**Usman Obaid**

**Id#13332**

**Programming Fundamental**

|  |  |  |
| --- | --- | --- |
| **Q.1** |  |  |
|  |  | **What is the purpose of *if statement*? Discuss its two different forms with examples.** |

**Ans** The if statement is used to check a condition and if the condition is true, we run a block of statements (called the if-block), else we process another block of statements (called the else-block). The else clause is optional.

## **Nested If else statement**:

When an if else statement is present inside the body of another “if” or “else” then this is called nested if else.

### **Example of nested if else:**

#**include <stdio.h>**

**int main()**

**{**

**int var1, var2;**

**printf("Input the value of var1:");**

**scanf("%d", &var1);**

**printf("Input the value of var2:");**

**scanf("%d",&var2);**

**if (var1 != var2)**

**{**

**printf("var1 is not equal to var2\n");**

**//Nested if else**

**if (var1 > var2)**

**{**

**printf("var1 is greater than var2\n");**

**}**

**else**

**{**

**printf("var2 is greater than var1\n");**

**}**

**}**

**else**

**{**

**printf("var1 is equal to var2\n");**

**} return 0;**

**}**

## **Else if statement:**

## The else if statement is useful when you need to check multiple conditions within the program, nesting of if-else blocks can be avoided using else if statement.

### **Example of else..if statement**

Lets take the same example that we have seen above while discussing nested if..else. We will rewrite the same program using else..if statements

**#include <stdio.h>**

**int main()**

**{**

**int var1, var2;**

**printf("Input the value of var1:");**

**scanf("%d", &var1);**

**printf("Input the value of var2:");**

**scanf("%d",&var2);**

**if (var1 !=var2)**

**{**

**printf("var1 is not equal to var2\n");**

**}**

**else if (var1 > var2)**

**{**

**printf("var1 is greater than var2\n");**

**}**

**else if (var2 > var1)**

**{**

**printf("var2 is greater than var1\n");**

**}**

**else**

**{**

**printf("var1 is equal to var2\n");**

**}**

**return 0;**

|  |  |
| --- | --- |
|  | **Write a C++ program to read two numbers from keyboard and then find the  LARGEST number of them.** |

**Ans #include <iostream>**

**using namespace std;**

**int main()**

**{**

**int num1, num2;**

**cout<<"Enter first number:";**

**cin>>num1;**

**cout<<"Enter second number:";**

**cin>>num2;**

**if(num1>num2)**

**{**

**cout<<"First number "<<num1<<" is the largest";**

**}**

**else**

**{**

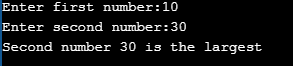
**cout<<"Second number "<<num2<<" is the largest";**

**}**

**return 0;**

**}**

**Output:**



|  |  |  |
| --- | --- | --- |
| **Q.2** | **a)** | **What are the Logical Operators? Explain them.** |

**Ans: LOGICAL OPERATORS:**

A **logical operator** takes Boolean operands and yields a Boolean result. In other words, its

operands and result are truth values (**true** or **false**).

**||(OR):**

The “OR” operator is represented with two vertical line symbols:

result = a || b;

**&&(AND):**

The AND operator is represented with two ampersands &&:

result = a && b;

**!(NOT):**

The boolean NOT operator is represented with an exclamation sign !.

The syntax is pretty simple:

result = !value;

|  |  |
| --- | --- |
| **b)**  **Ans** | **Write a C++ program to get Temperature in Fahrenheit *F* and then find the Atmosphere according to the below rules:**   * **If temperature *F* is above 40 degree Fahrenheit then display…………………..Very Hot.** * **If temperature *F* is between 35 & 40 degree Fahrenheit then display………Tolerable.** * **If temperature *F* is between 30 & 35 degree Fahrenheit then display………Warm.** * **If temperature *F* is less than 30 degree Fahrenheit then display………….……Cool.**   **#include <iostream>**  **Using namespace std;**  **int main() {**  **int f;**  **cout<<”enter the f:”;**  **cin>>f;**  **if(f<30){**  **cout<<”cool”;}**  **else if(f>30){**  **cout<<”warm”;}**  **else if(f>35)**  **{**  **cout<<”tolerate”;}**  **else if(f>4**  **{cout<<”very hot”;}**  **}** |

