

Variables, Data Types and Constants

Variable Analogy

- Retain the number 5 in your memory
- Memorize the number 2 at the same time
- You have now stored two different values in your memory
- Add 1 to the 1st number
- Now you should be retaining the numbers $6(5+1)$ and 2 in your memory
- Subtract this two values
- You should obtain $4(6-2)$

- The same process expressed in C++ instruction set looks like

`a = 5;`

`b = 2;`

`c = a + 1;`

`result = c - b;`

- Variable = Portion of memory to which we can store a value and from which we can later retrieve that value.

Variable

- is named storage location in the computer's memory for holding a piece of data



- Provides an “interface” to RAM
- Information stored in variables is actually stored in RAM
- Allow you to store and work with data in the computer's memory
- Named so, because their value(the data they contain) can change while the program is running

Identifier

- Program defined name that represents some element of a program e.g a variable name
- Should give an indication of what the variables are used for or should reflect their purpose
- Must be easy to guess what they are used for by reading their names
e.g. number (in the previous figure)

Legal(Valid) Identifiers

- The first character must be one of the letters a through z, A through Z, or an underscore character (_).
- After the first character you may use the letters a through z or A through Z, the digits 0 through 9, or underscores.
- Uppercase and lowercase characters are distinct. This means `ItemsOrdered` is not the same as `itemsordered`.

Which are legal identifiers?

dayOfWeek

Legal

3dGraph

Illegal. Variable names cannot begin with a digit.

_employee_num

Legal

June1997

Legal

Mixture#3

Illegal. Variable names may only use letters, digits and underscores

Valid Identifiers

- White spaces are not allowed in identifiers
- But can contain mix of upper case and lower case letters
- The convention is to capitalize the first letter of each subsequent word in identifier leaving all others in small letters

e.g. itemsOrdered

- Variable names are case sensitive

e.g. itemsOrdered is not the same as
itemsordered

Valid Identifiers

- Another rule is that, when inventing your own identifiers they can not match any keyword of the C++ language nor your compiler specific ones which are reserved keywords

asm	auto	break	bool	case
catch	char	class	const	const_cast
continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern
false	float	for	friend	goto
if	inline	int	long	mutable
namespace	new	operator	private	protected
public	register	reinterpret_cast	return	short
signed	sizeof	static	static_cast	struct
switch	template	this	throw	true
try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile
wchar_t	while			

Data types

- In programming, there are generally two types of data numbers , such as 3, and characters such as the letter 'A'
- Numbers are used to perform mathematical operations while
- Characters are used to print information to the screen
- In programming, the type of data that will be stored in the variable must be known

Data Types

- Numerical data types can be either integer or whole numbers or real or floating point numbers

E.g. Integers or whole numbers

5, 7, -129 and 32154

And real or floating point numbers

e.g. 3.14159, 6.7, 1.0002

Data Types

- The computer has to know what kind of data we want to store in them because it doesn't occupy the same amount of memory to store a simple number than to store a single letter or a large number
- Memory in computers is organized in bytes
- Byte
 - Minimum amount of memory that we can manage in C++
 - Can store relatively small amount of data i.e. one single character or a small integer (generally an integer between 0 and 255)

In addition, the computer can manipulate more complex data types that come from grouping several bytes, such as long numbers or non-integer data types

Data Types

- Signed types can represent both positive and negative values, whereas unsigned types can only represent positive values (and zero)

Summary of fundamental data types in C++

Name	Description	Size*	Range*
char	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
bool	Boolean value. It can take one of two values: true or false.	1byte	true or false
float	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)

Considerations for selecting numeric data type

- The largest and smallest numbers that may be stored in the variable
- Whether the variable stores signed numbers
- The number of decimal places of precision the variable has
- How much memory the variable uses

Guarantees

Size of integer is dependent on the type of system you are using but these are guaranteed to be satisfied:

- Integers are at least as big as short integers.
- Long integers are at least as big as integers.
- Unsigned short integers are the same size as short integers.
- Unsigned integers are the same size as integers.
- Unsigned long integers are the same size as long integers.

Char Data Type

- One byte integer data type
- Char variable can hold a single character, but strictly speaking it is an integer data type
- ASCII is commonly used for encoding characters

e.g. `cout<<65` is the same as
`cout<<'A'`

Because the character 'A' is encoded using the number 65

Strings

Allow a series of characters to be stored in consecutive memory locations

Can be virtually any length

The program knows how long a string is by appending an extra byte to the end of string constants and

In this byte the number 0 is stored called null terminator or null character

e.g.

