PROGRAMMING LANGUAGE C++

PhD course Work: PHYS 601

Presented By
Geeta Ray

Introduction to C++

- C++ is a programming language developed in the 1970's alongside the UNIX operating system.
- C++ provides a comprehensive set of features for handling a wide variety of applications, such as systems development and scientific computation.
- C++ is an “extension” of the C language, in that most C programs are also C++ programs.
- C++, as opposed to C, supports “object-oriented programming.”

C IS ALIVE IN C++

- C++ is a superset of C.
- Any correct C program is also a correct C++ program.

WHAT IS STILL THE SAME

- The syntax of statements
  - if-else, switch, the “?:” conditional
  - for/while/do-while loops
  - assignments
  - arithmetic/logic/relational/ bitwise expressions
  - declarations, structure

- Same preprocessor commands in C and C++.

Convenient syntax for inline comments

- Anything from // to the end of line is considered a comment and thus ignored by the compiler.

- The C-syntax for comments, /* … */, can still be used for multi-line comments.

DECLARATION ANYWHERE

- Declarations need no longer be at the head of blocks.
- Variables and functions can be declared any time, anywhere in a program, preferably as close to where a variable is used the first time.
- For example: i is declared within for

```
for (int i=0;i<n;i++)
```

Simplified IO

- Instead of the complicated syntax of **printf** and **scanf**, and the many variations of **print** and **scan**, C++ offers a much simpler syntax
- For standard output, use **cout**
- For standard input, use **cin**
- File IO is also simpler, and will be discussed later
- Note: one can still use the IO syntax of C in C++

A C++ PROGRAM

include headers:- these are modules that include functions that you may use in your program; we will almost always need to include the header that defines cin and cout; the header is called iostream.h

```
#include <iostream.h>

 main() {

//variable declaration
//read values input from user
//computation  and print output to user
return 0;
}
```

After you write a C++ program you compile it; that is, you run a program called **compiler** that checks whether the program follows the C++ syntax

- if it finds errors, it lists them
- If there are no errors, it translates the C++ program into a program in machine language which you can execute

NOTES

- what follows after // on the same line is considered comment
- all statements end with a semicolon
- **CASE MATTERS X is different from x**
  - Void is different than void.
  - Main is different that main
  - Void and Main are not recognized by C/C++

THE FAMOUS 'HELLO WORLD' PROGRAM

When learning a new language, the first program people usually write is one that salutes the world :)

Here is the Hello world program in C++.

```
#include <iostream.h>
 main() {
   cout << “Hello world!”;

    return 0;
}
```

Hello World Program

- How to compile?

  $ g++ hello.cpp

g++	compiling command
hello.cpp	source file
./a.out	executable file

Note: the default output filename is "a.out"

```
2 // A first program in
3 #include <iostream>
4
5 main()
6 {
7     std::cout << "Welcom
8 std::cout << "to C++!\n
9     return 0;          //
10 }
```

```
Welcome to C++!
```

Comments
Written between **/*** and ***/** or following a **//**.
Improve program readability and do not cause the computer to perform any action.

preprocessor directive
Message to the C++ preprocessor.
Lines beginning with **#** are preprocessor directives.
#include <iostream> tells the preprocessor to

C++ programs contain one or more functions, one of which must be **main**
Parenthesis are used to indicate a function
int means that **main** "returns" an integer value.

return is a way to exit a function from a function.
return 0, in this case, means that the program terminated normally.

Prints the *string* of characters contained between the

ncluding **std::cout**, the **<<**
ing **"Welcome to C++!\n"** and
), is called a *statement*.

All statements must end with a semicolon.

very function

- **`std::cout`**
  - Standard output stream object
  - “Connected” to the screen
  - **`std::`** specifies the "namespace" which **`cout`** belongs to
    - **`std::`** can be removed through the use of **`using`** statements
- **`<<`**
  - Stream insertion operator
  - Value to the right of the operator (right operand) inserted into output stream (which is connected to the screen)
  - **`std::cout << “Welcome to C++!\n”;`**
- **`\`**
  - Escape character
  - Indicates that a “special” character is to be output

Escape Sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double quote character.

```
2 // Printing a line with multiple
3 #include <iostream>
4
5  main()
6 {
7     std::cout << "Welcome ";
8     std::cout << "to C++!\n";
9
10        return 0;    // indicate that
11    }
program ended successfully
```

```
Welcome to C++!
```

Unless new line `'\n'` is specified, the text continues on the same line.

