temp=a;
     a=b;
   b=temp;
   }
```

الآن نستخدم الدالة في برنامج:

```
#include <iostream.h>
void  Swap(float &, float &);
int main( )
{ float x,y;
  cout << " Please enter values for x and y\n?";
  cin >> x>>y;
  cout <<" Before calling Swap, x="<<x<<" and y="<<y<<endl;
  Swap(x,y);
  cout <<" After calling Swap, x="<<x<<" and y="<<y<<endl;
  cin >> x;
  return 0;
}
 void  Swap(float &a, float &b)
 { float temp;
   temp=a;
```

```
  a=b;
  b=temp;
}
```

عينة من تنفيذ البرنامج:

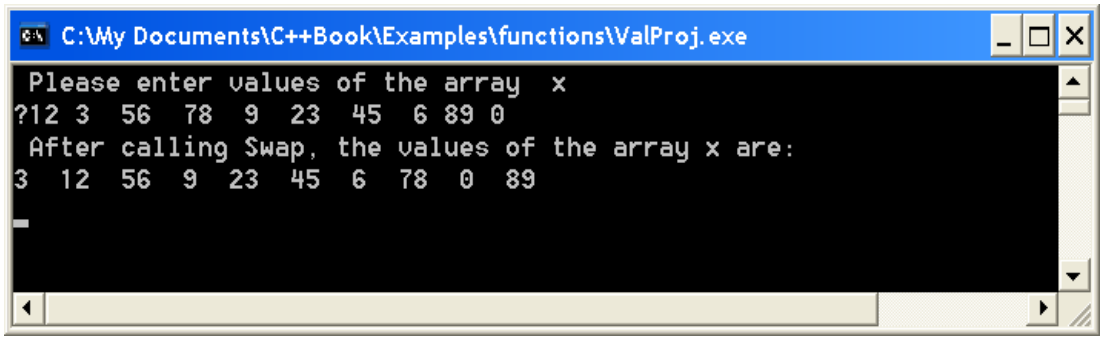


يمكن تطوير البرنامج أعلاه للتعامل مع مصفوفة كاملة و تبديل بعض عناصرها إذا استوفي شرطٌ معين. البرنامج أدناه يقرأ مصفوفة x ويقوم بتبديل كل عنصرين متجاورين إذا وجد أن السابق منهما أكبر من اللاحق:

مثال (6):

```
#include <iostream.h>
void  Swap(float &, float &);
int main( )
{ float x[10];
cout << " Please enter values of the array  x \n?";
for (int k=0; k<10; k++)
cin >> x[k];
for (int k=0; k<9; k++)
if (x[k] > x[k+1])
Swap(x[k],x[k+1]);
cout <<" After calling Swap, the values of the array x are: "<<endl;
for (int k=0; k<10; k++)
cout <<x[k]<<"  ";
cout <<endl;
cin >> x;
return 0;
}
void  Swap(float &a, float &b)
{ float temp;
temp=a;
a=b;
b=temp;
}
```

عينة من تنفيذ البرنامج:



مثال (7)

اكتب دالة تحسب مساحة مثلث باستخدام العلاقة:

$$Area = \sqrt{S(S-A)(S-B)(S-C)}$$

حيث

$$S=(A+B+C)/2$$

و A,B,C هي أطوال ضلع المثلث.

الحل:

الدالة المطلوبة يمكن صياغتها على النحو التالي:

```
float area(float A, float B, float C)
{ float S, TriArea;
S=(A+B+C)/2;
TriArea=sqrt(S*(S-A)*(S-B)*(S-C));
return TriArea;
}
```

مثال (8):

اكتب دالة تستقبل (كعناصر) عدد الساعات التي عملها موظف في أسبوع والأجرة التي تعطى للساعة الواحدة (rate)، ثم تحسب الأجر الأسبوعي للموظف حسب اللائحة التالية إذا كان عدد الساعات التي عملها الموظف أقل أو تساوي أربعين فإن :

الأجر الأسبوعي = عدد الساعات × أجرة الساعة

1- إذا كان عدد الساعات أكثر من 40 و أقل من أو يساوي 50، فإن:

الأجر الأسبوعي = (عدد الساعات × أجرة الساعة) + 15×أجر الساعة

2- أما إذا كان عدد الساعات أكثر من 50 فإن :

الأجر الأسبوعي = (عدد الساعات × أجرة الساعة) + 20×أجر الساعة

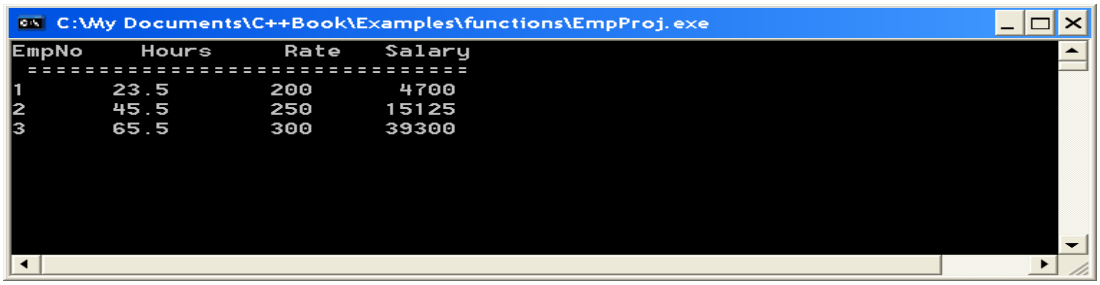
اكتب برنامجاً يقرأ بيانات عدد من الموظفين مخزنة في ملف Emp.txt ويستخدم الدالة أعلاه في حساب الأجر الأسبوعي لكل موظف و يطبع الناتج على الشاشة.

البرنامج:

```
#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>
float Salary(float, float);
int main()
{ ifstream InFile("Emp.txt", ios::in);
  float Hours, Rate, WeekSalary;
  int EmpNo;
  cout<<"EmpNo"<<setw(9)<<"Hours"<<setw(9)<<"Rate"<<setw(9)
<<"Salary"<<endl;
  cout <<" ================================"<<endl;
  while (InFile >>EmpNo>>Hours>>Rate)
  {  //InFile >>EmpNo>>Hours>>Rate;
  WeekSalary = Salary(Hours, Rate);
  cout <<EmpNo<<setw(10)<<Hours<<setw(10)<<Rate
        <<setw(10)<<WeekSalary<<endl;
  }
return 0;
}
float Salary(float hours, float rate)
```

```
{ float S, B;
  S= hours * rate;
  if (hours <= 40)  B=0;
  else if ((hours>40) && (hours <=50))  B=15 * rate;
  else if  ((hours>50) && (hours <=60)) B=20 * rate;
  else B=S;
  S = S + B;
  return S;
}
```

عينة من تنفيذ البرنامج:



6. تمرير المصفوفات كعناصر للإجراءات و الدوال

يجوز تمرير المصفوفات كعناصر للدوال كغيرها من المتغيرات، ولكن دون إعطاء حجمها في التعريف مباشرةً. مثلاً الدالة التالية تقوم بحساب المجموع ومجموع المربعات لمصفوفة معينة:

```
void SumArray(float x[],  int  size, float &sum, float &sumofsq)
                                                    { int  k;
                                          sum=0.0;   sumofsq=0.0;
                                           for (k=1; k<=size; k++)
                                                 {  sum=sum + x[k];
                                        sumofsq= sumofsq + sqr(x[k]);
                                                                  }
                                                                  }
```

يلاحظ من تعريف الدالة أعلاه أن المصفوفة x[] تم تعريفها دون إعطاء الحجم الفعلي للمصفوفة بين القوسين المربعين. استخدام القوسين المربعين في تعريف العنصر يدل على أن هذا العنصر

مصفوفة و ليس متغيراً عادياً. المصفوفات تعتبر عناصر مرجعية reference parameters وليست عناصر قيمة في عرف C++.

مثال (9):

اكتب برنامجاً يقرأ مصفوفة معينة ثم يكتشف إذا ما كانت عناصر المصفوفة مرتبة تصاعدياً أم لا. استخدم الدوال ما أمكن ذلك.

تحليل المسألة:

المعطيات :مصفوفة x حجمها n.

المطلوب: تحديد إذا ما كانت المصفوفة مرتبة تصاعدياً أم لا.

المتغيرات الوسيطة: نحتاج لمتغير منطقي Sorted يأخذ القيمة true إذا كانت المصفوفة مرتبة أو القيمة false إذا كانت المصفوفة غير مرتبة. كذلك نحتاج لمتغير تحكم k، مثلاً، لقراءة قيم المصفوفة.

تصميم الخوارزمية:

1- إقرأ المصفوفة.

2- حدد إذا ما كانت المصفوفة مرتبة أم لا.

3- اطبع النتيجة.

الخطوة II أعلاه مكن تفصيلها كما يلي:

1- افترض أن المصفوفة مرتبة Sorted=true

2- كرر لكل عنصر من عناصر المصفوفة x:

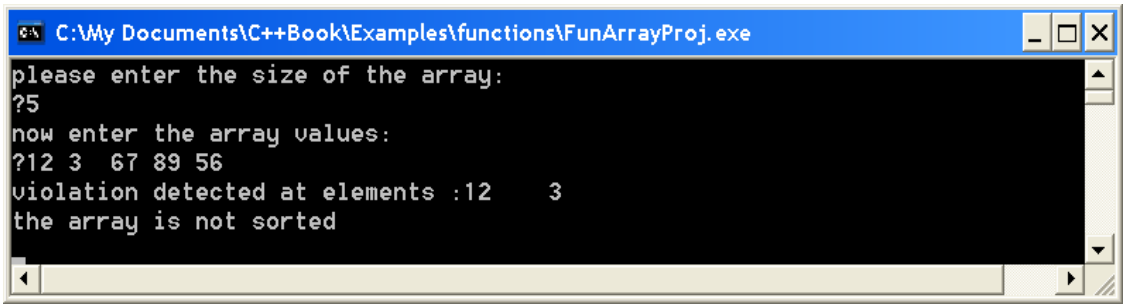
3- إذا كان العنصر الحالي أكبر من العنصر الذي يليه في المصفوفة فإن الافتراض 1 يكون خاطئاً وعليه Sorted=false.

يمكن استخدام دالة لقراءة المصفوفة (الخطوة أ)، كما يمكن استخدام دالة لتنفيذ الخطوة ب. التفاصيل في البرنامج أدناه.

البرنامج:

```
#include <iostream.h>
void ReadArray(int x[], int n);
bool TestSorted(int x[], int n);
int main(int argc, char* argv[])
{  int MyArray[10];
int N;
bool Sorted;
cout <<"please enter the size of the array:\n?";
cin >> N;
cout << "now enter the array values:\n?";
ReadArray(MyArray, N);
//now test if the array is sorted or not
Sorted = false;
Sorted = TestSorted(MyArray, N);
// now print the result
if (Sorted) cout << " the array is sorted "<<endl;
else cout << "the array is not sorted"<<endl;
cin >>N;
return 0;
}
void ReadArray(int x[], int n)
{ int i;
for (i=1; i<=n; i++)
cin >> x[i];
}
bool TestSorted(int x[], int n)
{ int i; bool ArraySorted;
ArraySorted = true;
i=1;
while ((i<n) && ArraySorted)
{ if (x[i] > x[i+1])
{ ArraySorted = false;
cout << "violation detected at elements :"
<<x[i]<< "    "<<x[i+1]<<endl;
}
else i = i+1;
}
return ArraySorted;
}
```

عينة من تنفيذ البرنامج:

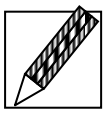


أسئلة تقويم ذاتي



1- ما هو الفرق بين عناصر القيمة و العناصر المتغيرة في دوال ++C؟
2- ما هو الفرق بين الدالة التي ترجع قيمة والدوال التي لاترجع قيمة void في ++C؟
3- هل بالإمكان استخدام جملة return في الدوال التي لاترجع قيمة void في ++C؟

تدريب (1)



1- اكتب دالة تساعد المبرمج في زخرفة النتائج المطبوعة بطباعة عدة اسطر باستخدام الرمز الذي يختاره. مثلاً: ..... My results
الدالة ينبغي أن تعطي المستخدم الخيار في اختيار الرمز الذي يستخدم في رسم الخطوط وعدد الخطوط.
2- اكتب دالة تحسب عدد مرات تكرار رموز العمليات و عدد جمل الإسناد الواردة في برنامج معين مكتوب بلغة ++C أفرض أن عدد رموز العمليات يعطى في المتغير Operator وعدد جمل الإسناد يعطى في المتغير AssignStmts. رموز العمليات تشمل {+, -, *, /}. عدد جمل الإسناد يكافئ عدد رموز الإسناد "=" الواردة في البرنامج.
3- اكتب دالة تحسب القيمة القصوى و القيمة الدنيا في مصفوفة معينة وتعدِّل عناصر المصفوفة بقسمة كل عنصر على القيمة القصوى.
4- اكتب دالة تحسب حاصل الضرب النقطي لمصفوفتين.

الخلاصة

عزيزي الدارس،

اشتملت هذه الوحدة على شرح للبرمجة التركيبية و تطبيق لذلك بتوظيف الدوال بأنواعها المختلفة وقد تم التعرف بعناصر الدوال وطرق استخدامها داخل البرنامج لحل المسائل المعقدة و إستخلاص النتائج بصورة منظمة وسلسة يسهل فهمها و استيعابها.

وحتى نستطيع عزيزي الدارس اتقان مهارات البرمجة نرجو منك إدخال جميع الأمثلة والتأكد من حلولها، وحل جميع التدريبات.

لمحة مسبقة عن الوحدة الدراسية التالية

عزيزي الدارس،

سنتناول في الوحدة القادمة موضوع السجلات وخوارزمية برمجتها وشرح للمتغيرات العامة والسجلات المتعددة الأوجه، وستسهم الوحدة القادمة في منحك صورة أوضح عن الوحدة الحالية والوحدات السابقة.

نرجو أن تستمتع بدراستها، ونتمنى لك التوفيق.

مسرد المصطلحات

الدوال التي ترجع قيمة واحدة Typed Functions

هي نوع من الدوال التي تُستدعى في البرنامج لحساب قيمة واحدة من علاقة حسابية معينة.

الدوال التي لا ترجع أي قيمة مباشرة عند استدعائها Void Functions

هي نوع من الدوال يرجع قيم بصورة غير مباشرة من خلال عناصر التمرير (ملفات).

واجهة الدالة:

هي عنوان الدالة والذي يتركب من نوع الدالة واسم الدالة وقائمة العناصر .

عناصر الادخال Value Parameters:

هي العناصر التي يتم عن طريقها ادخال قيم للدوال .

عناصر المتلقي Variable Parameters:

هي العناصر التي يمكن استخدامها في تلقي قيم من الدالة.

المصادر والمراجع

1) Deictl, H. & Deitel, P., **C++ How to Program.** Third Edition, Prentice Hall: 2001.
2) Savitch, W. **Problem Solving With C++: The Object of Programming**. Third Edition, Addison Wesley: 1999.
3) Friedman, F. and Koffman, E. **Problem Solving, Abstraction, and Design Using C++** (2 nd Edition). Addison Wesley: 1996.
4) Lippman, S. lajoie, J. **C++ Primer,** (3rd Edition). Addison Wesley: 1998.
5) Horstmann, C. **Computing Concepts With C++ Essentials.** John Wiley: 1997.
6) Stroustrup, B. **The C++ Programming Language.** (3 rd edition), Addison Wesley: 1990.
7) Ellis, M. and Bjarnc Stroupstrup, **The Annotated Reference Manual.** Addison Wesley: 1990.
8) Tcale, S. C++ IOS treams Handbook. Addison Wesley: 1993.
9) Meyers, S. **Effective C++: 50 Specific Ways to Improve Your Programs and Designs** (2 nd edition). Addison Wesley: 1996.
10) Meyers, S. **More Effective C++: 35 New Ways to Improve Your Programs and Designs.** Addison Wesley. 1996.
11) Schildt, H**. C++ From the Ground up.** Mc Graw Hill. 2003.
12) Savitch, W. **Absolute C++.** Addison Wesley: 2002.
13) Cogswell, J. **C++ all – in – one Desk Refcrence for Dummies.** 2002.


الوحدة السابعة
السجـــلات

محتويات الوحدة

الموضوع	الصفحة
المقدمة	**159**
تمهيد	159
أهداف الوحدة	159
1. تعريف السجلات	**160**
2. المتغيرات العامة (متعددة الأوجه)	**166**
3. السجلات متعددة الأوجه	**168**
4. الأنواع المتعددة	**173**
الخلاصة	**176**
لمحة مسبقة عن الوحدة التالية	**176**
مسرد المصطلحات	**177**
المصادر والمراجع	**178**

المقدمة

تمهيد

مرحباً بك عزيزى الدارس

بدأنا مشوارنا مع المتغيرات المركبة في الوحدة السادسة عندما تعرضنا للمصفوفات. إنّ هذه المصفوفات تستخدم إذا كانت العناصر المكونة متجانسة. أما إذا كانت العناصر المكونة غير متجانسة، فلا بد إذاً من استخدام نوع آخر من المتغيرات المركبة في C++ ألا و هو السجلات structures.

تحتوي هذه الوحدة على السجلات وخوارزمية برمجتها وشرح للمتغيرات العامة والسجلات المتعددة الأوجه.

أهداف الوحدة

عزيزي الدارس،

بعد دراستك لهذه الوحدة ينبغي أن تكون قادراً على أن:

- **تفرق** بين السجلات والمصفوفات.
- **تستخدم** السجلات في تصميمك لبرامج بلغة C++.
- **تبين** المتغيرات العامة وكيفية استخدامها في لغة C++.
- **تصمم** سجلات متعددة الأوجه.

1. تعريف السجلات

السجل: عبارة عن متغير مركب يتكون من عدة عناصر، غالباً ما تكون غير متجانسة ويعرّف باستخدام الكلمة الخاصة struct. مثلاً يمكن أن نعرف سجل لطالب معين على النحو التالي:

```
struct Student
    { string name; \\ {the string type is supported by C++}
      int id;
      float result [10] ;
      float gpa;
  };
```

لاحظ أن تعريف السجل يبدأ بالكلمة الخاصة struct، مصحوبة بشعار السجل Student، (شعار السجل المقصود به الإسم الذي يسبق تفاصيل التعريف) ثم عناصر السجل بين الحاصرتين. التعريف أعلاه ليس إلا قالباً فقط و لايعتبر متغيراً من وجهة نظر C++. أي أن Student لايمكن استخدامه كمتغير داخل البرنامج، ولكن يمكن استخدامه في تعريف متغيرات.

داخل البرنامج يمكن تعريف متغير Std من نوع السجل Student كما يلي:

```
Student Std;
```

كما يمكن تعريف متغيرات بإضافتها إلى تعريف السجل (عند نهايته) كما هو الحال في المثال التالي:

```
struct Student
    { string name; \\ {the string type is supported by C++}
      int id;
      float result [10] ;
      float gpa;
  } JuniorStudent, SeniorStudent;
```

في هذه الحالة JuniorStudent و SeniorStudent عبارة عن متغيرات من نوع Student (السجل الذي عرّفناه). لاحظ أنه في هذه الحالة يمكن حذف شعار السجلٍ Student إذا لم تكن هناك ضرورة لاستخدامه في تعريف متغيرات أخرى داخل البرنامج.

بعد ذلك يمكن التعامل مع عناصر السجل باستخدام الأسلوب النقطي. مثلاً:

```
cin >> Std.id);
cin >> Std.name;
Std.gpa = 3.57;
```

مثال (1)

اكتب برنامجاً يقرأ المعلومات المخزنة عن الكتب في مكتبة معينة ثم يطبع أسماء الكتب المنشورة في مجال علوم الحاسب بعد عام 1997. أفرض أن المعلومات مخزنة في ملف معين و تشمل على الأقل الآتي:

اسم الكتاب، رقم الكتاب، رمز التصنيف، مجال الكتاب، اسم المؤلف.

تحليل المسألة:

المعطيات: ملف يحتوي على المعلومات المذكورة أعلاه عن الكتب BooksFile. المعلومات داخل الملف مخزنة في شكل سجلات، كل سجل يحتوي على المعلومات التالية: name, SerialNo, Index, subject, author, year.

المطلوب: البحث عن الكتب المنشورة في مجال علوم الحاسب بعد عام 1997 وطباعة أسمائها.

تصميم الخوارزمية:

تعتبر هذه المسألة تطبيقاً مباشراً على عملية البحث الخطي، حيث نبحث عن معلومات معينة داخل قائمة ثم نطبعها. تفصيل الخوارزمية كالآتي:

كرر حتى نهاية الملف:

1- إقرأ سجل كتاب.

2- إذا كان مجال الكتاب هو 'computers' و سنة النشر>=1997 اطبع اسم الكتاب.

البرنامج:

```
#include <iostream.h>
#include <fstream.h>
struct BookStruct
 {string name;
  int SerialNo;
  string Index;
  string author;
  int year;
  string subject;
 };
int main(int argc, char* argv[])
{   BookStruct Book;
    ifstream BookFile("Books.txt", ios::in);
    while (!BookFile.eof())
    {BookFile>> Book.name
            >> Book.SerialNo
            >> Book.Index
            >> Book.author
            >> Book.subject
            >> Book.year;
     if (!BookFile.eof())
       if ((Book.subject == "computers")||(Book.year>=1997))
          cout << Book.name<<endl;
     }
     }
     cin >> Book.Index;
     return 0;
}
```

عينة من تنفيذ البرنامج:



لعل الدارس يلاحظ أننا استخدمنا "بلوغ نهاية الملف" كشرط مباشر لإنهاء التكرار:

```
while (!BookFile.eof())
{ ……
```

الدالة BookFile.eof() ترجع القيمة 0 عند بلوغ نهاية الملف. المشكلة أن هذه الدالة لاتستطيع إدراك أن الملف قد بلغ نهايته حتى نحاول قراءة قيمة من الملف بعد نهاية البيانات. هذا قد يؤدي إلى تكرار بعض الأوامر مرة إضافية لاحاجة لها. في البرنامج أعلاه يتم تكرار أمر الطباعة cout مرة إضافية بعد بلوغ نهاية الملف. تحسباً لذلك استخدمنا الأمر if (!BookFile.eof()) لمنع هذا التكرار الغبر ضروري.

استخدام السجلات في تعريف أنواع جديدة:

لعلك لاحظت في المثال السابق أننا عرّفنا قالب السجل template موضوع المسألة، ثم استخدمنا هذا القالب في تعريف السجل داخل البرنامج:

```
BookStruct Book;
```

وهذا يعني أن قالب السجل BookStruct استخدم كنوع من أنواع البيانات data type في تعريف المتغير Book، مثله مثل الأنواع الأخرى في C++. في الواقع هذه إحدى الطرق المتبعة في تعريف أنواع بيانات جديدة في C++ (إذا كان النوع مركباً). هناك طرق أخرى نعرض لها فيما بعد إن شاء الله.

مثال (2):

اكتب برنامجاً يقرأ سجلات الطلاب المخزنة في ملف ثم يطبع أسماء طلاب المنطقة الوسطى المتوقع تخرجهم هذا العام، علماً بأن سجل كل طالب يحتوى على إسمه، عنوانه، رقمه الجامعي و التاريخ المتوقع لتخرجه.

تحليل المسألة:

المعطيات: ملف سجلات الطلاب. كل سجل يحتوي على المعلومات التالية: Name, ID, Address, ExpectedGraDate.

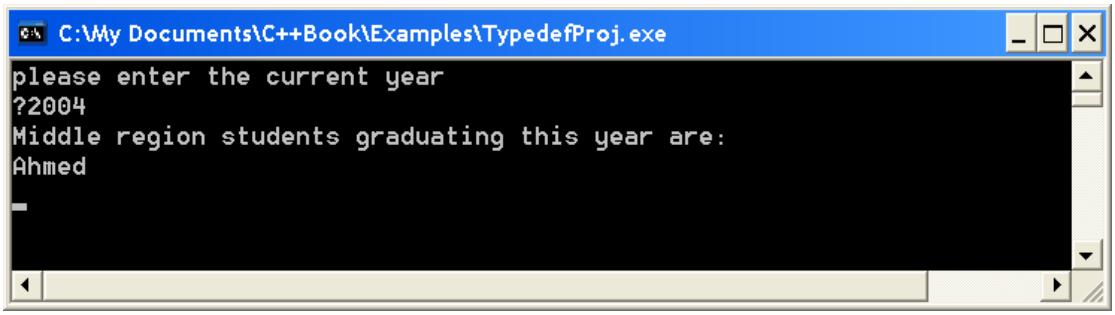
المطلوب: طباعة أسماء طلاب المنطقة الوسطى المتوقع تخرجهم هذا العام.

