**Programming Fundamental**

|  |  |  |
| --- | --- | --- |
| **SL** | **Name** | **Topics Name** |
| **01** | **C++ Getting Started** | **Compiler:**  A compiler, like ***GCC(***[***GNU Compiler Collections***](https://www.geeksforgeeks.org/gcc-command-in-linux-with-examples/)***)***, to translate the C++ code into a language that the compute will understand.  **C++ Install IDE:** An ***IDE*** (***Integrated Development Environment***) is used to **edit** AND **compile** the code.  Popular **IDE's** include ***Code::Blocks***, Eclipse, and ***Visual Studio***. These are all free, and they can be used to both edit and debug C++ code. **C++ Quickstart:** myfirstprogram.cpp |
| **02** | C++ Syntax | **Syntax:**  **#include <iostream>**  **#include <bits/stdc++.h> using namespace std;**  **int main() {**  **//All code run here..........!   cout << "Hello World!";   return 0; }**  **Line-1:** #include<iostream> is a **header file library** that lets us work with ***input*** and ***output*** objects, such as cout (used in line 5). Header files add functionality to C++ programs.  **Line-2:** using namespace std means that we can use names for **objects** and **variables** from the standard library. **Omitting Namespace:****#include <iostream>  int main() {   std::cout << "Hello World!";   return 0; }****Statements:**  1. **printf("Hello World! ");** 2. **printf("Have a good day!");** 3. **return 0;** |
| **03** | C++ Output | **C++ Output (Print Text):** **#include <iostream> using namespace std;  int main() {   cout << "Hello World!";   cout << "I am learning C++";   return 0; }**  **C++ New Lines:**  **#include <iostream> using namespace std;  int main() {   cout << "Hello World!" << "\n";**  **cout << "Hello World!" << "\n\n";**  **cout << "Hello World!" << endl;   cout << "I am learning C++";   return 0; }** |
| **04** | C++ Comments | **Single-line Comments:** **#include <iostream> using namespace std; int main() {**  **//cout << "Hello World!" << "\n\n";   //cout << "I am learning C++";   return 0; }** **C++ Multi-line Comments:** **#include <iostream> using namespace std; int main() {**  **/\*cout << "Hello World!" << "\n\n";   cout << "I am learning C++";\*/   return 0; }** |
| **05** | C++ Variables/Data Types | **C++ Variables:** Variables are containers for storing data values. In C++, there are different **types** of variables (defined with different keywords), for example:   * **int** - stores **integers** (whole numbers), without decimals, such as **123** or **-123**. ***Store 4 bytes***. * **double** - stores **floating** point numbers, with decimals, such as **19.99** or **-19.99. *Store 8 bytes***. * **char** - stores **single characters**, such as **'a'** or '**B**'. Char values are surrounded by single quotes. ***Store 1 bytes***. * **string** - **stores text**, such as **"Hello World".** String values are surrounded by double quotes. ***Store terminate null bytes***. * bool - stores values with two states: true or false. ***Store 1 bytes***.  **Declaring (Creating) Variables:****Syntax:** ***type variableName = value;***  ***Example:***  **int myNum = 5; //Integer (whole number without decimals) double myFloatNum = 5.99; //Floating point number (with decimals) char myLetter = 'D'; //Character string myText = "Hello"; // String (text) bool myBoolean = true; // Boolean(true or false)**  **cout<<"I am "<<myNum<<" years old.";** **Declare Many Variables:** **int x = 5, y = 6, z = 50; cout << x + y + z;** C++ Identifiers: All C++ ***variables*** must be ***identified*** with ***unique names***.  These unique names are called **identifiers**. // Good int minutesPerHour = 60;  // OK, but not so easy to understand what **m** actually is int m = 60; **The general rules for naming variables are:**   * Names can contain **letters**, **digits** and **underscores.** * Names must **begin** with a **letter** or an ***underscore*** (\_) * Names are **case-sensitive**. (**myVar** and **myvar** are different variables) * Names cannot contain **whitespaces** or **special characters like !, #, %, etc**. * **Reserved** words (like C++ keywords, such as int) cannot be used as names.  C++ Constants:When you do not want others (or yourself) to change existing variable values, use the const keyword (this will declare the variable as "constant", which means unchangeable and read-only):const int minutesPerHour; minutesPerHour = 60; // error |
