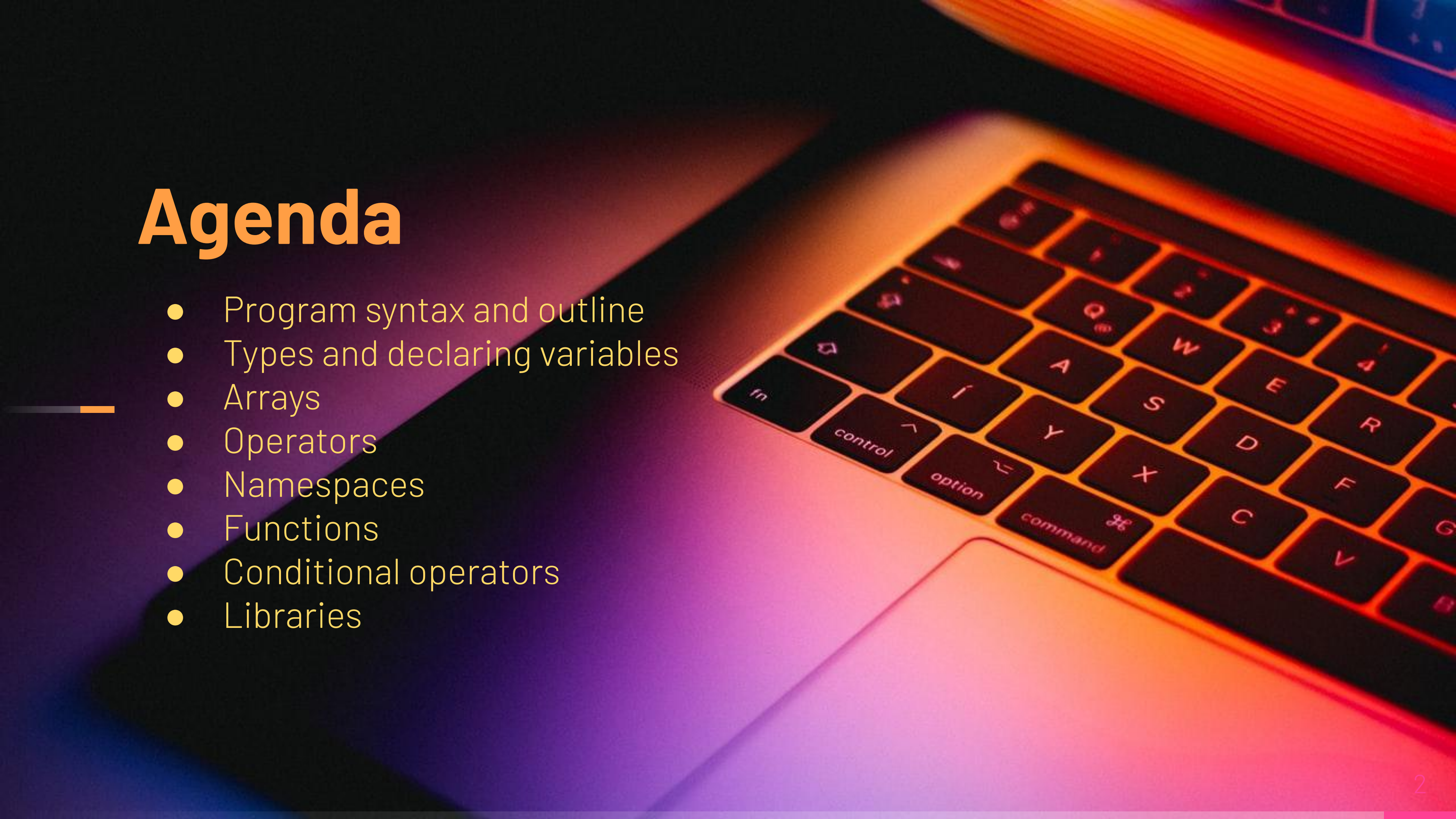
Introduction to C++

Arianna Colón Cesaní, Yarelis Acevedo Ríos, Tiahra Avilés
CMS Experimental Particle Physics Research Group, UPRM



Agenda

- Program syntax and outline
- Types and declaring variables
- Arrays
- Operators
- Namespaces
- Functions
- Conditional operators
- Libraries

Why C++?