```
#include <iostream.h>
#include <fstream.h>
typedef int YearType;
typedef struct
 { int day;
   int month;
   YearType year;
 } DateType;
typedef  struct
   { int  BoxNo;
     string City;
     string Region;
 } AddressType;
 struct StudentRecord
   {   string name;
       int id;
       AddressType Address;
       DateType ExpectedGradDate;
    };

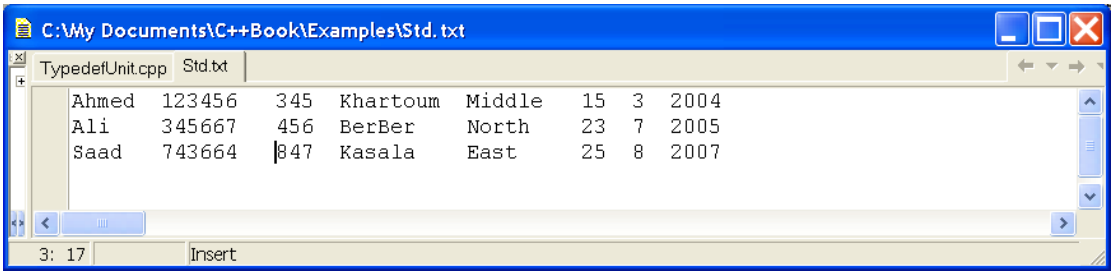
int main(int argc, char* argv[])
{   StudentRecord Student;
    YearType ThisYear;
    ifstream StdFile("Std.txt", ios::in);
    cout <<"please enter the current year\n?";
    cin >>ThisYear;
    cout << "Middle region students graduating this year are:\n";
    while (StdFile >> Student.name>>Student.id)
    { StdFile >> Student.Address.BoxNo;
      StdFile >> Student.Address.City;
```

```
    StdFile >> Student.Address.Region;
    StdFile >> Student.ExpectedGradDate.day;
    StdFile >> Student.ExpectedGradDate.month;
    StdFile >> Student.ExpectedGradDate.year;
    if ((Student.ExpectedGradDate.year==ThisYear)
        && (Student.Address.Region=="Middle"))
         cout <<Student.name<< endl;
    } // while
    return 0;
}
```

عينة من تنفيذ البرنامج:



محتويات ملف البيانات:



لاحظ أننا استخدمنا جملة typdef في تعريف بعض الأنواع الجديدة، وهذه طريقة أخرى لتعريف الأنواع الجديدة. الشكل العام لجملة typedef هو:

typedef < نوع معروف> < نوع جديد >

حيث <نوع معروف> إما أن يكون أحد الأنواع المبنية في اللغة أو نوع معرّف في اللغة باستخدام السجلات أو غيرها.

أسئلة تقويم ذاتي



1- في مثال (2) أعلاه هل يمكن إستخدام الأنواع المعرفة DateType و AddressType في تعريف بيانات داخل البرنامج إذا لم نستخدم جملة typedef في تعريفها؟ (أي إذا تم تعريفها كسجلات عادية).
2- هل يمكن تعريف السجلات داخل البرنامج (داخل الدالة الرئيسية ()main)؟ جرب بنفسك.
3- ماهو الفرق بين شعار السجل و إسم السجل (إسم المتغير من نوع سجل)؟ هل يمكن استخدام شعار السجل كمتغير داخل البرنامج؟
4- هل يمكن تعريف سجلات داخل سجلات؟ مثلاً هل يمكن تعريف سجل Address مباشرةً داخل السجل Student في المثال أعلاه؟ جرب بنفسك و اختبر الناتج.

2. المتغيرات العامة (متعددة الأوجه) unions

المتغير العام عبارة عن منطقة في الذاكرة قابلة لتخزين أنواع مختلفة من القيم، حسبما يقتضيه الحال. أنواع القيم المتوقع تخزينها لابد من تعريفها داخل المتغير العام. عملية التعريف تبدأ عادةً بقالب template يعرّف باستخدام الكلمة الخاصة union كما في المثال التالي:

```
union VarValue
{ int x;
double  y;
string name;
};
```

القالب VarValue أعلاه يمكن استخدامه بعد ذلك في تعريف متغير عام كما يلي:

```
VarValue  MyVar;
```

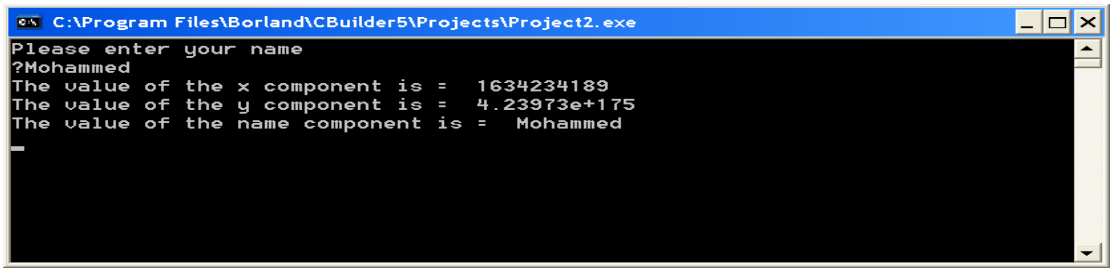
المتغير العام MyVar يمكن أن يأخذ أياً من أنواع القيم المدرجة في تعريف VarValue، ولكن ليست مجتمعة (وهذا هو الفرق بين المتغيرات العامة و السجلات). المتغير العام MyVar في هذه الحالة يمكن أن يلعب دور المتغير x في مرحلة معينة، ودور المتغير y في مرحلةٍ أخرى، ودور المتغير name في مرحلةٍ ثالثة.

المثال التالي يوضح كيفية استخدام المتغيرات العامة في برنامج:

مثال (3)

```
#include <iostream.h>
union VarValue
{ int x;
double y;
char name[20];
};
int main(int argc, char* argv[])
{  VarValue MyVar;
MyVar.x=25;
MyVar.y=12654.87;
cout <<"Please enter your name\n?";
cin >> MyVar.name;
cout <<"The value of the x component is =  "
<<MyVar.x<<endl;
cout <<"The value of the y component is =  "
<<MyVar.y<<endl;
cout <<"The value of the name component is =  "
<<MyVar.name<<endl;
return 0;
}
```

عينة من تنفيذ البرنامج:



كما يبدو من الناتج، أن القيمة الصحيحة للمتغير هي القيمة الأخيرة للمتغير (Mohammed في هذه الحالة). القيمة الأخرى أفسدت بدخول قيمة المتغير الأخير، ذلك أن المتغير العام يشير إلى منطقة واحدة داخل الذاكرة.

3. السجلات متعددة الأوجه Variant Records

كثيراً ما يواجه المصمم حالات تتشابه فيه السجلات و تشترك في بعض الحقول بينما تختلف في بعضها. خذ مثلاً مسألة حساب الرواتب في شركة معينة توظف عمالاً وموظفين مكتبين و باعة متجولين. العمال تحسب رواتبهم بالساعة، بينما تحسب رواتب الموظفين المكتبيين شهرياً. أما الباعة فيعطون راتباً شهرياً أساسياً بالإضافة إلى عمولة محسوبة كنسبة مئوية من حجم المبيعات و بدل نقل. إذا أردنا أن نكتب برنامجاً يحسب الراتب لكل فئة من هؤلاء فسنجد أنفسنا أمام ثلاثة أنواع من السجلات:

سجل العامل:

```
struct  Worker
{ string  name;
   int id;
   float age;
   float RatePerHour;
  float HoursWorked;
 };
```

سجل الموظف المكتبي:

```
struct  OfficeEmp
{ string  name;
   int id;
   float age;
  float Salary;
   float Bonus;
   float Deductions;
  };
```

سجل البائع:

```
struct  SalesEmp
{ string  name;
  int id;
  float age;
 float BaseSalary;
  float CommRate;
  float TAllowance;
  float Sales;
  };
```

ولعله من الواضح لك الآن أن هذه السجلات تشترك في الكثير من الحقول التي يتكرر تعريفها في كل سجل ولعلك ربما تتساءل: هل من سبيلٍ إلى دمج هذه السجلات في سجلٍ واحد بالرغم من الاختلاف الجزئي بينها؟ الإجابة على هذا السؤال: نعم، ففكرة المتغيرات متعددة الأوجه union التي رأيناها في مثال (3) أعلاه يمكن تطبيقها لتكوين سجل متعدد الأوجه يحتوي على جزء ثابت (مشترك) و جزء متغير تتنوع حقوله حسب قيمة حقلٍ خاص يسمى حقل الشعار tag field أو حقل الاختيار. مثلاً الحقول الثلاثة أعلاه يمكن دمجها في سجلٍ واحد على النحو التالي:

```
struct EmployeeStruct
 {string  name;
  int id;
  float age;
  char EmpCode;
  union
    { Worker W;
      OfficeEmp OEmp;
      SalesEmp SEmp;
    };
  };
```

انظر كيف استخدمنا جملة union في تعريف جزء متعدد الأوجه داخل السجل. يمكن استخدام هذا الجزء تارةً في معالجة معلومات العامل، وتارةً في معالجة معلومات الموظف المكتبي، وتارةً في معالجة معلومات البائع المتجول. ويتم معرفة نوع السجل الذي نتعامل معه من خلال حقل الشعار tag field. حقل الشعار في هذه الحالة هو الحقل الجديد:

char EmpCode;

لاحظ أن السجلات الثلاثة المستخدمة في الجزء متعدد الأوجه union تم تقليصها لتحتوي على الجزء المختلف. الجزء المشترك صار هو الجزء الثابت في السجل الجديد. (أنظر المثال أدناه).

هذا السجل يمكن استخدامه في قراءة البيانات و حساب الراتب لكل نوع من أنواع الموظفين أعلاه كما سيتضح من المثال التالي:



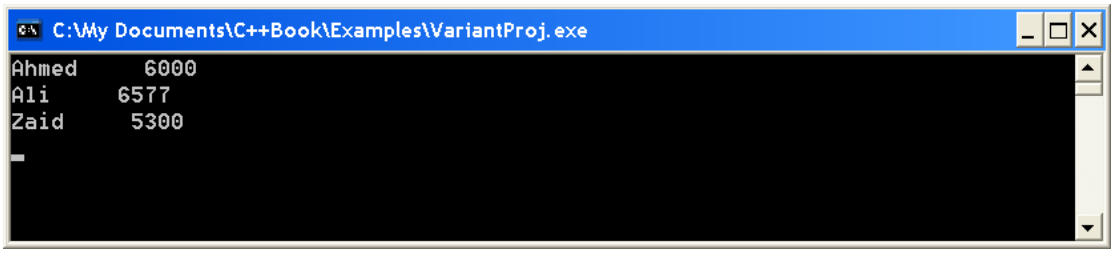
مثال (4)

```
#include <iostream.h>
#include <fstream.h>
struct  Worker
{ float RatePerHour;
float HoursWorked;
};
struct OfficeEmp
{ float Salary;
float Bonus;
float Deductions;
};
struct SalesEmp
{ float BaseSalary;
float CommRate;
float TAllowance;
float Sales;
};
struct EmployeeStruct
{string  name;
int id;
float age;
char EmpCode;
union
{ Worker W;
OfficeEmp OEmp;
SalesEmp SEmp;
};
};
```

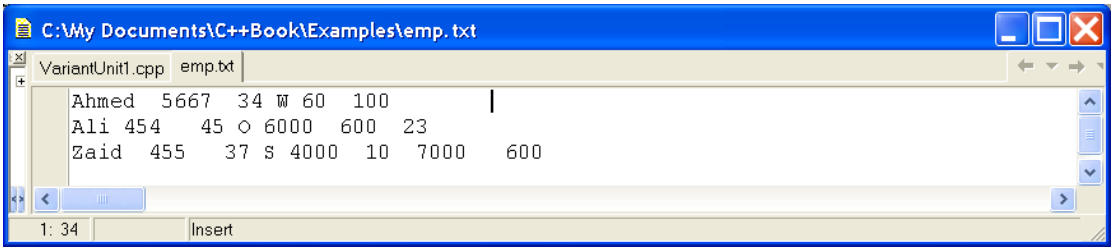


```
int main(int argc, char* argv[])
{ EmployeeStruct Employee;
float payment;
ifstream EmpFile("Emp.txt", ios::in);
while (EmpFile>>Employee.name>>Employee.id>>Employee.age
>>Employee.EmpCode)
{
switch(Employee.EmpCode)
{case 'W':
EmpFile>>Employee.W.RatePerHour
>> Employee.W.HoursWorked;
payment= Employee.W.RatePerHour *
Employee.W.HoursWorked;
break;
case 'O':
EmpFile>> Employee.OEmp.Salary>> Employee.OEmp.Bonus
>> Employee.OEmp.Deductions;
payment= Employee.OEmp.Salary + Employee.OEmp.Bonus
- Employee.OEmp.Deductions;
break;
case 'S':
EmpFile>> Employee.SEmp.BaseSalary>>
Employee.SEmp.CommRate
>> Employee.SEmp.Sales>> Employee.SEmp.TAllowance;
payment= Employee.SEmp.BaseSalary
+ Employee.SEmp.CommRate * Employee.SEmp.Sales/100
+ Employee.SEmp.TAllowance;
break;
default:
cout << "unknown code"<<endl;
} //switch
cout <<Employee.name<<"     "<<payment<<endl;
}// while
return 0;
}
```

عينة من تنفيذ البرنامج:



ملف البيانات الذي استخدم في اختبار البرنامج:



4. الأنواع المعددة Enumerated Types

تسمح C++ بتعريف أنواع محدودة القيم (يمكن سرد القيم الممكنة وقت التعريف). مثلاً يمكن تعريف نوع WeekDay على النحو التالي:

```
enum WeekDay  {Fri, Sat, Sun, Mon, Tus, Wed, Thr};
```

ثم استخدامه داخل البرنامج في تعريف متغبر:

```
WeekDay  Day;
```

☜ لاحظ أننا قمنا بتعداد كل القيم الممكنة للمتغير باستخدام كلمات عادية أو رموز عند تعريف النوع WeeDay. الحاسوب يتعامل مع هذه القيم داخلياً كأعداد صحيحة مرتبة.

الكلمات المدرجة بين القوسين تمثل كل القيم الممكنة للمتغير Day، غير أنها لا تمثل داخل الحاسوب ككلمات، و لكن كأعداد صحيحة تبدأ بالواحد لFri و تنتهي بالسبعة لThr. يمكن إسناد أو مقارنة هذه القيم مباشرةً مع المتغير المعرّف داخل البرنامج. مثلاً بالإمكان إسناد القيمة التالية للمتغير Day:

```
Day=Fri;
```

أو المقارنة:

```
cout << “"The day is Monday”;)Day==Mon(if
```

غير أن إدخال أو طباعة قيم المتغيرات المعددة ليس بنفس السهولة، ذلك لأن الحاسب يتعامل مع هذه القيم كأعداد صحيحة فقط. مثلاً إذا نفذنا الجملتين التاليتين:

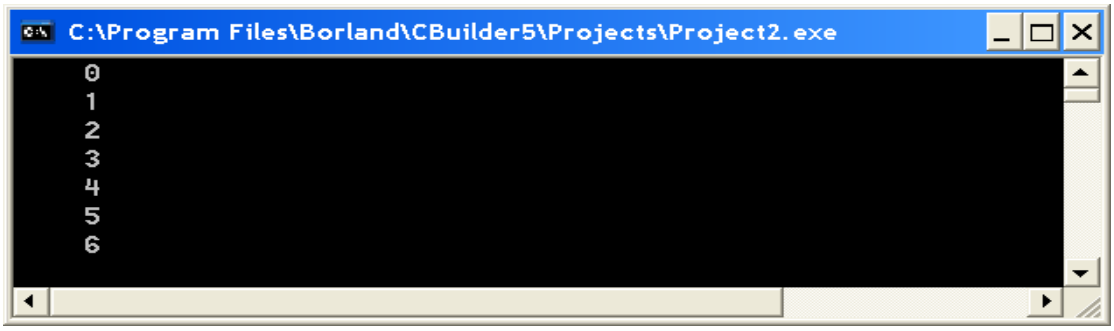
```
Day = Mon;
cout << Day;
```

فإن الحاسب يطبع القيمة العددية المقابلة لليوم Mon. المثال البسيط التالي يعطيك فكرة عما يمكن أن يحدث:

مثال (5):

```
#include <iostream.h>
enum WeekDay {fri, sat, sun, mon, tus, wed, thr};
int main(int argc, char* argv[])
{ WeekDay Day;
for (Day = fri; Day<=thr; Day=Day+1)
cout <<"    "<< Day <<endl;
return 0;
}
```

عينة من تنفيذ البرنامج:



واحدة من فوائد الأنواع المعددة أنه يمكن استخدامها في تعريف أنواع محدودة المدى، مفهومة المعنى للمستخدم.

تدريبات على السجلات:

1- باعتبار التعريفات التالية:

```
enum ColorType  {red, yellow, blue, green, white, black};
struct DressStruct
{ int  Length;
  int SleeveLength;
  ColorType Color;
 };
DressStruct  Dress;
```

اكتب جمل تفعل الآتي:

أ- تسند اللون white إلى الحقل المناسب في المتغير Dress.

ب- تقرأ الطول Length و طول الكم SleeveLength للمتغير Dress.

ت- تقرأ وتخزن المعلومات عن مجموعة من الثياب المطلوبة من بعض الزبائن وتخزنها في ملف.

ث- عدل التعريف أعلاه بحيث يأخذ في الاعتبار تاريخ التسليم وتكلفة الخياطة.

2- الأعداد المركبة تتكون من جزء حقيقي وجزء تخيلي، مثلاً: $x = r + mi$

هذه الأعداد يمكن تمثيلها في C++ باستخدام السجلات. اكتب برنامجاً يقرأ عددين مركبين ثم يحسب مجموعهما. مثّل الأعداد المركبة باستخدام سجل يتكون من حقلين: أحدهما للجزء الحقيقي و الآخر للجزء التخيلي.

3- مكتبة عامة تقدم لقرائها نوعين من المطبوعات: الكتب و المجلات العلمية. ترغب هذه المكتبة في تطوير نظام حاسوبي تدون فيه كل المطبوعات الموجودة لديها بغية تسهيل عملية البحث على روادها. المعلومات المطلوبة تخزينها عن الكتب تشمل الآتي:

الرقم العام ـــــ رقم التصنيف

اسم الكتاب ـــــ اسم المؤلف الناشر ـــــ سنة النشر

المجال

أما بالنسبة للمجلات فتشمل المعلومات الآتي:

الرقم العام ـــــ رقم التصنيف

اسم المجلة ـــــ العدد المجلد ـــــ تاريخ النشر

المجال

استخدم سجلات لتمثيل هذه المطبوعات ومن ثم اكتب برنامجاً يمكن استخدامه في البحث عن مطبوعة معينة.

الخلاصة

في هذه الوحدة وقفنا على تعريف السجل باستخدام الكلمة Struct في لغة C++ وتعريف المتغير باضافة تعريفه في نهاية تعريف السجل وطبقنا نماذج على ذلك، كذلك تعرفنا المتغيرات العامة وكيفية الاستدلال عليها داخل البرامج باستخدام كلمة Unions وأخيراً تعرفنا سجلات متعددة الأوجه وكيفية استخدامها داخل برامج لغة C++.

ننصحك عزيزى الدارس بتطبيق جميع الأمثلة على جهاز الحاسوب حتى تتمكن من استيعاب هذه الوحدة جيداً.

لمحة مسبقة عن الوحدة التالية

سندرس في الوحدة التالية المؤشرات واستخدامها داخل البرامج بلغة C++ كذلك سنتعرف على برمجة المؤشرات وعلاقتها بالمصفوفات. أيضاً في الوحدة القادمة سندرس باذن الله استخدام المؤشرات تخزين القوائم وكيفية حل مشكلة القوائم الضائعة باستخدام فكرة القوائم الديناميكية المترابطة، وكيفية البحث عن عنصر في هذه القوائم وأيضاً كيفية حذف عنصر منها.

مسرد المصطلحات

السجلات Structures

هو عبارة عن متغير مركب يتكون من عدة عناصر تكون متجانسة في الغالب.

المتغيرات العامة متعددة الأوجة: Unions

هي عبارة عن منطقة في الذاكرة قابلة لتخزين أنواع مختلفة من القيم حسبما يقتضيه الحال وتستخدم كلمة Union في تعريف جزء متعد الأوجة داخل السجل.

السجلات متعددة الأوجه Variant Records:

هي السجلات المتشابهة التي تشترك في بعض الحقول وتختلف في بعضها الآخر.

الأنواع المعددة Enumerated Types:

هي المتغيرات لتي لها أنواع محددة القيم وتسمح لغة C++ بتعريف هذه القيم داخل أقواس {...}.

المصادر والمراجع

1) Deictl, H. & Deitel, P., **C++ How to Program.** Third Edition, Prentice Hall: 2001.
2) Savitch, W. **Problem Solving With C++: The Object of Programming**. Third Edition, Addison Wesley: 1999.
3) Friedman, F. and Koffman, E. **Problem Solving, Abstraction, and Design Using C++** (2 nd Edition). Addison Wesley: 1996.
4) Lippman, S. lajoie, J. **C++ Primer,** (3rd Edition). Addison Wesley: 1998.
5) Horstmann, C. **Computing Concepts With C++ Essentials.** John Wiley: 1997.
6) Stroustrup, B. **The C++ Programming Language.** (3 rd edition), Addison Wesley: 1990.
7) Ellis, M. and Bjarnc Stroupstrup, **The Annotated Reference Manual.** Addison Wesley: 1990.
8) Tcale, S. C++ IOS treams Handbook. Addison Wesley: 1993.
9) Meyers, S. **Effective C++: 50 Specific Ways to Improve Your Programs and Designs** (2 nd edition). Addison Wesley: 1996.
10) Meyers, S. **More Effective C++: 35 New Ways to Improve Your Programs and Designs.** Addison Wesley. 1996.
11) Schildt, H**. C++ From the Ground up.** Mc Graw Hill. 2003.
12) Savitch, W. **Absolute C++.** Addison Wesley: 2002.
13) Cogswell, J. **C++ all – in – one Desk Refcrence for Dummies.** 2002.


الوحدة الثامنة
المؤشـــرات

محتويات الوحدة

الموضوع	الصفحة
المقدمة	**181**
تمهيد	181
أهداف الوحدة	181
1. كيف نعرف المؤشرات؟	**182**
2. استخدام المؤشرات داخل البرنامج	**183**
3. العمليات الحسابية على المؤشرات	**187**
4. المؤشرات والمصفوفات	**190**
5. المؤشرات وسلاسل الحروف	**192**
6. استخدام المؤشرات في تخزين القوائم	**193**
1.6 القوائم الضائعة	193
2.6 القوائم الديناميكية المترابطة	194
الخلاصة	**200**
لمحة مسبقة عن الوحدة التالية	**200**
مسرد المصطلحات	**201**
المصادر والمراجع	**202**

المقدمة

تمهيد

مرحباً بك عزيزي الدارس

المؤشرات عبارة عن نوع خاص من المتغيرات لا يحتوي مباشرةً على القيمة المعنية وإنما يحتوي على عنوانها في الذاكرة (أي يشير إليها) ولها استخدامات واسعة في التعامل مع التراكيب الديناميكية. التراكيب الديناميكية هي التراكيب التي يتغير حجمها أثناء تنفيذ البرنامج و يصعب تقدير حجمها الحقيقي عند كتابة البرنامج.

تأتي هذه الوحدة لتوضح لنا معنى المؤشرات وكيفية استخدامها داخل البرنامج لاجراء العمليات الحسابية. كما سنتعرض لعلاقة المؤشرات بالمصفوفات وكيفية استخدام المؤشرات في تخزين القوائم وعليه نجد أن هذه لوحدة ذات أهمية خاصة في علم ادارة المعلومات وقواعد البيانات.

أهداف الوحدة

عزيزي الدارس،

بعد فراغك من دراسة هذه الوحدة ينبغي أن تكون قادراً على أن:

- **تعرّف** معنىالمؤشرات واستخداماتها داخل البرامج.
- **تبين** العلاقة بين المؤشرات والمصفوفات.
- **تتعرف** أنواع القوائم.
- **تصمم** برامج لتخزين القوائم وإدارة البيانات واستخدام المؤشرات بلغة C++.
- **تحذف** أو تضيف عنصرا للقوائم باستخدام المؤشرات.

1. كيف تعرّف المؤشرات؟

لا تعرّف المؤشرات باستخدام كلمة مباشرة مثل ما هو الحال في المتغيرات البسيطة الأخرى التي تعرضنا لها، و إنما تعرف باستخدام النجمة * قبل اسم المتغير، وبعد ذكر نوعه :

```
int  *xPtr, x, y,z;
double  *dPtr, p, s;
```

في الأمثلة أعلاه xPtr و dPtr عبارة عن مؤشرات تشير إلى أماكن في الذاكرة. xPtr يشير إلى مكان يصلح لتخزين الأعداد الصحيحة، بينما يشير dPtr إلى مكان لتخزين الأعداد الحقيقية الكبيرة.

قاعدة (1)

>نوع المتغير>* <اسم المتغير; <

مزيد من الأمثلة على تعريف المؤشرات:

```
int *pi;
float *pf;
char *pc;
Bool  *pb;
```

أسئلة تقويم ذاتي



إذا كان حجم الذاكرة RAM على جهازك 32MB،
ماهو أكبر عنوان على جهازك؟
ماهو الفرق بين العناوين والأعداد الصحيحة؟

.استخدام المؤشرات داخل البرنامج

ذكرنا في الفقرة أعلاه أن المؤشرات تحتوى على عنوان القيمة المشار إليها في ذاكرة الحاسب لا القيمة نفسها. مثلاً، المتغير pi في المثال أعلاه يستخدم للإشارة إلى مكان في الذاكرة يمكن أن يخزن فيه عدد صحيح. السؤال الذي يطرح نفسه: كيف يتسنى للبرنامج ربط المؤشرات بعناوين فعلية في الذاكرة وكيف يمكن تخزين قيم في هذه العناوين أو استرجاع قيم منها؟ أو بصيغة أخرى: من أين نحصل على قيم المؤشرات.

الإجابة عن هذا السؤال أن لغة C++ و شبيهاتها من لغات البرمجة توفر عوامل operators لربط المؤشرات بعناوين فعلية في الذاكرة. مثلاً، في C++، يمكن استخدام العامل new والعامل & (address operator) لربط المؤشر بعنوان معين في الذاكرة قبل استخدامه لتخزين أية معلومات. المثال التالي يوضح كيف يتم ربط المؤشرات بعناوين في الذاكرة و كيف يمكن إيداع قيم في الأماكن التي تشير إليها هذه العناوين:

```
int *iPtr;
pi =  new int;
cin >> pi*;  {pi} إقرأ عدداً صحيحاً وخزنه في المكان الذي يشير إليه المؤشر
```

أو

```
int *iPtr;
iPtr =  new int; // إنشاء نسخة من المؤشر و إعطاؤه قيمة فعلية في الذاكرة
cin >> x;
*iPtr = x; {iPtr} ضع x في المكان الذي يشير إليه المؤشر
```

لاحظ أن الجملة iPtr * = x; تسند قيمة إلى المكان الذي يشير إليه المؤشر iPtr، لا إلى iPtr نفسه. عموماً إذا كان لدينا مؤشران pi و pj من نفس النوع فإن الجملة