| **06** | C++ User Input | C++ User Input: **cin** is a predefined variable that ***reads data from the keyboard*** with the extraction operator (**>>**).  **int x;  cout << "Type a number: "; // Type a number and press enter cin >> x; // Get user input from the keyboard cout << "Your number is: " << x; // Display the input value** |
| **07** | C++ Operators | **Operators in C++ can be classified into 6 types:**   1. Arithmetic Operators   **Binary Operators:**  Addition(+),Subtraction(-),  Multiplication(\*),**Division**(/),**Modulus**(%)  **Unary Operators:**  **Increment(++)**  **Decrement(--)**  **Use as postfix/prefix:**  **#include <iostream>**  **using namespace std;**  **int main() {**  **int x = 5;**  **++x;**  **cout << x <<endl; //6**  **cout << ++x <<endl; //7**    **int y = 5;**  **y++;**  **cout << y <<endl; //6**  **cout << y++ <<endl; //6**    **return 0;**  **}**   1. Relational Operators   **Is Equal To (==)**  **Greater Than**  **(>)**  **Greater Than or Equal To** **(>=)**  **Less Than** **(<)**  **Less Than or Equal To** **(<=)**  **Not Equal To** **(!)**   1. Logical Operators   **Logical AND (&&)**  **Logical OR (||)**  **Logical NOT (!)**   1. Bitwise Operators   **Binary AND (&)**  **Binary OR (|)**  **Binary X-OR (^)**  **Left Shift (<<)**  **Right Shift (>>)**  **One’s Complement (~)**  **Binary AND(&): (Multiple)**   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **32** | **=** | **1** | **0** | **0** | **0** | **0** | **0** | | **12** | **=** | **0** | **0** | **1** | **1** | **0** | **0** | | **32&12** | | **0** | **0** | **0** | **0** | **0** | **0** | | **Decimal value= 0** | | | | | | | |   **Binary OR(|): (Add)**   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **32** | **=** | **1** | **0** | **0** | **0** | **0** | **0** | | **12** | **=** | **0** | **0** | **1** | **1** | **0** | **0** | | **32|12** | | **1** | **0** | **1** | **1** | **0** | **0** | | **Decimal value= 44** | | | | | | | |   **Binary X-OR(^): (Odd I/P 1, then 1)**   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **32** | **=** | **1** | **0** | **0** | **0** | **0** | **0** | | **12** | **=** | **0** | **0** | **1** | **1** | **0** | **0** | | **32^12** | | **1** | **0** | **1** | **1** | **0** | **0** | | **Decimal value= 44** | | | | | | | |   **Left Shift(<<): 32<<2**   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **32** | **=** | **1** | **0** | **0** | **0** | **0** | **0** | | **Left-1** |  |  | **1** | **0** | **0** | **0** | **0** | | **Left-2** |  |  |  | **1** | **0** | **0** | **0** | | **32<<2** | | **= 8** | | | | | | | **Sort: 32/2n** | | | | | | | |   **Right Shift(>>): 32<<2**   |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **32** | | **=** |  |  | **1** | **0** | **0** | **0** | **0** | **0** | | **Right-1** | |  |  | **1** | **0** | **0** | **0** | **0** | **0** | **0** | | **Right-2** | |  | **1** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | | **32>>2** | | | **= 128** | | | | | | | | | **Sort:** | **Sort: 32\*2n** | | | | | | | | | |   **One’s Complement (~)**  **Changes binary digits 1 to 0 and 0 to 1.**  **int b = 3;**  **(~b); //returns -4**   1. Assignment Operators   **Assignment Operator (=)**  **Add and Assignment Operator (+=)**  **Substruct and Assignment Operator (-=)**  **Multiply and Assignment Operator (\*=)**  **Divide and Assignment Operator (/=)**   1. Ternary or Conditional Operators   **Expression1 ? Expression2: Expression3**  **int result = (a < b) ? b : a;**  **cout<<"The greatest number is "<<result<<endl;** |
| **08** | Conditon | **C++ Conditions and If Statements:** **if (condition) {   *// block of code to be executed if the condition is true* }** **Example:** **if (20 > 18) {   cout << "20 is greater than 18"; }** C++ Else:if (condition) {   *// block of code to be executed if the condition is true* } else {   *// block of code to be executed if the condition is false* }**Example:**int time = 20; if (time < 18) {   cout << "Good day."; } else {   cout << "Good evening."; } // Outputs "Good evening.C++ Else If:if (condition1) {   *// block of code to be executed if condition1 is true* |
|  |  | } else if (condition2) {   *// block of code to be executed if the condition1 is false and condition2 is true* } else {   *// block of code to be executed if the condition1 is false and condition2 is false* }**Example:****int time = 22; if (time < 10) {   cout << "Good morning."; } else if (time < 20) {   cout << "Good day."; } else {   cout << "Good evening."; } // Outputs "Good evening."****C++ Short Hand If Else/Short Hand If...Else (Ternary Operator):**variable = (condition) ? expressionTrue : expressionFalse; |
