



The Basics of a C++ Program

- A C++ program is a collection of one or more subprograms (functions)
- Function
 - Collection of statements
 - Statements accomplish a task
- Every C++ program has a function called **main**



The Basics of a C++ Program

- Programming language
 - a set of rules, symbols, special words
- Rules
 - syntax – specifies legal instructions
- Symbols
 - special symbols (**+** **-** ***** **!** ...)
- Word symbols
 - reserved words
 - (**int**, **float**, **double**, **char** ...)



Example Program

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Welcome to C++ Programming"<<endl;
    return 0;
}
```

Welcome to C++ Programming

Program Output

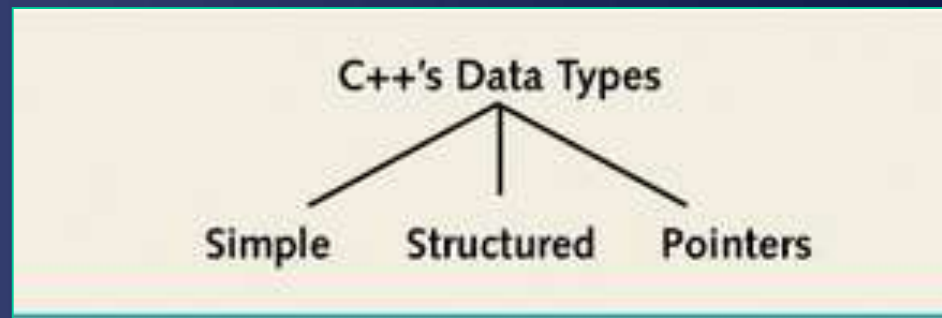


Identifiers

- Rules for identifiers
 - must begin with letter or the underscore _
 - followed by any combination of numerals or letters
 - recommend meaningful identifiers
- Evaluate the following
 - ElectricCharge**
 - 23Skidoo**
 - snarFbLat**

Data Types

- Definition:
 - a set of values
 - combined with a set of operations





Data Types

- Simple data types include
 - Integers
 - Floating point
 - Enumeration
- Integer data types include

char

short

int

long

bool

Numerals, symbols,

Numbers without decimals

Values *true* and *false* only

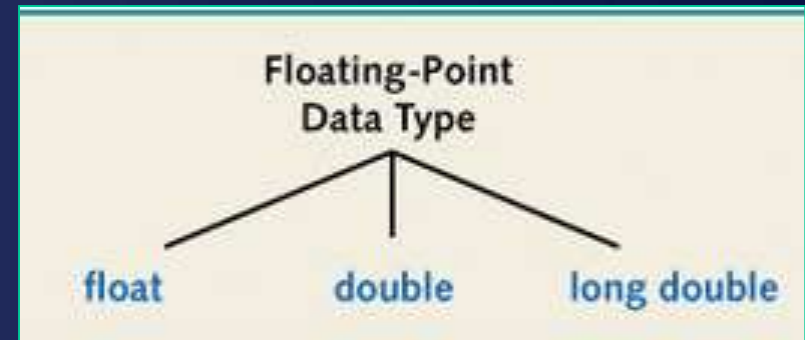
Floating-Point Types

- Stored using scientific notation
 - the sign of the number,
 - the significant digits of the number
 - the sign of the power of 10
 - the power of 10

Real Number	C++ Floating Point-Notation
75.924	7.592400E1
0.18	1.800000E-1
0.0000453	4.530000E-5
-1.482	-1.482000E0
7800.0	7.800000E3

Data Types

- Different floating-point types



- Note that various types will
 - have different ranges of values
 - require different amounts of memory

Data Type	Values	Storage (in bytes)
int	-2147483648 to 2147483647	4
bool	true and false	1
char	-128 to 127	1



Data Types

- The **string** Type
 - a programmer-defined type
 - requires **#include <string>**
- A string is a sequence of characters

"Hi Mom"

"We're Number 1!"

"75607"



Arithmetic Operators and Operator Precedence

10

- Common operators for calculations

$+$ $-$ $*$ $/$ $\%$

- Precedence same as in algebraic usage
 - Inside parentheses done first
 - Next $*$ $/$ $\%$ from left to right
 - Then $+$ and $-$ from left to right
- Note operator precedence chart, page 1035



Expressions

- An expression includes
 - constants
 - variables
 - function calls
 - combined with operators

`3 / 2 + 5.0`

`sin(x) + sqrt(y)`



Expressions

- Expressions can include
 - values all of the same type
 $3 + 5 * 12 - 7$
 - values of different (compatible) types
 $1.23 * 18 / 9.5$
- An operation is evaluated according to the types of the operands
 - if they are the same, the result is the type of the operands
 - if the operands are different (int and float) then the result is float



Type Casting

- Implicit change of type can occur
 - when operands are of different type
- It is possible to explicitly specify that an expression be converted to a different type

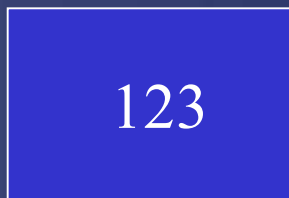
```
static_cast < type > (expression)
```