`Using namespace std` statements

Eliminate the need to use the **`std::`** prefix

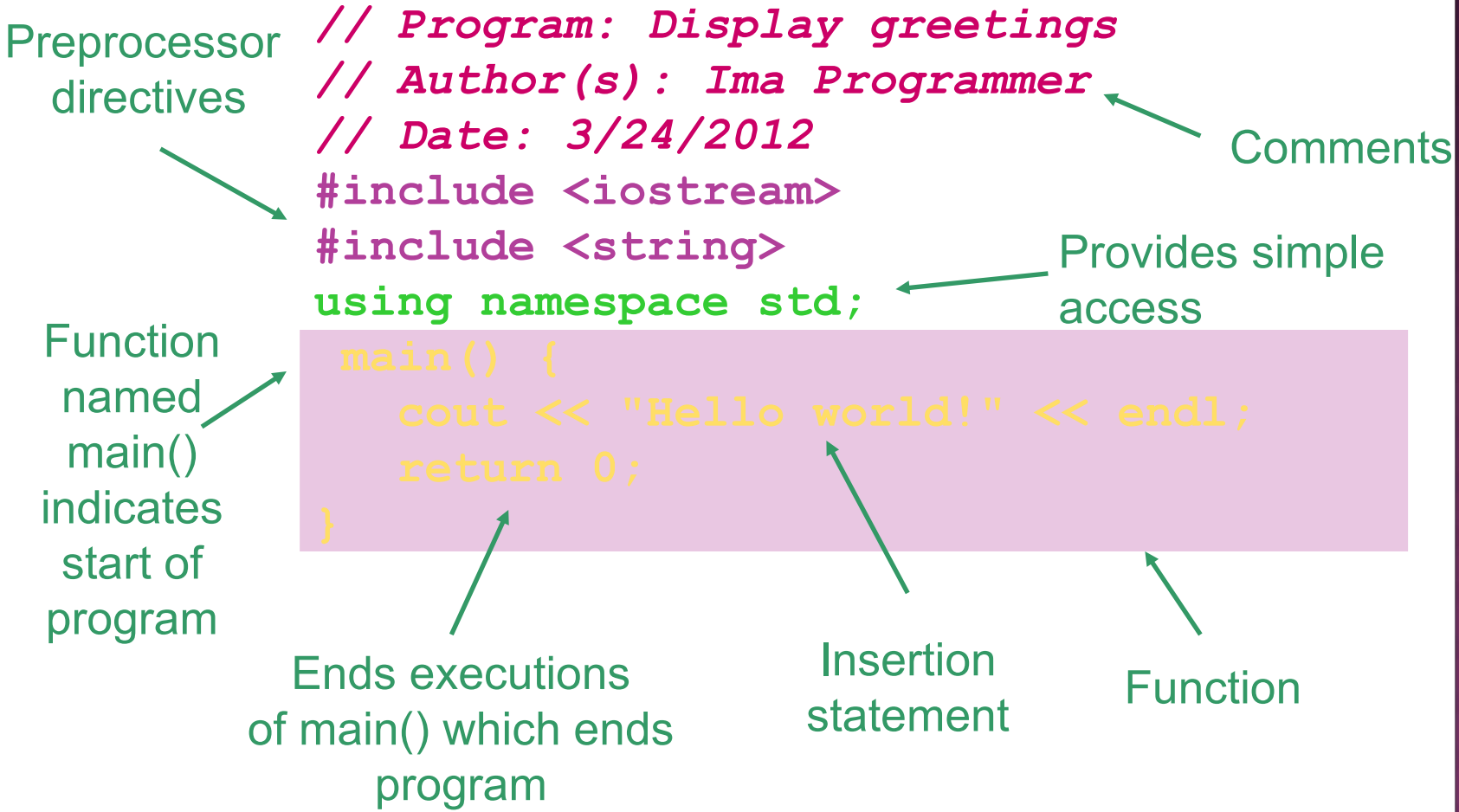
i.e.

```
using std::cout;
using std::cin;
using std::endl;
```