| **09** | C++ Switch | **C++ Switch Statements:** **switch(expression) {   case x:     *// code block*     break;   case y:     *// code block*     break;   default:     *// code block* }** **Example:****int day = 4; switch (day) {   case 1:     cout << "Monday";     break;   case 2:     cout << "Tuesday";     break;   case 3:     cout << "Wednesday";     break;   case 4:     cout << "Thursday";     break;   case 5:     cout << "Friday";     break;   case 6:     cout << "Saturday";     break;   case 7:     cout << "Sunday";     break; } // Outputs "Thursday" (day 4)** |
| **10** | Loop | C++ While Loop:Initialization;while (condition) { *// code block to be executed* *//Increment/decrement;* }**Example:****int i = 0; while (i < 5) {   cout << i << "\n";   i++; }**C++ Do/While Loop:do { *// code block to be executed* } while (condition);**Example:**int i = 0; do {   cout << i << "\n";   i++; } while (i < 5);C++ For Loop:for (initialization;*condition*;*increment/decrement*) {   *// code block to be executed* }**Example:** **for (int i = 0; i <= 10; i = i + 2) {   cout << i << "\n"; }** C++ Nested Loops:// Outer loop for (int i = 1; i <= 2; ++i) {   cout << "Outer: " << i << "\n"; // Executes 2 times    // Inner loop   for (int j = 1; j <= 3; ++j) {     cout << " Inner: " << j << "\n"; // Executes 6 times (2 \* 3)   } } **Outer: 1**  **Inner: 1**  **Inner: 2**  **Inner: 3**  **Outer: 2**  **Inner: 1**  **Inner: 2**  **Inner: 3** C++ The foreach Loop: **for (type variableName : arrayName) {   // code block to be executed }** **Example:** **int myNumbers[5] = {10, 20, 30, 40, 50}; for (int i : myNumbers) {   cout << i << "\n"; }** |
| **11** | C++ Break and Continue | **C++ Break:****for (int i = 0; i < 10; i++) {   if (i == 4) {     break;   }   cout << i << "\n"; }****C++ Continue:** **for (int i = 0; i < 10; i++) {   if (i == 4) {     continue;   }   cout << i << "\n"; }** |
| **12** | C++ Arrays |  |
| **13** | C++ References | **Creating Reference:** A **reference** variable is a "reference" to an existing variable, and it is created with the **&** operator: **string food = "Pizza"; string &meal = food;  cout << food << "\n";  // Outputs Pizza cout << meal << "\n";  // Outputs Pizza**C++ Memory Address: The **&** operator was used to create a **reference** variable. But it can also be used to get the ***memory address*** of a variable. **Example:** **string food = "Pizza"; cout << &food; // Outputs 0x6dfed4** |
| **14** | C++ Pointers | **Creating Pointers:**A **pointer** however, is a variable that **stores the memory address as its value**.**Example:** **string food = "Pizza";  // A food variable of type string string\* ptr = &food;    // A pointer variable, with the name ptr, that stores the address of food  // Output the value of food (Pizza) cout << food << "\n";  // Output the memory address of food (0x6dfed4) cout << &food << "\n";  // Output the memory address of food with the pointer (0x6dfed4) cout << ptr << "\n";** C++ Dereference:We used the pointer variable to get the memory address of a variable (used together with the & reference operator). However, you can also use the pointer to get the value of the variable, by using the \* operator (the dereference operator):**Example:** **string food = "Pizza";  // Variable declaration string\* ptr = &food;    // Pointer declaration  // Reference: Output the memory address of food with the pointer (0x6dfed4) cout << ptr << "\n";  // Dereference: Output the value of food with the pointer (Pizza) cout << \*ptr << "\n";** C++ Modify Pointers:**Example:**string food = "Pizza"; string\* ptr = &food;  // Output the value of food (Pizza) cout << food << "\n";  // Output the memory address of food (0x6dfed4) cout << &food << "\n";  // Access the memory address of food and output its value (Pizza) cout << \*ptr << "\n";  // Change the value of the pointer \*ptr = "Hamburger";  // Output the new value of the pointer (Hamburger) cout << \*ptr << "\n";  // Output the new value of the food variable (Hamburger) cout << food << "\n"; |
