Introduction to C++

CΦ

Centre For Innovation

Programming Club
IIT Madras

CΦ

Centre For Innovation

What is C++?

- C++ is a High Level Object Oriented Programming Language
- Built as an extension to C
- One of the most popular programming languages in the World

Omicron

CΦ

Centre For Innovation

Why use C++

- C++ is very powerful i.e It has a support for a lot of features
- C++ is faster than Python
- C++ is widely used in the CP community - lot of resources exist

Omicron

Centre For Innovation

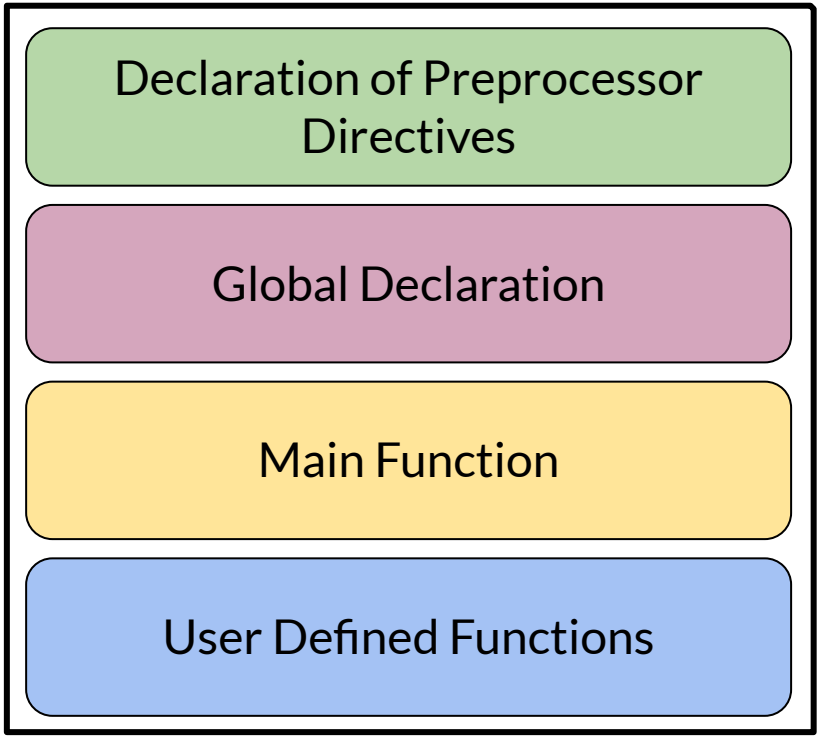
Standard Libraries in C++

- The Standard Template Library (STL) is a set of C++ template classes
- Provides common programming data structures and functions such as lists, stacks, arrays, etc
- Mainly three parts
  - Containers
  - Algorithms
  - Iterators
- More Details in another session - Stay Tuned xD