- Widely used in many disciplines
- Object oriented
- Plenty of libraries to choose from
- Fast and strongly typed
- Used for hardware and operating systems

Program Outline

```
//First Program

#include <iostream>

int main() {
    std::cout << “Hello World!”;
    Return 0;
}
```

Types and declaring variables

Type	Example
Integer	`int Num = 15;`
Double	`double Num = 5.99;`
Character	`char Letter = 'D';`
String	`string Text = "Hi";`
Boolean	`bool Boolean = true;`

Arrays

Syntax:

```
type arrayName [ arraySize ];
```

Creating an Array:

```
double numbers[4] = {15.0, 2.3, 7.4, 17.5};
char ac[3]={'a','b','c'};
```

Changing an element:

```
numbers[3] = 50.7
```

Accessing a specific element:

```
std::cout<<numbers[2]; (would provide 7.4)
double NUM = numbers[2];
```

2D and 3D Arrays

Initializing a 2D array:

```
int x[3][4] = {{0,1,2,3}, {4,5,6,7}, {8,9,10,11}};
```

The previous 2D array has 3 rows and 4 columns.

Calling an element: `std::cout<<`**`x[2][1]`**`;` (would provide 9)

Initializing a 3D array:

```
int x[2][3][4] =
 {
   { {0,1,2,3}, {4,5,6,7}, {8,9,10,11} },
   { {12,13,14,15}, {16,17,18,19}, {20,21,22,23} }
 };
```

The previous 3D array has two 2D arrays, 3 rows and 4 columns.

Operators

Binary and Assignment Operators

```
int i = 1 + 4 - 2;     // equals 3
i *= 3;                // equals 9
i /=2;                 //equals 4
i = 23 % i;             //Modulus, equals 3
```

- If one or both of the operands are floating point values, the *division operator* performs floating point division (the fraction is kept.)
- If both of the operands are integers, the *division operator* performs integer division instead (drops any fractions).

Comparisons

```
bool a = (3==3);     //true
bool b = (3 != 3);   //false
bool c = (4 < 4);    //false
bool d = (4 <= 4);   //true
```

- Checks whether or not what is inside the parentheses is true.
- Output: True=1, False=0
- >= means "more than **or** equal to"
- <= means "less than **or** equal to"
- != means "**not** equal to"

Example:

```
#include <iostream>


int main()
{
    bool a = (3==3);
    std::cout<< a;

    return 0;
}
```

Output: 1

Incrementing and Decrementing

```
x = x+1;

is the same as:

 x++;
```

```
x = x-1;

is the same as:

 x--;
```

```
x = x+1;

can be written as:

 ++x; // prefix form
 x++; // postfix form
```

Example:

```
x = 5

++ x;     // x becomes 6
x ++;     // x becomes 7
-- x;     // x becomes 6
x --;     // x becomes 5
```

```
int i = 1;
int j = ++i          // i = 2, j = 2
int k = i++;         // i = 3, k = 2
int l = - -i;        //i = 2, l = 2
int m = i - -;       // i = 1, m = 2
```

- For the use of the ++ operator as a prefix, the value is **incremented by 1** and **then** it **returns the value**.
- If you use the ++ operator as a postfix, the original value is **returned first**; **then** it is **incremented by 1**.
- The - - operator works in a similar way, except it decreases the value by 1.

Namespaces

- Prevent name conflicts in large projects.
- Symbols declared inside a namespace block are placed in a named scope that prevents them from being mistaken for identically-named symbols in other scopes.

```
#include <iostream>
using namespace std; //standard
// first name space
namespace first
{
   int val1 = 500;
}
int main()
{
   cout << first::val1 <<"\n";
   return 0;
}
```

Output:

```
  500
```

Functions

Functions with return type

One Parameter

```
#include <iostream>

using namespace std;

int square(int a) {

    return a*a;
}

int main(){
    cout<<square(3);

    return 0;
}
```

Output: 9

Multiple Parameters

```
#include <iostream>

using namespace std;

int sum(int a, int b, int c) {

    return a+b+c;
}

int main(){
    cout<<sum(1,2,3);

    return 0;
}
```

Output: 6

Function Without Return Type

```
#include <iostream>

using namespace std;

void hello() {

     cout<<"Hello everyone";
}

int main(){
    hello();

    return 0;
}
```

Output: Hello everyone

Control Instructions

Control instructions: If

```
if (condition1) {
    Instructions1 ;
} else if (condition2) {
    Instructions2 ;
} else {
    Instructions3;
}
```

- Will only implement “else if” and “else” if the first condition isn’t met and so on.
- The braces are optional if there is a single instruction.