```
*pi = *pj;
```

تسند القيمة التي يشير إليها pj إلى pi.

أما الجملة:

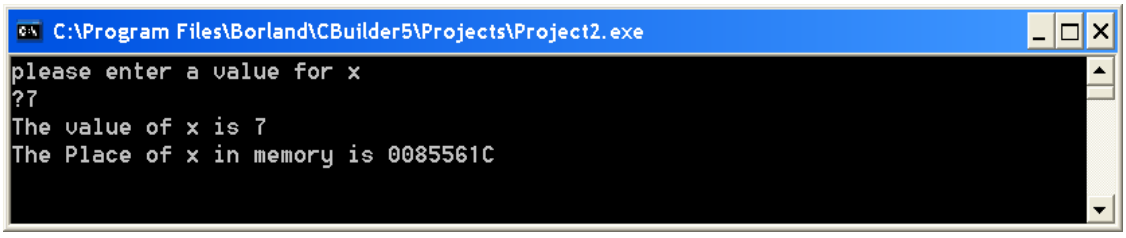
pi = pj;

فتغير قيمة المؤشر نفسه ليحتوى على العنوان pj.

المثال البسيط أدناه يوضح كيف يستخدم العامل new لربط المؤشر بقيمة فعلية في الذاكرة. قيمة المؤشر الفعلية تظهر في نافذة ناتج البرنامج أدناه. لاحظ أن استخدام الجملة cout <<iPtr; يطبع قيمة المؤشر (عنوان منطقة معينة في الذاكرة)، بينما الجملة cout << *iPtr; تطبع قيمة المتغير الذي يشير إليه المؤشر.

مثال (1):

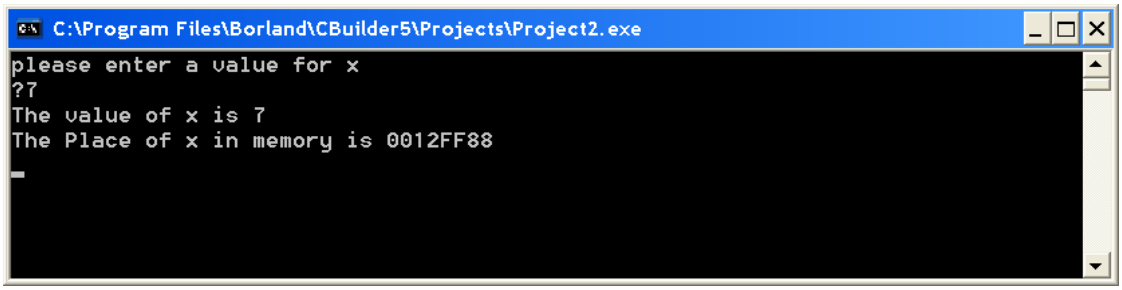
```
#include <iostream.h>
int main(int argc, char* argv[])
{ int x;
int *xPtr;
xPtr = new int;
cout << "please enter a value for x\n?";
cin >> x;
*xPtr = x;
cout <<"The value of x is "<<*xPtr<<endl
<<"The Place of x in memory is "<<xPtr<<endl;
cin >>x;
return 0;
}
```



المثال أدناه يوضح كيفية استخدام العامل & لربط المؤشر بعنوان متغير معين. الجملة iPtr = &x; تأخذ عنوان المتغير x وتخزنه في المتغير iPtr.

مثال (2)

```
#include <iostream.h>
int main(int argc, char* argv[])
{ int x;
int *xPtr;
xPtr = &x;
cout << "please enter a value for x\n?";
cin >> x;
*xPtr = x;
cout <<"The value of x is "<<*xPtr<<endl
<<"The Place of x in memory is "<<xPtr<<endl;
cin >>x;
return 0;
}
```



أسئلة تقويم ذاتي

1- ماهو الناتج المتوقع عند تنفيذ الجمل التالية:

a. `float y=15.49, *yPtr; yPtr=&y; cout <<*yPtr;`

b. `int z=7, *aPtr; aPtr=new int; *aPtr=z+1; cout<<aPtr<<" "<<*aPtr;`

2- اكتب برنامجاً يقرأ قيمة ثلاثة متغيرات حقيقية x,y,z ويطبع عنوان كل منها في الذاكرة.

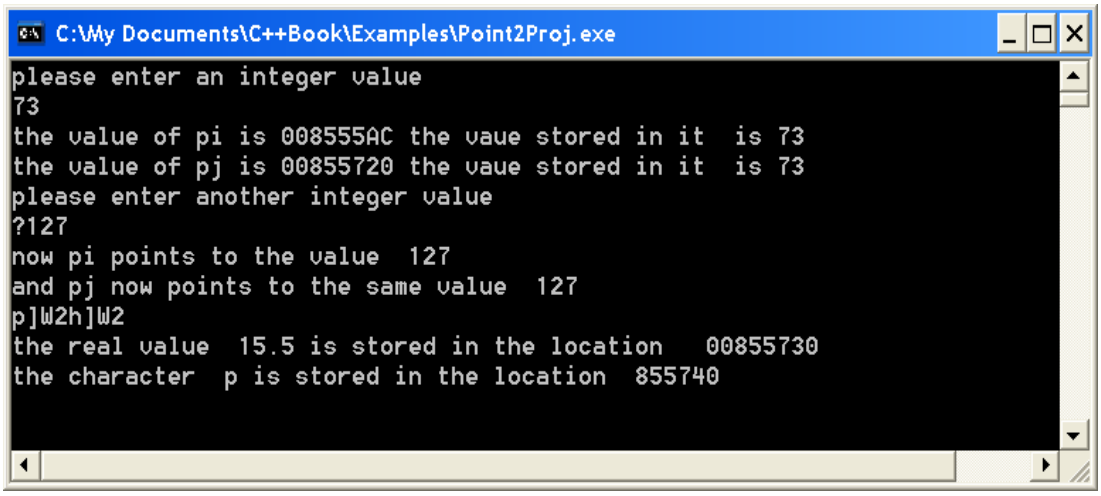
3- اكتب برنامجاً يطبع محتويات بعض العناوين في ذاكرة الحاسب الذي تعمل عليه.

المثال المبسط أدناه يوضح كيفية استخدام المؤشرات في برنامج

مثال (3)

```
#include <iostream.h>
#include <iomanip.h>
int main(int argc, char* argv[])
{   int *pi, *pj, x;
    float *pr, z;
    char *pc, ch;
    bool pb;
    pi = new int;
cout << "please enter an integer value\n";
    cin >> *pi;
cout <<"the value of pi is " << pi
<<" the vaue stored in it  is " << *pi<<endl;
    pj = new int;
    *pj = *pi;
cout << "the value of pj is "<< pj
<<" the vaue stored in it  is "<< *pj<<endl;
cout << "please enter another integer value\n?";
    cin >> x;
    *pi=x;
cout << "now pi points to the value  "<< *pi<<endl;
    pj=pi;
cout << "and pj now points to the same value  "<< *pj<<endl;
    z=15.5;
    pr = new float;
    *pr = z;
    ch='p';
    pc = new char;
    *pc=ch;
cout <<pc <<endl;
cout <<"the real value  "<< z
<<" is stored in the location   "<<pr<<endl;
cout << "the character  "<<*pc
<<" is stored in the location  " << pc<< endl;
    cin >> x;
    return 0;
    }
```

عينة من تنفيذ البرنامج:



3. العمليات الحسابية على المؤشرات

يمكنك عزيزي الدارس، إجراء بعض العمليات الحسابية على المؤشرات مثل الإضافة و الطرح، لكن ينبغي ملاحظة القيمة الجديدة للمؤشر إذ أنها تعتمد على نوع القيمة التي يشير إليها المؤشر. مثلاً، إذا كان iPtr يشير إلى عدد صحيح في الذاكرة، فإن إضافة 1 إلى قيمة iPtr تعني في الواقع إضافة 4 بايتات إلى iPtr:

```
int *iPtr;
iPtr = new int;
iPtr = iPtr + 1;
```

السبب في ذلك أن المتغير الصحيح (int) يحتاج إلى أربع بايتات في الذاكرة.

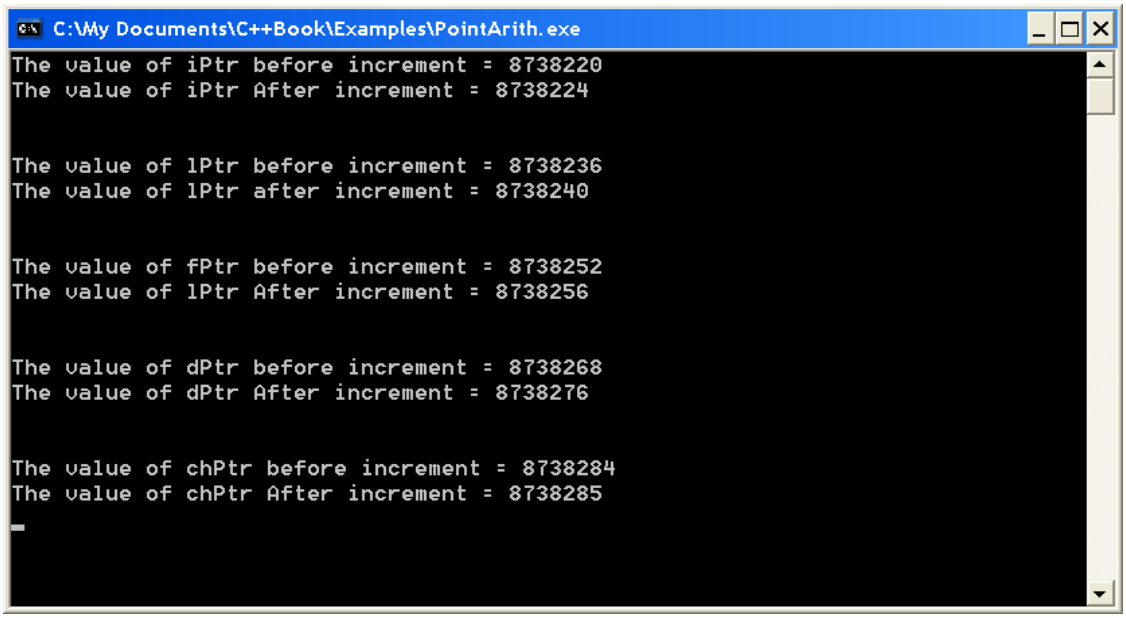
البرنامج أدناه يوضح أثر زيادة قيمة المؤشر بواحد لكل الأنواع المعروفة. بتفحص شاشة نتيجة البرنامج، يتضح لك أن القيمة الفعلية للمؤشر تغيرت بمقدار 4 للأعداد الصحيحة و الحقيقية العادية (int, long, float)، بينما تغيرت بمقدار ثمانية للأعداد الحقيقة المضاعفة (double)، وواحد للمتغيرات الحرفية (char).



مثال (4):

```
#include <iostream.h>
int main(int argc, char* argv[])
{ int x;
int *iPtr;  long *lPtr;
float *fPtr;  double *dPtr;
char *chPtr;
//The effect of incrementin pointers
iPtr = new int;  lPtr = new long;
fPtr = new float;  dPtr = new double;
chPtr = new char;
cout << "The value of iPtr before increment = "<<(long)iPtr<<endl;
iPtr = iPtr + 1;
cout << "The value of iPtr After increment = "<<(long)iPtr<<endl;
cout <<endl<<endl;
cout << "The value of lPtr before increment = "<<(long)lPtr<<endl;
lPtr = lPtr + 1;
cout << "The value of lPtr after increment = "<<(long)lPtr<<endl;
cout <<endl<<endl;
cout << "The value of fPtr before increment = "<<(long)fPtr<<endl;
fPtr = fPtr + 1;
cout << "The value of lPtr After increment = "<<(long)fPtr<<endl;
cout <<endl<<endl;
cout << "The value of dPtr before increment = "<<(long)dPtr<<endl;
dPtr = dPtr + 1;
cout << "The value of dPtr After increment = "<<(long)dPtr<<endl;
cout <<endl<<endl;
cout << "The value of chPtr before increment =
"<<(long)chPtr<<endl;
chPtr = chPtr + 1;
cout << "The value of chPtr After increment = "<<(long)chPtr<<endl;
cin >> x;
return 0;
}
```

عينة من تنفيذ البرنامج:



☜ ملحوظة: تغيير الأنواع Type Casting

لاحظ أننا استخدمنا العبارات (long)fPtr, (long)chPtr... في البرنامج أعلاه. الهدف من هذه العبارات هو تغيير المؤشرات إلى أعداد صحيحة ليسهل طبعها. عملية تغيير النوع عملية عامة مسموح بها في معظم لغات البرمجة وتخضع غالباً لقواعد عامة يمكن الرجوع إليها في مراجع المترجمات المستخدمة.

عند تنفيذ هذا البرنامج تظهر النتائج التالية على الشاشة:

أسئلة تقويم ذاتي



باعتبار التعريف:

```
struct UnitX
{ int x;    double y; char ch; } *MyVar;
```

وبافتراض العمليات التالية داخل البرنامج:

```
MyVar = new UnitX;
MyVar = MyVar + 1;
```

ماهي الزيادة التي تطرأ على قيمة المؤشر MyVar؟ (عدد البايتات).

4. المؤشرات والمصفوفات

هناك علاقة وثيقة بين المصفوفات و المؤشرات في سي++، ذلك أن إسم المصفوفة (المجرد) يعتبر مؤشراً إلى بدايتها. مثلاً، إذا قمنا بتعريف مصفوفة xArray و مؤشر xPtr، فبمكن استخدام xPtr لاستعراض عناصر المصفوفة كما يلي:

```
flaot xArray[10], *xPtr;
xPtr = xArray;  //xPtr now points to the beginning of the array
for (i=0, i<10; i++)
{ cin >> *xPtr;
xPtr = xPtr + 1;
}
for (int i=0; i<10; i++)
cout <<i<<" "<<xArray[i]<<endl;
```

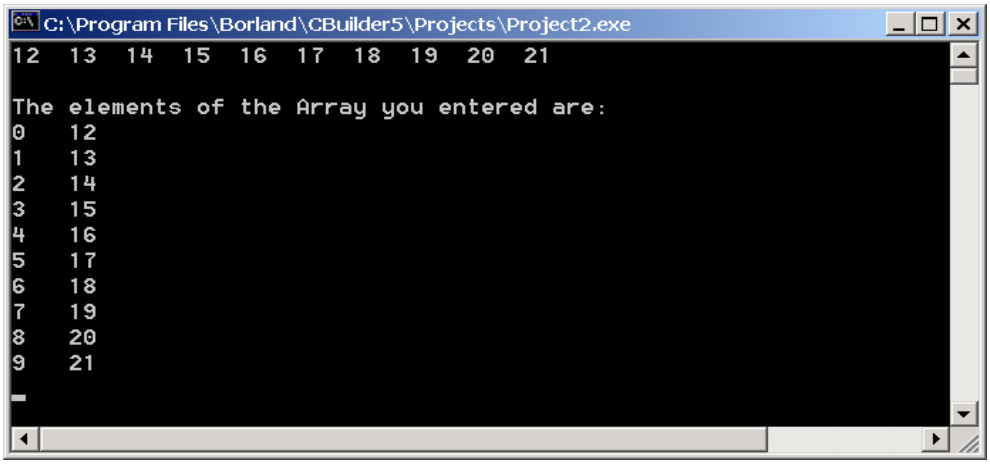
☜ لاحظ أننا استخدمنا التعبير xPtr للإنتقال من عنصر إلى آخر في المصفوفة، ثم طبعنا قيم المصفوفة باستخدام الترقيم المباشر xArray[i]. البرنامج معطى بكامله أدناه:



مثال (5)

```
int main(int argc, char* argv[])
{float xArray[10], *xPtr;
xPtr = xArray;  //xPtr now points to the beginning of the array
for (int i=0; i<10; i++)
{ cin >> *xPtr;
xPtr = xPtr + 1;
}
cout << endl<<"The elements of the Array you entered are:\n";
for (int i=0; i<10; i++)
cout <<i<<"  "<<xArray[i]<<endl;
cin >> *xPtr;
return 0;
}
```

عينة من تنفيذ البرنامج:



مثال (6):

```
int main(int argc, char* argv[])
{ int b[] = { 10, 20, 30, 40 }, i, offset;
int *bPtr = b;   // set bPtr to point to array b

cout << "Array b printed with:\n"
<< "Array subscript notation\n";

for ( i = 0; i < 4; i++ )
cout << "b[" << i << "] = " << b[ i ] << '\n';

cout << "\nPointer/offset notation where\n"
<< "the pointer is the array name\n";

for ( offset = 0; offset < 4; offset++ )
cout << "*(b + " << offset << ") = "
<< *( b + offset ) << '\n';

cout << "\nPointer subscript notation\n";

for ( i = 0; i < 4; i++ )
cout << "bPtr[" << i << "] = " << bPtr[ i ] << '\n';

cout << "\nPointer/offset notation\n";

for ( offset = 0; offset < 4; offset++ )
cout << "*(bPtr + " << offset << ") = "
<< *( bPtr + offset ) << '\n';
cin >> *bPtr;
return 0;
}
```

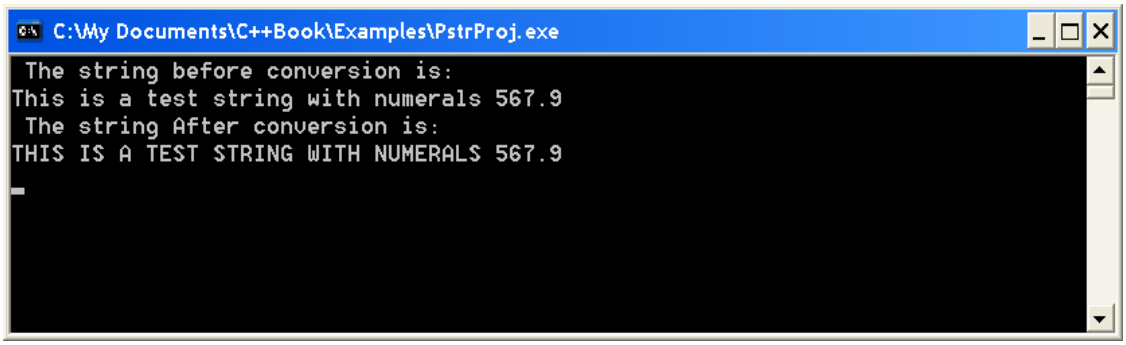
5. المؤشرات وسلاسل الحروف

تعتبر سلاسل الحروف strings حالة خاصة من المصفوفات، ويمكن استخدام المؤشرات في التعامل معها، كما هو واضح من المثال أدناه:

مثال (7):

```
#include <iostream.h>
void ConvertToUpper(char *);
int main(int argc, char* argv[])
{ char TestStr[]="This is a test string with numerals 567.9";
cout << " The string before conversion is:\n"<<TestStr<<endl;
ConvertToUpper(TestStr);
cout << " The string After conversion is:\n"<<TestStr<<endl;
cin >>TestStr;
return 0;
}
void ConvertToUpper(char *sPtr)
{ while (*sPtr != '\0')
{ if (islower(*sPtr))
*sPtr = toupper(*sPtr);
++sPtr;
}
}
```

عينة من تنفيذ البرنامج:



6. استخدام المؤشرات في تخزين القوائم

1.6 القوائم الضائعة

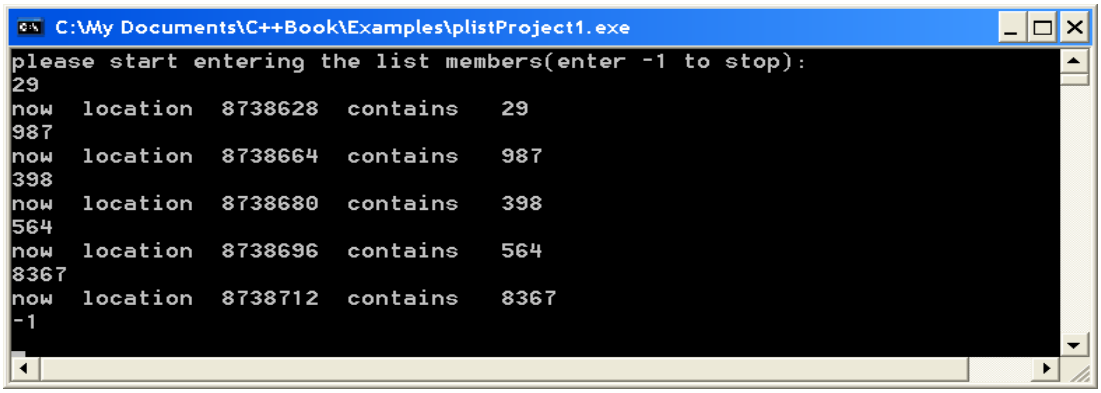
المثال البسيط أدناه يوضح كيف يمكن أن نستخدم مؤشراً لاستحداث وتخزين قائمة متناثرة من الأعداد الصحيحة. بالرغم من أن القائمة يتم تخزينها بسلام (كما تدل على ذلك شاشة متابعة البرنامج أدناه)، إلا أننا نفاجأ بعد الانتهاء من تخزين القائمة أننا لن نستطيع الوصول إلا إلى القيمة الأخيرة. السبب في ذلك أننا استخدمنا مؤشراً واحداً لاستحداث القائمة وكلما استخدمنا الإجراء new أحدثنا عنواناً جديداً في المؤشر وبالتالي مسحنا القيمة السابقة للمؤشر. لعلك الآن فهمت لماذا وصفنا هذه القوائم بأنها "ضائعة".



مثال (8):

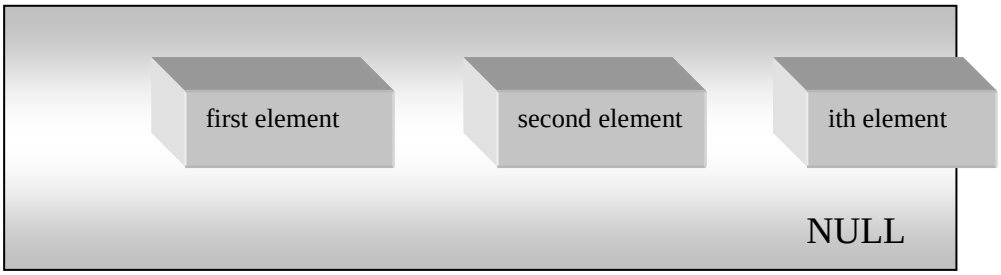
```
#include <iostream.h>
int main(int argc, char* argv[])
{ int *pi,*pj;
int i,x;
cout <<"please start entering the list members(enter -1 to stop):\n";
cin >>x;
while (x>=0)
{  pi = new int;
*pi=x;
cout <<"now  location  "<< (long)pi << "  contains   "<<x<<endl;
cin >> x;
}
return 0;
}
```

عينة من تنفيذ البرنامج:



2.6 القوائم الديناميكية المترابطة

مشكلة "القوائم الضائعة" أعلاه يمكن حلها باستخدام فكرة القوائم الديناميكية المترابطة. تبنى هذه القوائم بحيث يشير كل عنصر إلى سابقه في القائمة. العنصر الأول في القائمة لا سابق له، لذا فإن يشير إلى "لاشيء". في لغة C++ هذا ال"لاشيء" يعبر عنه بالقيمة "NULL". كل عنصر بعد العنصر الأول يحتفظ بمؤشر إلى قيمة سابقه بالإضافة إلى قيمته. بالطبع في هذه الحالة لابد من استخدام السجلات في تمثيل القائمة لأن عناصر القائمة صارت مركبة (قيمة العنصر بالإضافة إلى مؤشر يدل على العنصر السابق):



أ- تعريف القوائم المترابطة في لغة C++

تعرف القائمة المترابطة في سي++ باستخدام مؤشر إلى سجل. السجل يحتوى على قسمين: قسم البيانات وقسم الربط. قسم البيانات يحتوى على قيمة أو قيم العنصر بينما يحتوى

قسم الربط على مؤشر يشير إلى العنصر التالي في القائمة. مثلاً بالنسبة لقائمة الأعداد الصحيحة التي حاولنا معالجتها في المثال أعلاه تعرّف القائمة المترابطة على النحو التالي:

أولاً: نقوم بتعريف سجل يمثل عناصر القائمة. كل عنصر في القائمة يمكن تمثيله كما يلي:

```
struct NODE {
 NODE *Next;
 int Value;
};
```

☜ لاحظ أن شعار السجل NODE استخدم في تعريف الحقل Next و الذي يمثل قسم الربط في هذه القائمة. هذا النوع من التعريف الذاتي مسموح به في سي++ ومعظم لغات البرمجة المنسقة.

ثانياً: يتم تعريف القائمة الفعلية داخل البرنامج باستخدام مؤشر على النحو التالي:

```
NODE *ListP;
```

عناصر القائمة يتم التعامل معها داخل البرنامج بذكر المؤشر ListP و الحقل المشار إليه مباشرةً دون ذكر اسم السجل:

مثلا: `cin >>ListP->value;`

أو `ListP->value =x;`

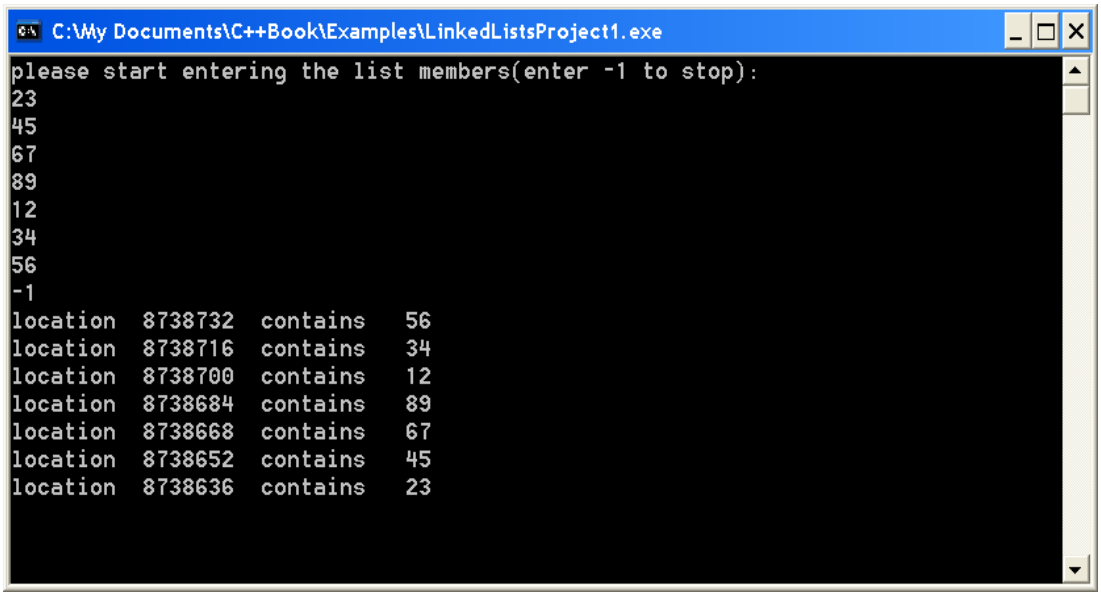
أو `ListP->Next=Prev;`

البرنامج التالي يوضح كيف تبنى القائمة المترابطة ثم كيف تطبع بعد ذلك:

مثال (9):