Omicron

Centre For Innovation

Structure of a C++ Program



Omicron

Centre For Innovation

Simple Program in C++

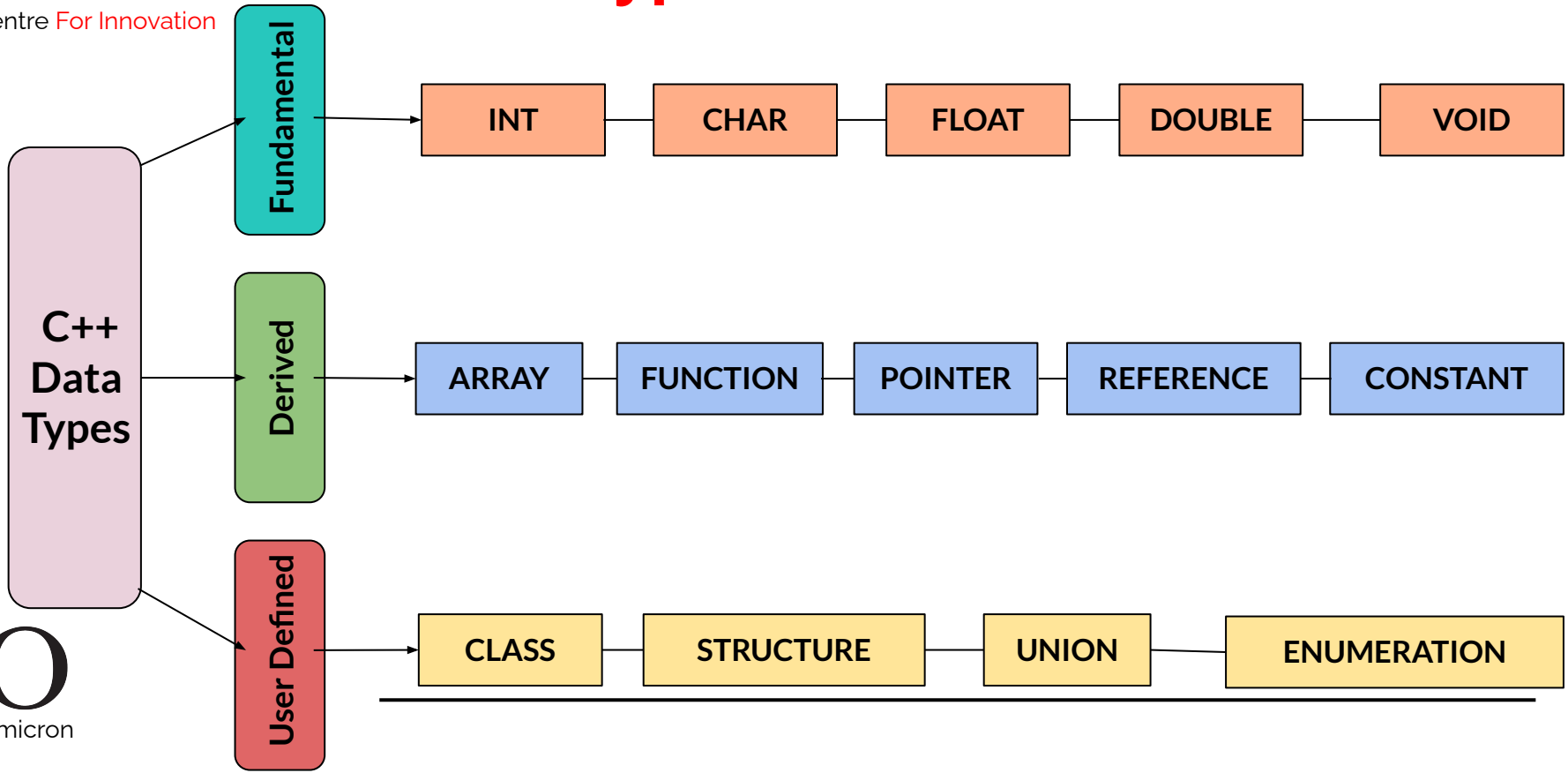
```
#include <iostream>  // Header File
using namespace std;

int main() {
   cout << "Hello, World!";  /* Prints Hello World */
   return 0;
}
```

Omicron

CΦ

Centre For Innovation

Data Types

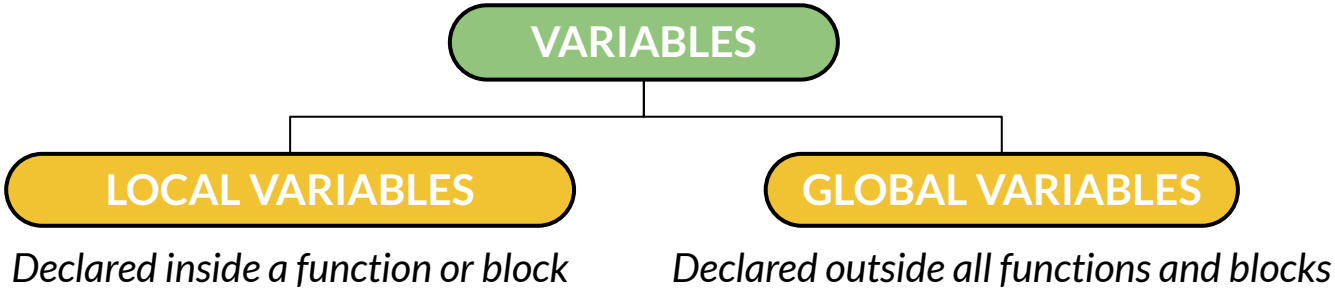


Omicron

Centre For Innovation

Scope of Variables

- The program parts in which the variable can be accessed
- Determined by place of declaration



Omicron

Centre For Innovation

Local Variables vs Global Variables

```
//Local Variable
#include<iostream>
using namespace std;

void f1(){
    int age1=18;
}
int main(){
     age1 = 17;
     f1();
     cout<<"Age is: "<<age1;
     return 0;
}
```

```
//Global Variable
#include<iostream>
using namespace std;

int age2 = 5;
void display(){
    cout<<age2<<endl;
}
int main(){
    display();
    age2 = 10;
    display();
}
```