Example: Using “If” Statements

```
#include <iostream>
using namespace std;

void func(int a) {
    cout<<"Is the input less than, equal to, or more than one?"<<"\n";
    if (a==1){
        cout<<"Answer: the number is equal to one"<<"\n";
    }   else if (a<1){
        cout<<"Answer: the number is less than one"<<"\n";
    }   else if (a>1){
        cout<<"Answer: the number is more than one"<<"\n";
    }
}
```

How to call the function?

```
int main() {
    int a;
    cin>>a;
    func(a);
    return 0;
}
```

- int a declares the variable
- cin>> takes user input
- func (a) evaluates the function using the user input and provides output
- return 0 ends the program

Control instructions: Conditional operator

Syntax

test? expression 1 : expression2;

- If test is true, expression 1 is returned
- Else expression 2 is returned

```
#include <iostream>
using namespace std;

int main() {

   int x, y ;

   cout<<"Enter the value of y: ";

   cin>>y;

   x = (y < 5) ? 1 : 2;

   cout << "The value of x is: "
<< x;

   return 0;
}
```

Control instructions: switch

```
switch (identifier){
        case c1 : instructions1 ; break ;
        case c2 : instructions2 ; break ;
        case c2 : instructions3 ; break ;
        default : instructions; break ;
```

- The switch expression is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed

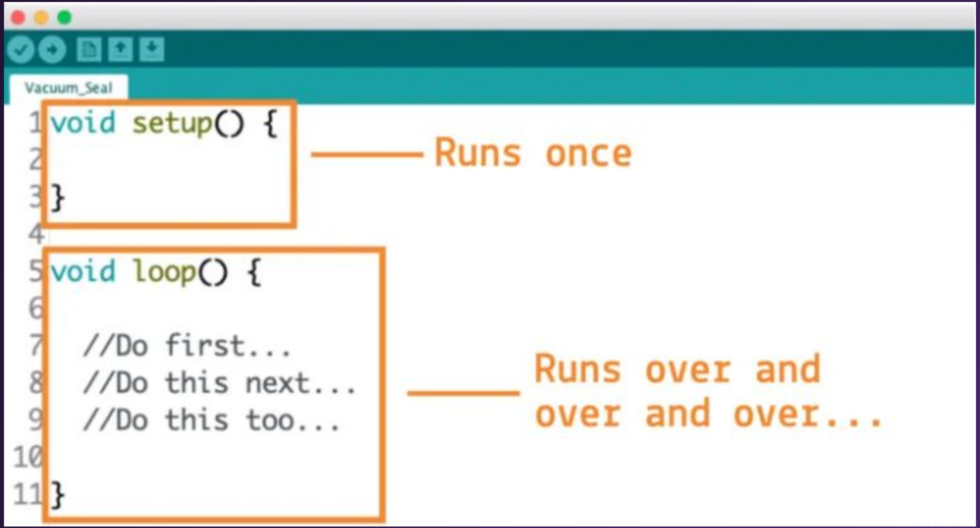
```
#include <iostream>
using namespace std;

int main() {
   cout<<"Select Language: French (1), German (2), English (3)"<<"\n";
   int Language;
   cin>>Language;
   switch (Language) {
       case 1:
        cout<< "Bonjour!";
        break;
       case 2:
        cout<< "Guten tag!";
        break;
       case 3:
        cout<< "Good Morning!";
        break;
       default:
        cout<<"Selection is not within range (1,2,3)";
   }
   return 0;
}
```

Libraries: most common & their uses

- **Boost**- linear algebra, pseudorandom number generation, multithreading, image processing,
- **QT**- building graphical programs that could run on Windows, Linux, and macOS.
- **GSL**- mathematical routines, such as complex numbers, matrix, vectors, and calculus.
- **Asio**- apps and games for mobile devices, dynamic and interactive websites.
- **Eigen** -math and scientific projects, linear algebra, matrices, vectors, numerical solvers.
- **Dlib** - real-world ML and complex algorithms.
- **Poco C++** - network-based web apps for desktop, mobile, and embedded systems.

Language Setup for the Arduino



```
void setup() {

}

void loop() {

  //Do first...
  //Do this next...
  //Do this too...

}
```

Thank You!

Tomorrow:

- Learn about the Arduino!

Questions?