```
#include <iostream.h>
struct NODE {
NODE *Next;
int Value;
};
int main(int argc, char* argv[])
{  NODE *ListP, *Prev, *CurrentP;
int i,x;
cout << "please start entering the list members(enter -1 to stop):\n";
//  {ÈäÇÁ ÇáÞÇÆãÉ ÇáãÊÑÇÈØÉ}
cin >>x; Prev=NULL;
while (x>=0)
{ ListP =  new NODE;
ListP->Value=x;
ListP->Next=Prev;
Prev=ListP;
cin >> x;
}
// {ØÈÇÚÉ ÇáÞÇÆãÉ ÇáãÊÑÇÈØÉ}
//  {now print out the list}
CurrentP=ListP;
while (CurrentP!=NULL)
{ cout << "location  "<<(long)CurrentP<<" contains  "<<
CurrentP->Value<<endl;
CurrentP=CurrentP->Next;
}
cin >>x;
return 0;
}
```

الشاشة التالية تعطى تسجيلاً لمسار تنفيذ البرنامج أعلاه:



☜ لاحظ أن عملية الطباعة تمت بصورة منفصلة بعد تخزين القائمة بكاملها.

لعل الدارس لاحظ هذه المرة أننا لم نستخدم الأسلوب النقطي المعهود في الوصول إلى عناصر السجل، واستخدمنا سهماً أفقياً، كما هو الحال في االجمل التالية:

```
ListP->Value=x;
ListP->Next=Prev;
```

السبب في ذلك أن المتغير ListP مؤشرٌ إلى السجل و ليس اسماً له. القاعدة العامة في لغة C++ هي استخدام السهم الأفقي للوصول إلى عناصر السجل في حالة استخدام المؤشرات.

ب- البحث عن عنصر في القائمة المترابطة

يمكن البحث عن عنصر في القائمة المترابطة بصورة خطية ابتدءاً بالعنصر الأول. نقطة البداية في هذه الحالة هو المؤشر على راس القائمة. نقارن العنصر المطلوب بالقيمة المخزنة في المكان الذي يشير إليه هذا المؤشر. إذا تمت المساواة توقفنا وإلا حركنا المؤشر ليصبح:

```
p = p->next;
```

تستمر هذه العملية حتى نجد العنصر الذي نبحث عنه أو نصل إلى نهاية القائمة p=NULL.

ج- حذف عنصر من القائمة المترابطة

عملية حذف عنصر من القائمة المترابطة في غاية البساطة وتتلخص في تخطي العنصر المراد حذفه. مثلاً إذا كان العنصر المراد حذفه يشار إليه بالمؤشر q والعنصر الذي قبله يشار إليه بالمؤشر p فإن عملية الحذف تتم كما يلي:

```
p->next = q->next;
```

✍ تدريبات على المؤشرات

باعتبار التعريفات التالية:

```
int  x;
int  *k1, *k2;
float *r1,*r2;
```

ما هو الصواب وما هو الخطأ من الجمل التالية، وفي حالة الصواب ما هو أثر الجملة؟

```
(a)      cout <<k1;
(b)      cin >> *k1;
(c)      k1=k2;
(d)      New(x);
(e)      if   (*k1= = NULL)  k1=k2;
(f)      { *k1= 17;
         k1 = new int;
         }
```

باعتبار التعريف التالي:

```
struct NumNode
{  int value;
NumNode *next;
};
int main()
{  NumNoe  *node1, *node2;
int *intp;
```

وبافتراض أننا نفذنا العمليتين:

```
node1 = new NumNode;
node2 = new NumNode;
```

ما هو أثر العمليات التالية:

(a)
```
node1->value = 456;
node2->value = 678;
node1->next = node2;
cout << node1->value<<endl;
cout << node2->next->value<<endl;
```

(b)
```
*intp= 78;
node1->value = *intp;
node2->value = 493;
node1->next = node2;
node2->next = node1;
*intp= node1->next->next->value;
cout << node2->next->value<<endl;
cout << *intp;
```

اكتب برنامجاً يبني قائمة ديناميكية مترابطة لسجلات مجموعة من السيارات التي تخص معرضاً معيناً. أفرض أن السجلات مخزنة فى ملف معين والمعلومات المخزنة في كل سجل تشمل الآتي:

{ idرقم الشاسيه}
type
model
year

اكتب دالة تحسب عدد العناصر في قائمة مترابطة واستخدمها لحساب عدد العناصر في القائمة التي بنيتها في البرنامج السابق.

اكتب دالة تحدد إذا ما كانت العناصر المخزنة في قائمة مترابطة مرتبة تصاعدياً أم لا. اختبر الدالة باستخدام أحد البرامج السابقة.

اكتب دالة تعكس ترتيب العناصر في قائمة مترابطة.

الخلاصة

في هذه الوحدة المهمة تعرفنا المؤشرات وطريقة عملها وعلاقتها بالمصفوفات وكذلك تعرفنا القوائم وأنواعها وكيفية استخدام المؤشرات في تخزين هذه القوائم وحذف واضافة عنصر لهذه القوائم.

وتعتبر هذه الوحدة أساس نظم ادارة قواعد البيانات التي تعتمد عليها الثورة المعلوماتية في عصرنا الحديث.

لمحة مسبقة عن الوحدة التالية

ستكون الوحدة القادمة عن أساليب البحث والترتيب ومن أساليب الترتيب سنتعرف على خوارزمية الترتيب الفقاعي والترتيب بالاختيار.

أما أساليب البحث سنتعرف على خوارزمية البحث الخطي وخوارزمية البحث الثنائي.

مسرد المصطلحات

المؤشرات Pointers:

هي عبارة عن نوع خاص من المتغيرات لا يحتوى على قيمة ولكن يحتوى على عنوان القيمة في الذاكرة (أي يشير اليها).

تغيير الأنواع Type Casting:

هو تغيير المؤشرات الى أعداد صحيحة ليسهل طبعها.

سلاسل الحروف String

عبارة عن حالة خاصة عن المصفوفات تحمل حروف القوائم المترابطة في لغة C++.

تعّرف باستخدام المؤشر نحو السجل حيث يحتوى السجل على قسمين:

1) قسم البيانات.
2) قسم الربط.

المصادر والمراجع

1) Deictl, H. & Deitel, P., **C++ How to Program.** Third Edition, Prentice Hall: 2001.
2) Savitch, W. **Problem Solving With C++: The Object of Programming**. Third Edition, Addison Wesley: 1999.
3) Friedman, F. and Koffman, E. **Problem Solving, Abstraction, and Design Using C++** (2 nd Edition). Addison Wesley: 1996.
4) Lippman, S. lajoie, J. **C++ Primer,** (3rd Edition). Addison Wesley: 1998.
5) Horstmann, C. **Computing Concepts With C++ Essentials.** John Wiley: 1997.
6) Stroustrup, B. **The C++ Programming Language.** (3 rd edition), Addison Wesley: 1990.
7) Ellis, M. and Bjarnc Stroupstrup, **The Annotated Reference Manual.** Addison Wesley: 1990.
8) Tcale, S. C++ IOS treams Handbook. Addison Wesley: 1993.
9) Meyers, S. **Effective C++: 50 Specific Ways to Improve Your Programs and Designs** (2 nd edition). Addison Wesley: 1996.
10) Meyers, S. **More Effective C++: 35 New Ways to Improve Your Programs and Designs.** Addison Wesley. 1996.
11) Schildt, H**. C++ From the Ground up.** Mc Graw Hill. 2003.
12) Savitch, W. **Absolute C++.** Addison Wesley: 2002.
13) Cogswell, J. **C++ all – in – one Desk Refcrence for Dummies.** 2002.


الوحدة التاسعة
خوارزميات البحث والترتيب

محتويات الوحدة

الموضوع	الصفحة
المقدمة	**204**
تمهيد	204
أهداف الوحدة	204
1. خوارزمية الترتيب الفقاعي	**205**
2. خوارزمية الترتيب بالاختيار	**209**
3. خوارزمية البحث الخطي	**212**
4. خوارزمية البحث الثنائي	**213**
الخلاصة	**215**
مسرد المصطلحات	**216**
المصادر والمراجع	**217**

المقدمة

تمهيد

مرحباً بك عزيزي الدارس

البحث و الترتيب عمليتان أساسيتان يمارسهما الحاسوب في الكثير من التطبيقات ذات المساس بحياتنا اليومية، ومن الطبيعي أن يبذل المختصون جهداً كبيراً في تفعيل هاتين العمليتين و تقليل الزمن الذي يستغرقانه. الموضوع، وإن كان كبيراً، لابد من التطرق له في مقررات البرمجة الأساسية. في هذا الفصل نعرض بعض خوارزميات الترتيب في أبسط صورها ثم بعد ذلك نتطرق لبعض خوارزميات البحث.

أهداف الوحدة

عزيزي الدارس،

بعد فراغك من دراسة هذه الوحدة ينبغى أن تكون قادراً على أن:

- **تصمم** خوارزمية الترتيب الفقاعي والترتيب بالاختيار.
- **تصمم** خوارزمية البحث الخطي والبحث الثقافي.
- **تميز** من أساليب الترتيب المختلفة وأساليب البحث المختلفة.

1. خوارزمية الترتيب الفقاعي Bubble Sort

سميت هذه الخوارزمية بخوارزمية الترتيب الفقاعي لأن العناصر غير المرتبة تتصاعد إلى مواقعها مثل فقاعات الهواء في الماء. يمكن تلخيص هذه الخوارزمية كما يلى:

1- إقرأ المصفوفة المراد ترتيب عناصرها.

2- كرر:

1) أفرض أن المصفوفة مرتبة sorted=true.
2) كرر لكل عنصر من عناصر المصفوفة:
3) إذا كان العنصر الحالي أكبر من الذي يليه:

i- بدل قيمتي العنصرين.

ii- غيِّر الافتراض الأول الذي افترضناه في الخطوة i- أعلاه sorted=false.

3- حتى sorted=true.

المثال العددي أدناه يوضح كيف تعمل الخوارزمية:

افرض أن المصفوفة المراد ترتيبها هي:

7 3 22 97 1 2 5 27 14 92

الجولة الأولى:

في البداية sorted=true

تطبيق الخطوة 2 من الخوارزمية أعلاه ينتج عنه التبديلات أدناه(التي تحتها خط):

k=1	7	3	14	92	27	5	2	1	97	22
k=2	7	3	14	27	92	5	2	1	97	22
k=3	7	3	14	27	5	92	2	1	97	22
k=4	7	3	14	27	5	2	92	1	97	22
k=5	7	3	14	27	5	2	1	92	97	22
k=6	7	3	14	27	5	2	1	92	97	22
k=7	7	3	14	27	5	2	1	92	22	97
k=8	7	3	14	27	5	2	1	92	22	97
k=9	7	3	14	27	5	2	1	92	22	97

عند نهاية هذه الجولة نكتشف أن المتغير sorted = flase و عليه نبدأ جولة جديدة:

الجولة الثانية:

نفترض مجدداً أن sorted=true ثم نطبق الخطوة 2 أعلاه مرة أخرى:

بعد تطبيق الخطوة 2 للمرة الثانية تصبح المصفوفة:

97 92 7 3 22 27 1 2 5 14 k=9

عند نهاية هذه الجولة نجد أن sorted = false وعليه نبدأ جولة أخرى جديدة.

الجولة الثالثة:

sorted:=true

نطبق الخطوة 2 للمرة الثالثة:

عند نهاية هذه الجولة تصبح المصفوفة:

97 92 27 7 3 22 14 1 2 5 k=9

sorted = false

وعليه نبدأ جولة جديدة:

الجولة الرابعة:

sorted := true

نطبق الخطوة 2 للمرة الرابعة ليصبح الناتج:

97 92 27 22 7 3 14 5 1 2 k=9

sorted=false

وعليه لابد من خطوة أخرى:

الجولة الخامسة:

sorted:=true

نطبق الخطوة 2 للمرة الخامسة ليصبح الناتج:

97 92 27 22 14 7 3 5 2 1 k=9

sorted=false

وعليه لابد من جولة أخرى:

sorted:=true

نطبق الخطوة 2 لنحصل في نهايتها على:

k=9 1 2 3 5 7 14 22 27 92 97

sorted=false

بما أن sorted مازال false فلابد من إجراء جولة أخرى توكيدية:

الجولة السادسة:

نطبق الخطوة 2 لنحصل في نهايتها على:

k=9 1 2 3 5 7 14 22 27 92 97

sorted=true

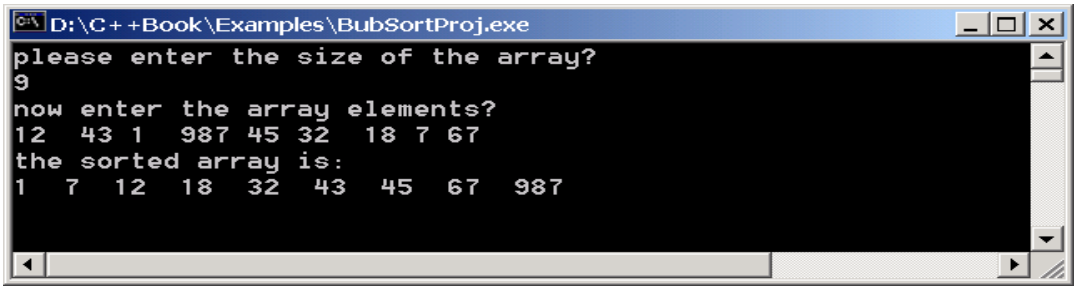
بما أنه لم يحدث تغيير في المصفوفة هذه المرة فإن sorted ظل محتفظاً بقيمته true وعليه فلا داعي لجولة أخرى وبكون هذا بمثابة إشعار بأن المصفوفة تم ترتيبها تصاعدياً.

البرنامج:

مثال (1)