Omicron

Centre For Innovation

Operators

I/O	>>, <<
Arithmetic	+, -, *, /, %
Increment/ Decrement	++, --
Relational	<, >, <=, >=, !=
Logical	||, &&, !
Conditional	?:
Bitwise	|, &, ^, >>, <<, ~
Others	sizeof()

Omicron

Centre For Innovation

Decision Making

- Decision making statements decide flow of program execution
- C++ has three decision making statements:
  - *if* - single selection statement
  - *if .. else* - double selection statement
  - *switch* - multiple selection statement

Omicron

Centre For Innovation

The *if* Statement

- *if* is the most simple decision statement
- Used to decide whether a block of code is executed or not.

```
int a;
cin >> a;
if(a % 2 == 1){
      a++;
}
```

Omicron

Centre For Innovation

The *if .. else* Statement

- The *if* statement tells us what code to execute when a condition is true
- If the condition is false, the *else* statement tells us another block of code to be executed

```
int a;
cin >> a;
if(a % 2 == 1){
    cout << "is odd";}
else{
    cout << "is even";}
```

Omicron

Centre For Innovation

The *switch* Statement

- *switch* is used to select between multiple cases.

```
char x = 'W';
switch (x)
{
     case 'W':cout << "You win!\n";          break;
     case 'D':cout << "It's a draw\n";       break;
     case 'L':cout << "You lose!\n";         break;
     default:  cout << "invalid case -_-\n";
               break;
}
```

Omicron

Centre For Innovation

The *if..else..if* ladder

- The *if..else* statement can be extended to cover multiple conditions like a *if..else..if* ladder

```
int a = 0;
if(a > 0) {
     cout << "a is +ve";}
else if(a < 0) {
     cout << "a is -ve";}
else{
     cout << "a is 0";}
```

Omicron

Centre For Innovation

The nested *if* statement

- The *if* statement can be extended to cover multiple conditions.

```
int a = 0;
if(a % 2 == 0){
    if(a % 4 == 0){
        if(a% 8 == 0){
            cout << "a is a multiple of 8";}
        cout << "a is a multiple of 4";}
    cout << "a is a multiple of 2";}
```

Omicron

Centre For Innovation

Loop Statements

- Loops are used to repeatedly execute the same statement
- A loop is a sequence of instructions that is repeated until a certain condition is met
- C++ has three types of loops:
  - *for*
  - *while*
  - *do while*

Omicron

Centre For Innovation

The *for* loop

- The loops run as long as the condition is true
- Format:
  - *for(<initialization>;<condition>;<updation>){<statements>}*

```
int i = 0;
for(i = 1; i <= 10; i++)
{
    cout << "Hi there! \n";
}
```

Omicron

Centre For Innovation

The *while* loop

- Similar to *for* loop, used when no. of iterations is unknown
- Format:
  - *while(<condition>){ <statements>}*

```
int a = 128;
while(a > 1)
{
    a = a/2;
}
```

Omicron

Centre For Innovation

The *do while* loop

- Similar to *while* loop, but the test condition is at the end of the loop, so the loop runs at least once
- Format:
  - *do { <statements> } while(<condition>) ;*

```
int i = 2;
do
{
    cout << "Hello World \n";
    i++;
} while (i < 2);
```

Omicron

Centre For Innovation

Functions

- Named unit of a group of program statements used to perform a certain task
- Can be invoked from other parts of the program as and when required
- Important for reusing code
- C++ provides some predefined functions but we can also create our own functions to perform certain actions

Omicron

Centre For Innovation

Defining a Function

<return-type> function-name (parameter list){

// function-body

}

```
void swap(int x, int y) {

    int temp;
    temp = x;
    x = y;
    y = temp;

    return;
}
```

Omicron

Centre For Innovation

Declaring a Function

function-name (type varname1, type varname2, ... , type varnamen);

```
void swap (int x, int y);
```

- For a function it's prototype must be provided before it's use.
- However, if the called function's definition appears before the calling function's definition, then the called function's prototype may be skipped because a definition itself is a prototype of a function.