S	e	b	a	s	t	i	a	n	\0
---	---	---	---	---	---	---	---	---	----

Variable Declaration

- Tells the compiler the variable's name and the type of data it will hold
- The syntax for variable declaration is to write the specifier of the desired data type (like int, bool, float...) followed by a valid variable identifier

e.g. `int a;`

`float myNumber;`

- Once declared the variables can be used within the rest of the program(must be declared before we use them)

Variable declaration

- To declare more than one variable of the same type, it can be done in a single statement by separating their identifiers with commas

e.g.

```
1 int a;  
2 int b;  
3 int c;
```

 Can be written as

```
int a, b, c;
```

Assignment(=)

number = 5;

- The = operator copies the variable on its right to the variable names on its left
- The variable name must always appear on the left
- Which of the following are legal C++ assignment statements?

a = 7;

legal

7 = a;

Illegal, because the variable name must be on the left

7 = 7;

Illegal, because the left hand signed of the assignment operator is not a variable

Initialization of Variables

- When declaring a variable, its value is initially undetermined
- Makes a variable to store a concrete value at the same moment that it is

```
1 // initialization of variables
2
3 #include <iostream>
4 using namespace std;
5
6 int main ()
7 {
8     int a=5;           // initial value = 5
9     int b=2;           // initial value = 2
10    int result;         // initial value undetermined
11
12    a = a + 3;
13    result = a - b;
14    cout << result;
15
16    return 0;
17 }
```

Constant

- Unlike a variable, it is a data item whose value cannot change during the program's execution
- Includes literals, defined constants and declared constants

Literals

Are Used to express particular values within the source code of a program

e.G `a = 5;`

5 is a literal constant

Literals like variables are considered to have a specific data type

Literals can be Integer

Numerals, floating point numbers, character and string literals

Literals

Integer Numerals

- Numerical constants that identify integer decimal values

e.g 75

- C++ allows use of octal(base 8) and hexadecimal(base 16) numbers as string literals

e.g.

```
1 75           // decimal
2 0113         // octal
3 0x4b         // hexadecimal
```

Literals

Integer Literals

- are by default of type `int`
- Can be forced to be unsigned by appending the `u` character to it, or long by appending `l`

e.g. `75` `//int`

`75u` `//unsigned int`

`75l` `//long`

`75ul` `//unsigned long`

Literals

Floating point numbers

- Express numbers with decimals and or exponent

e.g.

```
1 3.14159      // 3.14159
2 6.02e23      // 6.02 x 10^23
3 1.6e-19      // 1.6 x 10^-19
4 3.0          // 3.0
```

- Default value of floating point literals is *double*
- To explicitly express a float or long double numeric literal, you can use the f or l suffixes

E.g

```
1 3.14159L      // long double
2 6.02e23f      // float
```

Literals

Character and string literals

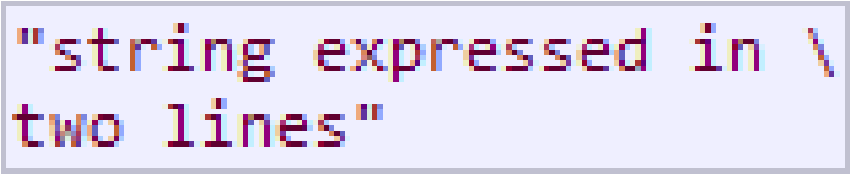
```
1 'z'  
2 'p'  
3 "Hello world"  
4 "How do you do?"
```

- The first two are single character constants and the following two are several characters
- To represent a single character, we enclose it between single quotes(') and to express a string(which generally consists of more than one character), we enclose it in double quotes(“)

Literals

String literals

- can extend to more than a single line of code by putting a backslash sign (\) at the end of each unfinished line

e.g. 

- Can concatenate several string constants by separating them by one or several blank spaces

e.g. "this forms" "a single" "string" "of characters"

Escape codes

- Special characters, that are difficult or impossible to express otherwise in the source code of a program, like

<code>\n</code>	newline
<code>\r</code>	carriage return
<code>\t</code>	tab
<code>\v</code>	vertical tab
<code>\b</code>	backspace
<code>\f</code>	form feed (page feed)
<code>\a</code>	alert (beep)
<code>\'</code>	single quote (')
<code>\"</code>	double quote (")
<code>\?</code>	question mark (?)
<code>\\</code>	backslash (\)

~ tab(`\t`)

Literals

Boolean literals

There are two valid boolean values:
true and false

Can be expressed in C++ as values of
type bool by using the boolean literals
true and false

Defined Constants

- can define your own names for constants that you use very often without having to resort to memory-consuming variables, simply by using the `#define` preprocessor directive

e.g.

```
1 #define PI 3.14159
2 #define NEWLINE '\n'
```

```
#include<iostream>
using namespace std;
#define PI 3.14159
#define NEWLINE '\n'
Int main()
{
    cout<<PI<<NEWLINE;
    return 0;
}
```

- When the compiler preprocessor encounters the `#define` directive, it literally replaces any occurrence of the identifier by the code to which they have been define

Declared Constants(const)

- With const prefix, you can declare constants with a specific type in the same way as you would do with a variable

```
1 const int pathwidth = 100;  
2 const char tabulator = '\t';
```

- The const qualifier specifies that the value of the variable will not change throughout the program
- Why bother with using a variable when we can simply place the literal constant in its place everywhere in the code?

Exercise.

List all the variables and constants that appear in the following program.

```
// This program uses variables and constants
#include <iostream>
using namespace std;
int main()
{
    int little;
    int big;
    little = 2;
    big = 2000;
    cout << "The little number is " << little << endl;
    cout << "The big number is " << big << endl;
    return 0;
}
```