```
include <iostream.h>#
void Exchange(float &a, float &b);
int main(int argc, char* argv[])
{  const Limit=100;
float x[Limit];
int k, size;
bool sorted;
cout <<"please enter the size of the array?\n";
cin >> size;
cout <<"now enter the array elements?\n";
for (k=0; k<size; k++)
cin >> x[k];
//now start sorting
do
{ sorted=true;
for (k=0; k<size-1; k++)
if (x[k]>x[k+1])
{ Exchange(x[k], x[k+1]);
sorted=false;
}
} while (!sorted);
//now print the sorted array
cout <<"the sorted array is:\n";
for (k=0; k<size; k++)
cout <<x[k]<<"  ";
return 0;
}
void Exchange(float &a, float &b)
{ float temp;
temp=a;
a=b;
b=temp;
}
```

عينة من تنفيذ البرنامج:



أسئلة تقويم ذاتي



استخدم خوارزمية االترتيب الفقاعي في ترتيب القائمة التالية يدوياً:
12, 1, 8, 19, 7
احسب عدد الدورات التي استغرقها الترتيب.

2. خوارزمية الترتيب بالاختيار

1) إقرأ المصفوفة المراد ترتيبها x وأفرض أن حجمها n.
2) اجعل المتغير k يشير إلى بداية المصفوفة k=0
3) كرر: من k=1 حتى k=n-1

1- ابحث عن أصغر عنصر في المصفوفة smallest ابتداءً من الموقع k.

2- بدل هذا العنصر مع العنصر الأول في المصفوفة:

إذا كان موقع smallest هو j:

```
x[j]:=x[k];
x[k]:=smallest
```

الخطوة 3-i تبحث عن أصغر عنصر في المصفوفة من الموقع k إلى نهاية المصفوفة وقد تعرضنا لكيفية البحث عن أصغر عنصر في الفصول السابقة.

المثال العددي أدناه يشرح الخوارزمية أعلاه:

افرض أن المصفوفة المراد ترتيبها هي:

92 14 27 5 2 1 97 22 3 7

الخطوة الأولى:

k=1

أصغر عنصر في الجزء من المصفوفة الذي يبدأ بالعنصر الأول(ابتداءً من k=1) هو 1.

نبدل هذا العنصر بالعنصر الأول(92) لتصبح المصفوفة:

1 14 27 5 2 92 97 22 3 7

الخطوة الثانية:

k=2

اصغر عنصر في الجزء من المصفوفة الذي يلي العنصر الأول هو 2. نبدل هذا العنصر بالعنصر الثاني (14) لنحصل على المصفوفة:

1 2 27 5 14 92 97 22 3 7

الخطوة الثالثة:

في هذه الخطوة نأخذ في الاعتبار الجزء من المصفوفة الذي يلي العنصر الثاني (k=3) ونكتشف أن أصغر عنصر في هذا الجزء هو 3. نبدل هذا بالعنصر الثالث (27) لنحصل على:

1 2 3 5 14 92 97 22 27 7

الخطوة الرابعة:

نكرر نفس العملية في الخطوات السابقة باعتبار k=4 لنحصل فنكتشف أن العنصر الأصغر في هذا الجزء هو العنصر الرابع نفسه، لذا لا داعي للتبديل و تبقى المصفوفة كما في الخطوة السابقة:

1 2 3 5 14 92 97 22 27 7

الخطوة الخامسة (k=5):

بتكرار نفس العمل في الخطوات السابقة نجد أن العنصر الأصغر في هذا الجزء هو العنصر 7.

نبدل هذا مع العنصر الخامس لتحصل على المصفوفة:

1 2 3 5 7 92 97 22 27 14

الخطوة السادسة (k=6) تقود إلى المصفوفة:

1 2 3 5 7 14 97 22 27 92

الخطوة السابعة (k=7) تقود إلى المصفوفة:

1 2 3 5 7 14 22 97 27 92

الخطوة الثامنة (k=8) تقود إلى المصفوفة:

1 2 3 5 7 14 22 27 97 92

الخطوة التاسعة(k=9) يتم فيها آخر تبديل لنحصل على المصفوفة المرتبة:

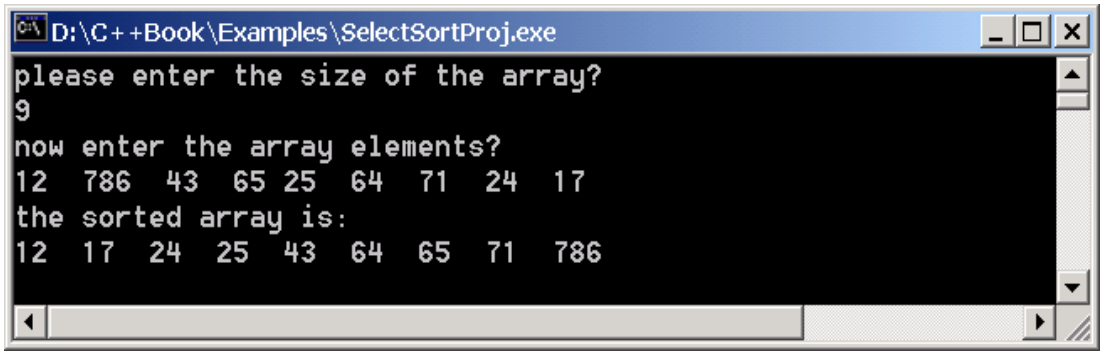
1 2 3 5 7 14 22 27 92 97

البرنامج:

(مثال 10)

```
#include <iostream.h>
int main(int argc, char* argv[])
{  const Limit=100;
float x[Limit],smallest;
int k,i,j,Spos, size;
bool sorted;
cout <<"please enter the size of the array?\n";
cin >> size;
cout <<"now enter the array elements?\n";
for (k=0; k<size; k++)
cin >> x[k];
//now start sorting
for (i=0; i<size-1; i++)
{ smallest=x[i]; Spos=i;
for (j=i; j<size; j++)
if  (x[j]<smallest)
{ smallest=x[j];
Spos=j;
}
x[Spos]=x[i];
x[i]=smallest;
}//for i
//now print the sorted array
cout <<"the sorted array is:\n";
for (k=0; k<size; k++)
cout <<x[k]<<"  ";
return 0;
}
```

عينة من تنفيذ البرنامج:



3. خوارزمية البحث الخطي

لقد استخدمنا خوارزمية البحث الخطى بصورة غير مباشرة في عدة أمثلة. وكما يدل الاسم فإن البحث عن العنصر المطلوب بصورة خطية من بداية القائمة إلى نهايتها ويتوقف البحث عند وجود العنصر أو الوصول إلى نهاية القائمة. الخوارزمية يمكن تلخيصها في الإجراء التالي:

```
void  LinearSearch(float x[], int size, float  ItemToFind, bool  &found,
                                                        int &Location)
                                                              {  int k;
                                                    found=false;   =0;
                                          while ((k<size) and (not found))
                                                                    {
                                                               k=k+1;
                                                 if (x[k] == ItemToFind )
                                                                    {
                                                             found=true;
                                                            Location=k;
                                                                    }
                                                                    }
                                                                    }
```

في هذا الإجراء x هي القائمة التي سنبحث فيها عن العنصر المطلوب ItemToFind.

4. خوارزمية البحث الثنائي

تستخدم خوارزمية البحث الخطى إذا كانت القائمة x التي نبحث فيها غير مرتبة. أما إذا كانت القائمة مرتبة فمن الأمثل استخدام خوارزمية البحث الثنائي التي تتبع أسلوباً تنصيفياً في الوصول إلى العنصر المطلوب ويمكن تلخيصها كما يلي:

1- أفرض أن أو القائمة هو First وأن آخرها هو Last و found=false
2- حدد وسط القائمة Middle=(First +last) DIV 2.
3- حدد إذا ما كان العنصر المطلوب في النصف الأسفل أو النصف الأعلى من القائمة.
4- غيِّر قيمة First و Last ليشيرا إلى بداية و نهاية هذا النصف.
5- كرر الخطوتين 2 – 4 حتى تصل إلى مرحلة يتساوى فيها العنصر المطلوب مع وسط القائمة أو يستحيل التنصيف مرةً أخرى. إذا تساوى العنصر المطلوب مع محتوى وسط القائمة اجعل found:=true وإلا يكون found:=false.

الخوارزمية يمكن تلخيصها في الدالة التالية:

```
void BinarySearch(float x[], int size, float ItemToFind, bool
&found, int &Location)
{ int First, Last, Middle;
First=1; Last=size;
found=false;
while ((First < Last) && (!found)) do
{
Middle = (First + Last) / 2;
if (ItemToFind < x[middle])
Last = Middle –1;
else (if ItemToFind > x[middle])
First= Middle + 1;
else
{
found=true;
Location=Middle;
}
}
}//end function
```

✍ تدريبات

1- استخدم خوارزمية الترتيب الفقاعي يدوياً في ترتيب القائمة التالية ترتيباً تصاعدياً:

65 897 3 9 16 89 23

2- احسب عدد مرات التكرار التي استغرقتها العملية حتى تم ترتيب القائمة.

3- أعد عملية الترتيب في السؤال أعلاه باستخدام خوارزمية الترتيب بالإختيار هذه المرة. أي الطريقتين أفضل و لماذا؟

4- حول كل من البرنامجين 10-1 و 10-2 إلى دالة يمكن استدعاؤها من داخل برنامج وجرب أداء الدالتين الناتجتين في برنامج.

5- اكتب برنامجاً يبحث عن كتاب معين ضمن قائمة كتب معطاة. القائمة تحتوي على أسماء الكتب بصورة غير مرتبة.

6- اكتب برنامجاً يرتب قائمة أسماء الكتب المذكورة في السؤال 4 أعلاه ترتيباً تصاعدياً.

7- اكتب برنامجاً يبحث عن سجل طالب معين ضمن قائمة سجلات ويطبع اسمه ومعدله التراكمي. افترض أن قائمة سجلات الطالب مرتبة تصاعدياً حسب الرقم الجامعي. كل سجل يحتوي على الإسم، الرقم، المعدل التراكمي.

الخلاصة

عزيزي الدارس،

تعرفنا في هذه الوحدة أنواع الترتيب، الترتيب الفقاعي والترتيب بالاختيار وكيفية عملها في جولات لانجاز مهامها.

أيضاً تعرفنا طرق البحث الخطي والبحث الثنائي والخطوات التي يتم بها البحث وبذلك تكون قد تعرفت على أساسيات البرمجة في ادارة المعلومات من تخزين وحذف واضافة عنصر كما في الوحدة السابقة وترتيب والبحث عن عنصر كما في هذه الوحدة ونتمنى من المولى أن يكون مقررنا هذا قد ارضي طموحك وعزز معرفتك.

مسرد المصطلحات

الترتيب الفقاعي:

تشبه عملية الترتيب فقاعات الهواء في الماء حيث تتصاعد العناصر غير المرئية الى مواقعها.

البحث الخطي:

يتم البحث عن العنصر المطلوب بصورة خطية من بداية القائمة الى نهايتها ويتوقف البحث عن وجود العنصر أو الوصول الى نهاية القائمة.

البحث الثنائي:

يتم البحث الثنائي باتباع أسلوب تنصيفي في الوصول الى العنصر المطلوب ويجب أن تكون القائمة مرئية حتى يتم البحث الثنائي فيها.

المصادر والمراجع

1) Third Edition, Prentice Hall: 2001.
2) Savitch, W. **Problem Solving With C++: The Object of Programming**. Third Edition, Addison Wesley: 1999.
3) Friedman, F. and Koffman, E. **Problem Solving, Abstraction, and Design Using C++** (2 nd Edition). Addison Wesley: 1996.
4) Lippman, S. lajoie, J. **C++ Primer,** (3rd Edition). Addison Wesley: 1998.
5) Horstmann, C. **Computing Concepts With C++ Essentials.** John Wiley: 1997.
6) Stroustrup, B. **The C++ Programming Language.** (3 rd edition), Addison Wesley: 1990.
7) Ellis, M. and Bjarnc Stroupstrup, **The Annotated Reference Manual.** Addison Wesley: 1990.
8) Tcale, S. C++ IOS treams Handbook. Addison Wesley: 1993.
9) Meyers, S. **Effective C++: 50 Specific Ways to Improve Your Programs and Designs** (2 nd edition). Addison Wesley: 1996.
10) Meyers, S. **More Effective C++: 35 New Ways to Improve Your Programs and Designs.** Addison Wesley. 1996.
11) Schildt, H**. C++ From the Ground up.** Mc Graw Hill. 2003.
12) Savitch, W. **Absolute C++.** Addison Wesley: 2002.
13) Cogswell, J. **C++ all – in – one Desk Refcrence for Dummies.** 2002.

2004 / 234