Allow us to write cout instead of

`std::cout`

A First Program - Greeting.cpp



VARIABLE DECLARATION

type variable-name;

Meaning: variable <variable-name> will be a variable of type <type>

Where type can be:

- int //integer
- double //real number
- char //character

Example:

```
int a, b, c;
double x;
int sum;
char my-character;
```

Data types

Name	Description	Size*	Range*
char	Character or small integer	1 byte	signed: -128 to 127 unsigned: 0 to 255
short int (short)	Short integer	2 bytes	signed: -32768 to 32767 unsigned: 0 to 65535
int	Integer	4 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
long int (long)	Long integer	4 bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
float	Floating point number	4 bytes	3.4e +/- 38
double	Double precision floating point number	8 bytes	1.7e +/- 308
long double	Long double precision floating point number	8 bytes	1.7e +/- 308

INPUT STATEMENTS

cin >> variable-name;

Meaning: read the value of the variable called <variable-name> from the user

Example:

```
cin >> a;
cin >> b;
cin >> x;
cin >> my-character;
```

OUTPUT STATEMENTS

cout << variable-name;

Meaning: print the value of variable <variable-name> to the user

cout << “any message “;

Meaning: print the message within quotes to the user

cout << endl;

Meaning: print a new line

Example:

```
cout << a;
cout << b << c;
cout << “This is my character: “ << my-character << “ he he he”
     << endl;
```

Arithmetic Operations

Operator Name	Symbol
Multiplication	*
Division	/
Modulus	%
Addition	+
Subtraction	-

BOOLEAN CONDITIONS

..are built using

- Comparison operators

==	equal
!=	not equal
<	less than
>	greater than
<=	less than or equal
>=	greater than or equal

- Boolean operators

&&	and
||	or
!	not

EXAMPLES

Assume we declared the following variables:

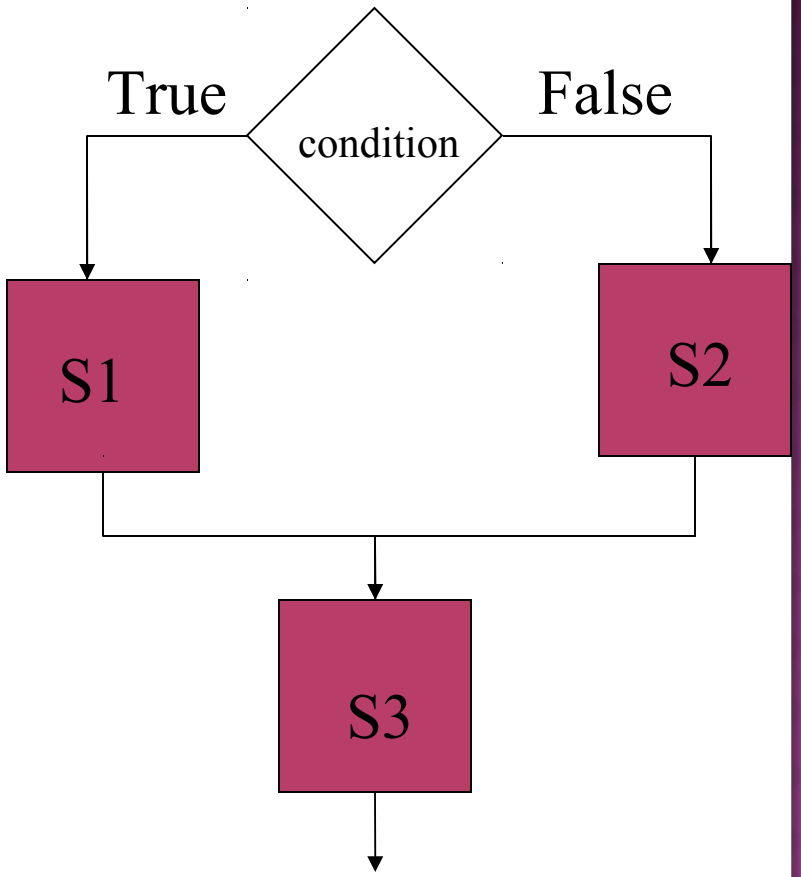
```
int a = 2, b=5, c=10;
```

Here are some examples of boolean conditions we can use:

- `if (a == b) ...`
- `if (a != b) ...`
- `if (a <= b+c) ...`
- `if(a <= b) && (b <= c) ...`
- `if ((a < b) && (b<c)) ...`

IF STATEMENTS

```
if (condition)
{
   S1;
}
else
{
   S2;
}
S3;
```



IF EXAMPLE

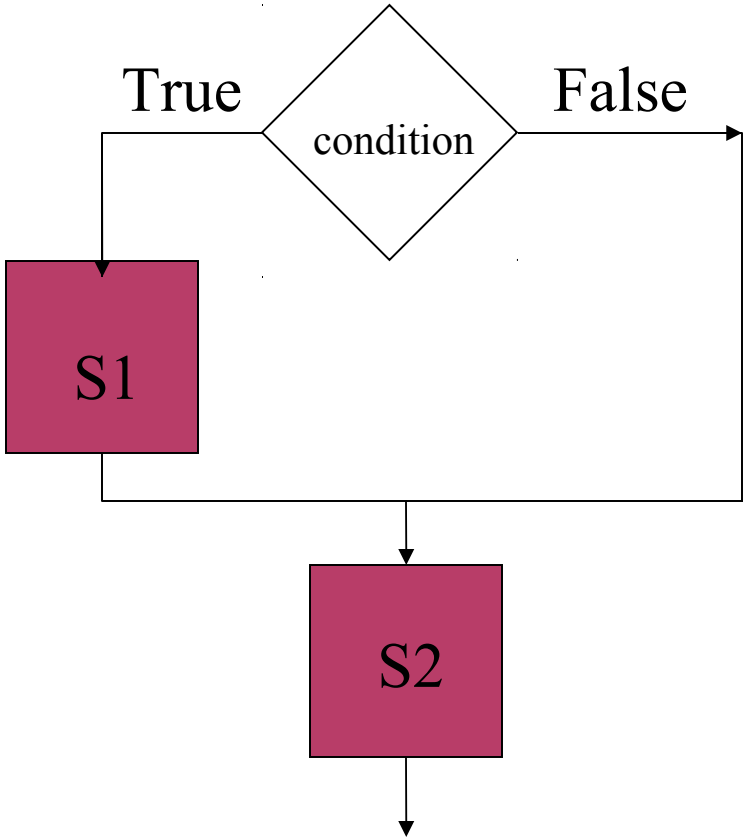
```
#include <iostream.h>

void main() {
int a,b;
cin >> a >> b;

if (a <=b) {
    cout << “min is “ << a << endl;
    }
else {
    cout << “ min is “ << b << endl;
}
cout << “happy now?” << endl;
}
```

WHILE STATEMENTS

```
while (condition)
{
   S1;
}
S2;
```



WHILE EXAMPLE

```
//read 100 numbers from the user and output their sum
#include <iostream.h>

void main()
{
    int i, sum, x;
    sum=0;
    i=1;
    while (i <= 100)
        {
        cin >> x;
        sum = sum + x;
        i = i+1;
    }
    cout << “sum is “ << sum << endl;
}
```

Functions

Functions are easy to use; they allow complicated programs to be broken into small blocks, each of which is easier to write, read, and maintain. This is called modulation.

- How does a function look like?

returntype function_name(*parameters...*)

{

local variables declaration;

function code;

return result;

}

- Sample function

```
int addition(int x, int y)
{
    int add;
    add = x + y;
    return add;
}
```

- How to call a function?

```
main()
{   int result;
    int i = 5, j = 6;
    result  = addition(i, j);
   }
```

thankyou