```
static_cast <int> (3.5 * 6.9 / x)
```



Input

- Storing data in the computer's memory requires two steps
 1. Allocate the memory by declaring a variable
 2. Have the program fetch a value from the input device and place it in the allocated memory location



x

```
cin >> x
```





Allocating Memory

- Variable
 - A memory location whose content may change during program execution
- Declaration:
 - Syntax:
type identifier;
 - Example:
double x;
int y = 45;

Note optional
initialization of the
variable





Allocating Memory

■ Named Constant

- A memory location whose content cannot be changed

■ Declaration

- Syntax:

const type identifier = value;

- Example

const double PI = 3.14159;

Note required
initialization of the
named constant



Putting Data Into Variables

- At initialization time
- Assignment statement
 - Syntax:
variable = expression;
 - Example
x = 1.234;
volume = sqr (base) * height;
- Input (read) statement
 - Syntax:
cin >> variable ;
 - Example
cin >> height;

Program
Example



Increment and Decrement Operators

18

- Pre-increment `++x;`
equivalent to `x = x + 1;`
 - Pre-decrement `--x;`
 - Changes the value before execution of a statement `y = ++x;`
- Post-increment `intVal++;`
 - Post-decrement `intVal--;`
 - Changes the value after execution of the statement `y = x++;`



Output

- Values sent to an output device
 - Usually the screen
 - Can also be a file or some device

- Syntax for screen output:

`cout << expression << ...`

- Example

`cout << "The total is " << sum << endl;`

Output
command

Insertion
operator

Sample
program

Values to
printed

Manipulator
for carriage
return



Output

- Escape sequences also used to manipulate output

```
cout << "The total is\t" << sum << endl;
```

	Escape Sequence	Description
\n	Newline	Cursor moves to the beginning of the next line
\t	Tab	Cursor moves to the next tab stop
\b	Backspace	Cursor moves one space to the left
\r	Return	Cursor moves to the beginning of the current line (not the next line)
\\	Backslash	Backslash is printed
\'	Single quotation	Single quotation mark is printed
\"	Double quotation	Double quotation mark is printed



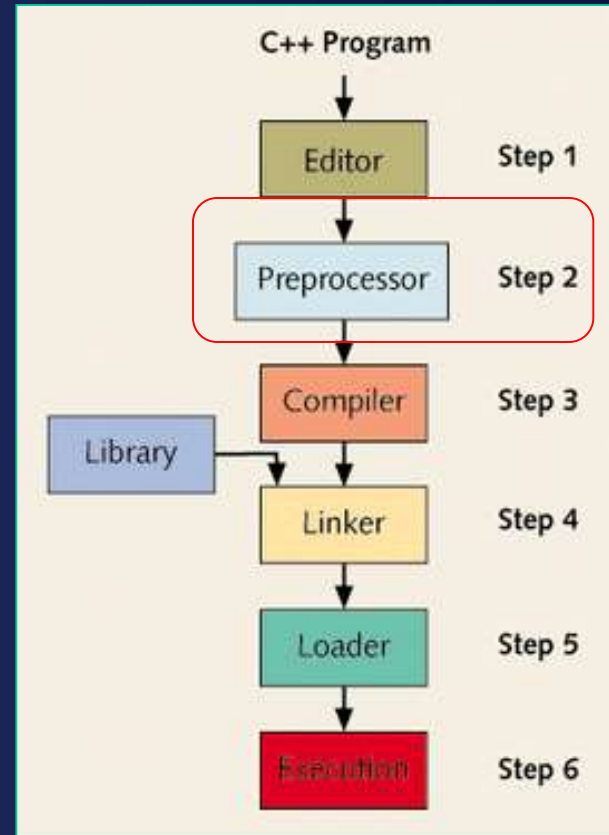
Preprocessor Directives

- Commands supplied to the preprocessor
 - Runs before the compiler
 - Modifies the text of the source code before the compiler starts
- Syntax
 - start with **#** symbol
 - **#include <headerFileName>**
- Example **#include <iostream>**



Preprocessor Directives

- Note the preprocessor step in the sequence





Namespace

- The `#include <iostream>` command is where `cin` and `cout` are declared
- They are declared within a namespace called `std`
- When we specify `using namespace std;`
 - Then we need not preface the `cin` and `cout` commands with `std::cin` and `std::cout`



Program Style and Form

- Every program must contain a function called main

```
int main (void)  
{ ... }
```

- The **int** specifies that it returns an integer value
- The **void** specifies there will be no arguments
- Also can say

```
void main( )  
{ ... }
```



Program Style and Form

- Variables usually declared
 - inside main
 - at beginning of program
- Use blanks and space to make the program easy for humans to read
- Semicolons ; required to end a statement
- Commas used to separate things in a list



Program Style and Form

■ Documentation

- Comments specified between
`/* this is a comment */`
and following `// also a comment`
- Always put at beginning of program
`/* name,`
`date,`
`cpo,`
`purpose of program`
`*/`



Program Style and Form

- Names of identifiers should help document program

```
double electricCharge;  
// instead of ec
```

- Prompt keyboard entry

```
cout << "Enter the value for x -> ";  
cin >> x;
```