|  |  |  |
| --- | --- | --- |
| **Q.3**  **Ans:** |  | **What does *Looping* mean? Explain different loops in C++.**  A loop is used for executing a block of statements repeatedly until a particular condition is satisfied. For example, when you are displaying number from 1 to 100 you may want set the value of a variable to 1 and display it 100 times, increasing its value by 1 on each loop iteration.  In C++ we have three types of basic loops: for, [while](https://beginnersbook.com/2017/08/cpp-while-loop/) and [do-while](https://beginnersbook.com/2017/08/cpp-do-while-loop/). **For loop in C++:** Here in the loop initialization part I have set the value of variable i to 1, condition is i<=6 and on each loop iteration the value of i increments by 1.  **#include <iostream>**  **using namespace std;**  **int main(){**  **for(int i=1; i<=6; i++){**  **cout<<"Value of variable i is: "<<i<<endl;**  **}**  **return 0;**  **}** **While loop in C++:** loops are used for executing a block of program statements repeatedly until the given loop condition returns false.  **#include <iostream>**  **using namespace std;**  **int main(){**  **int i=1;**    **while(i<=6){**  **cout<<"Value of variable i is: "<<i<<endl; i++;**  **}**  **}** **do-while loop in C++:** do-while loop is similar to while loop, however there is a difference between them: In while loop, condition is evaluated first and then the statements inside loop body gets executed, on the other hand in do-while loop, statements inside do-while gets executed first and then the condition is evaluated.  **#include <iostream>**  **using namespace std;**  **int main(){**  **int num=1;**  **do{**  **cout<<"Value of num: "<<num<<endl;**  **num++;**  **}while(num<=6);**  **return 0;**  **}**   1. **Write a C++ program to read a number from keyboard and then determine whether it is Even or Odd number?**   **#include <iostream>**  **using namespace std;**  **bool checkEvenOdd(int num);**  **int main(){**  **int num;**  **bool isEven;**  **cout<<"Enter any number: ";**  **//Storing the entered value in variable num**  **cin>>num;**  **//Calling the function that checks even odd**  **isEven = checkEvenOdd(num);**  **if(isEven)**  **cout<<num<<" is an even number";**  **else**  **cout<<num<<" is an odd number";**    **return 0;**  **}**  **/\* This function checks whether the passed number is even**  **\* or odd. If the number is even then this function returns**  **\* true else it returns false.**  **\*/**  **bool checkEvenOdd(int num){**  **bool b;**  **/\* If number is perfectly divisible by 2 then it is**  **\* an even number else it is an odd number**  **\***  **\*/**  **if (num % 2 == 0)**  **b=true;**  **else**  **b=false;**  **return b;**  **}**  **Output:**      **Q4 a). What is the purpose of using break and continue statements?**  **Ans: Break Statements:**  The break statement has the following two usages in C++ −   * When the break statement is encountered inside a loop, the loop is immediately terminated and program control resumes at the next statement following the loop. * It can be used to terminate a case in the switch statement.   If you are using nested loops (i.e., one loop inside another loop), the break statement will stop the execution of the innermost loop and start executing the next line of code after the block.  **Continue Statements:**  The **continue statement** is used inside [loops](https://beginnersbook.com/2014/01/c-loops-examples/). When a continue statement is encountered inside a loop, control jumps to the beginning of the loop for next iteration, skipping the execution of statements inside the body of loop for the current iteration.  **b) Write a C++ program to find the sum of the following numbers:**  **1+2+3+………+10**  **Ans:**  #include <iostream>  using namespace std;  int main(){  int n, sum=0;  cout<<"Enter the sum of following numbers: ";  cin>>n;  if(n<=0){  cout<<"Invalid value of n";  }  else{  int i=1;  while(i<=n){  sum=sum+i;  i++;  }  cout<<"Sum of following numbers is: "<<sum;  }  return 0;  }  **Output:**    **Q 5: What is an array? Explain One- Dimensional and Two – Dimensional Arrays with examples.**  **Ans: Array:**  An **array** is a collection of variables that are of similar data types and are alluded by a common name. The main topic of our discussion is the difference between One-dimension and Two-Dimension array. A one-dimensional array is a list of variables with the same data type, whereas the two-Dimensional array is ‘array of arrays’ having similar data types.  A specific element in an array is accessed by a particular index of that array. Arrays in Java work differently as compared to C++. C++ do not have bound checking on arrays whereas, Java have strict bound checking on arrays.  **One-Dimensional** **Arrays**:  **One-Dimensional** or Single-Dimensional array is considered as the ”list of variables of similar data types”, and each variable can be distinctly accessed by specifying its index in square brackets preceded by the name of that array.1d array  In C++, the declaration of an array variable with the size is enough to allocate space for them in memory.  **Example:**  **#include <iostream> #include <conio.h> using namespace std; void main() { int val\_array[3]; int int\_val=1,counter; cout<<"Please enter three numbers that you want to multiply"<<endl; for(counter=0;counter<3;counter++) { cin>>val\_array[counter]; int\_val = int\_val\*val\_array[counter]; } cout<<"The multiplication of these three numbers is = "<<int\_val; getch();** }  **Output:**    **Two-Dimensional** **Arrays**:  C++ support multidimensional array. One of the simplest forms of a multidimensional array is a **two-dimensional array** or **2-D array**. A two-Dimensional array can be expressed as ‘array of arrays’ or ‘array of one-dimensional arrays’.  To declare the two-dimensional array variable, we have to specify the array name followed by two square brackets where the second index is the second set of square brackets.  A two-dimensional array is stored in the form of the row-column matrix, where the first index designates the row and second index shows the column. The second or the rightmost index of an array alters very fastly as compared to the first or left-most index while accessing the elements of an array.2d array    **Examples:** |

**#include<stdio.h>**

**int main(){**

**/\* 2D array declaration\*/**

**int disp[2][3];**

**/\*Counter variables for the loop\*/**

**int i, j;**

**for(i=0; i<2; i++) {**

**for(j=0;j<3;j++) {**

**printf("Enter value for disp[%d][%d]:", i, j);**

**scanf("%d", &disp[i][j]);**

**}**

**}**

**//Displaying array elements**

**printf("Two Dimensional array elements:\n");**

**for(i=0; i<2; i++) {**

**for(j=0;j<3;j++) {**

**printf("%d ", disp[i][j]);**

**if(j==2){**

**printf("\n");**

**}**

**}**

**}**

**return 0;**

**}**

**Output:**