Omicron

Calling a function

Centre For Innovation

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

void swap(int x, int y);
int main () {
   // local variable declaration
   int a = 200, b=100;
   // calling the function to swap the values
   swap(a, b);
   return 0;
}
```

Formal Parameter:

The variable that appears in the definition of the function

Actual Parameter:

The variable or expression corresponding to a formal parameter that appears in the function

Omicron

Centre For Innovation

Call by Value

- Values of actual parameters are copied into formal parameters
- Any change that occurs inside function is on the function's copy of the argument value
- The original copy of the argument value remains intact
- Even if the names of actual and formal parameters match, separate copy of values is created by the function

Omicron

Centre For Innovation

Call by Value (example)

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

void swap(int x, int y);
int main () {

   int a = 100, b=200;
   cout << "Before swap, values of a and b :" << a << b << endl;
   swap(a, b);
   cout << "After swap, values of a and b :" << a << b << endl;
   return 0;
}
```

Omicron

Centre For Innovation

Reference

```
#include <iostream>
using namespace std;
int main () {
   int i;
   int& r = i;
   i = 5;
   cout << "Value of i : " << i << endl;
   cout << "Value of i reference : " << r  << endl;
   return 0;
}
```

Omicron

Centre For Innovation

Call by Reference

- Formal parameters become references to the actual parameters
- Function does not create its own copy of the argument values
- Any change that occurs inside function gets reflected on the passed values
- Only variables can be passed by reference, not constants or expressions

Omicron

Centre For Innovation

Call by Reference (example)

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

void swap(int &x, int &y);
int main () {

   int a = 100, b=200;
   cout << "Before swap, values of a and b :" << a << b << endl;
   swap(a, b);
   cout << "After swap, values of a and b :" << a << b << endl;
   return 0;
}
```

Omicron

Centre For Innovation

Recursive Functions

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

int fact(int n) {
    if ((n==0)||(n==1))
     return 1;
    else
     return n*fact(n-1);}

int main() {
    int n = 4;
    cout<<"Factorial of "<<n<<" is "<<fact(n);
    return 0;}
```

A function calling itself in its own body

Omicron

Centre For Innovation

Arrays

- Defined as a set of homogeneous data items
- A data structure which allows a collective name to be given to a group of elements which all have the same type
- Syntax:
  - *datatype <array_name>[array_size];*

Omicron

Centre For Innovation

```
int a[5] = {1,2,3,4,5};
cout << a[0];
a[2] = 90;
cout << a[2];
```

- In *int a[5]*
  - *int = datatype*
  - *a = name of the array*
  - *5 = size of the array*
- The size of the array must be an integer constant and the data type can be any valid C++ data type.

Omicron

Centre For Innovation

```
int a[5] = {1,2,3,4,5};
cout << a[0];
a[2] = 90;
cout << a[2];
```

- In C++, elements of an array can be initialized one by one or using a single statement (like above example)
- Element can be accessed by *array_name[<index>]*
- Indexing in C++ starts from 0, i.e. first element has index 0 and the last element (in array of size n) is n-1

Omicron

Centre For Innovation

Arrays (contd.)

- C++ supports multidimensional arrays

```
float t[2][3]; //similar to a 2 x 3 matrix
```

- We can generate a pointer to the array by specifying name of the array without any index
- We can pass an array to a function using the above mentioned pointer
- We can also return arrays from a function

Omicron

Centre For Innovation

Class

- A class is a user defined datatype
- It contains its own data and functions
- An Object is an instance of a class
- Enables us to “bundle” together data and accompanying functions

We know this sounds a bit vague xD, stay with us

Omicron

Centre For Innovation

```
Class Abc{
private:
   int a;int b;                          //data
public:
   void putvalue(int a1, int b1){    //functions
      a = a1;
      b = b1;}
   int sum(){
      return a + b;}
};

int main(){
   Abc obj;      //obj.a = error, as a is a private member
   obj.putvalue(3, 4);
   cout << obj.sum();
}
```

Omicron

Centre For Innovation

Setting up C++ in your PC

- IDEs work great if you're looking for a setup that you can edit in, compile, build and run all at the same place
- In Linux, you can download the g++ compiler and use it along with any text editor
  - g++ <filename>.cpp -o <execname>
  - ./<execname>
- In Windows, VS Code is a good option if you're looking for a lightweight setup with IDE like properties
  - https://code.visualstudio.com/docs/cpp/config-mingw

Omicron

CΦ

Centre For Innovation

Resources/Useful Sites

- C++ Tutorials
  - https://www.geeksforgeeks.org/c-plus-plus/
  - https://www.tutorialspoint.com/cplusplus/index.htm
- Competetive Programming
  - https://codeforces.com/
  - https://www.codechef.com/

Omicron