| **15** | C++ Functions | C++ Functions:A function is a **block of code** which ***only runs when it is called***. You can ***pass data***, known ***as parameters***, into a function. **Create a Function:**  1. Return data type. 2. Function name. 3. Parameter/Argument. (If you want).  **Function Declaration and Definition:**  * **Declaration:** the **return type**, the **name of the function**, and **parameters** (if any). * **Definition:** the body of the function (***code to be executed***).   **Return-type function-name(parameters){** Body of the function (***code to be executed***).  **}**  **void myFunction() {   // code to be executed }** **Call a Function:** **// Create a function void myFunction() {   cout << "I just got executed!"; }  int main() {   myFunction(); // call the function   return 0; }  // Outputs "I just got executed!"** C++ Function Parameters:**Example:** **void functionName(parameter1, parameter2, parameter3) {   // code to be executed }**  **void myFunction(string fname) {   cout << fname << " Refsnes\n"; }  int main() {   myFunction("Liam");   myFunction("Jenny");   myFunction("Anja");   return 0; }  // Liam Refsnes // Jenny Refsnes // Anja Refsnes** C++ Default Parameters:**Example:**void myFunction(**string country =**"Norway") {   cout << country << "\n"; }  int main() {   myFunction("Sweden");   myFunction("India");   **myFunction();**   myFunction("USA");   return 0; }  // Sweden // India // Norway // USAC++ Multiple Parameters:**Example:**void myFunction(**string fname,**int**age**) {   cout << fname << " Refsnes. " << age << " years old. \n"; }  int main() {   myFunction("Liam"**, 3**);   myFunction("Jenny"**, 14**);   myFunction("Anja"**, 30**);   return 0; }  // Liam Refsnes. 3 years old. // Jenny Refsnes. 14 years old. // Anja Refsnes. 30 years old.C++ The Return Keyword:**Example:**int myFunction(int x) {   return 5 + x; }  int main() {   cout << myFunction(3);   return 0; }  // Outputs 8 (5 + 3)int myFunction(int x, int y) {   return x + y; }  int main() {   cout << myFunction(5, 3);   return 0; }  // Outputs 8 (5 + 3)C++ Functions - Pass By Reference:**Example:** **void swapNums(int &x, int &y) {   int z = x;   x = y;   y = z; }  int main() {   int firstNum = 10;   int secondNum = 20;    cout << "Before swap: " << "\n";   cout << firstNum << secondNum << "\n";    // Call the function, which will change the values of firstNum and secondNum   swapNums(firstNum, secondNum);    cout << "After swap: " << "\n";   cout << firstNum << secondNum << "\n";    return 0; }** C++ Pass Array to a Function:**Example:**void myFunction(int myNumbers[5]) {   for (int i = 0; i < 5; i++) {     cout << myNumbers[i] << "\n";   } }  int main() {   int myNumbers[5] = {10, 20, 30, 40, 50};   myFunction(myNumbers);   return 0; } |
| **16** | **C++ Function Overloading** | **Function Overloading:****int myFunction(int x) float myFunction(float x) double myFunction(double x, double y)****Example:** **int plusFuncInt(int x, int y) {   return x + y; }  double plusFuncDouble(double x, double y) {   return x + y; }  int main() {   int myNum1 = plusFuncInt(8, 5);   double myNum2 = plusFuncDouble(4.3, 6.26);   cout << "Int: " << myNum1 << "\n";   cout << "Double: " << myNum2;   return 0; }** |
| **17** | C++ Variable Scope | **Local Scope:****void myFunction() {   // Local variable that belongs to myFunction   int x = 5;    // Print the variable x   cout << x; }  int main() {   myFunction();**   **return** 0; } A **local variable** cannot be used outside the function it belongs to.  **void myFunction() {   // Local variable that belongs to myFunction   int x = 5; }  int main() {   myFunction();    // Print the variable x in the main function   cout << x;   return 0; }** **Global Scope:** **// Global variable x int x = 5;  void myFunction() {   // We can use x here   cout << x << "\n"; }  int main() {   myFunction();    // We can also use x here   cout << x;   return 0; }**  **// Global variable x int x = 5;  void myFunction() {   // Local variable with the same name as the global variable (x)   int x = 22;   cout << x << "\n"; // Refers to the local variable x }  int main() {   myFunction();    cout << x; // Refers to the global variable x   return 0; }** |
| **18** | C++ Recursion | **Recursion:** Recursion is the technique of making a **function call itself**. This technique provides a way to break complicated problems down into simple problems which are easier to solve.**Example:****int sum(int k) {   if (k > 0) {     return k + sum(k - 1);   } else {     return 0;   } }  int main() {   int result = sum(10);   cout << result;   return 0; }** 10 + sum(9) 10 + ( 9 + sum(8) ) 10 + ( 9 + ( 8 + sum(7) ) ) ... 10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0) 10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0 |
|  |  |  |
|  |  |  |