**SVKM’s NMIMS**

**Mukesh Patel School of Technology Management & Engineering**

**Computer Engineering Department**

Program: BTech Integrated Sem III

**Course: C Programming**

**2022-2023**

**Experiment No.04**

PART A

(PART A: TO BE REFFERED BY STUDENTS)

A.1 Aim: To study Type casting and decision statement

1. Write a C program to calculate square roots of quadratic equation
2. Write a C program to accept a float number and display the integer part using type casting technique.
3. Write a program to check whether an integer is positive negative or Zero
4. Write a program that determines if a year is a leap year.
5. Write a menu driven program to perform add / subtract / multiply / divide based on the users choice. The user will indicate the operation to be performed using the signs i.e. + for addition, - for subtraction and so on.
6. Program to find the largest of three numbers
7. Check whether given character is vowel or consonant.
8. Program to calculate Arithmetic Operations depending on operator

A.2 Prerequisite:

Basic decision statements 1 of C program like use of “If” , “ If and else “ ,nes ted “If and else” and “GOTO statements “.

Basic decision statement 2 and Basic program control 1 in C program like“**Switch statements** “

A.3 Outcome:

After successful completion of this experiment students will be able to

Develop and Execute C programs using decision statements

A.4 Theory :

**IF :**

Decision making structures require that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.



Syntax:

The syntax of an **if...else** statement in C programming language is:

if(boolean\_expression)

{

/\* statement(s) will execute if the boolean expression is true \*/

}

Next statements

Example:

#include <stdio.h>

void main ()

{

/\* local variable definition \*/

int a = 100;

/\* check the boolean condition \*/

if( a % 2 ==0 )

{

/\* if condition is true then print the following \*/

printf("a is a even number\n" );

}

printf("value of a is : %d\n", a);

}

**IF and ELSE**

An **if** statement can be followed by an optional **else** statement, which executes when the boolean expression is false.

Syntax:

The syntax of an **if...else** statement in C programming language is:

if(boolean\_expression)

{

/\* statement(s) will execute if the boolean expression is true \*/

}

else

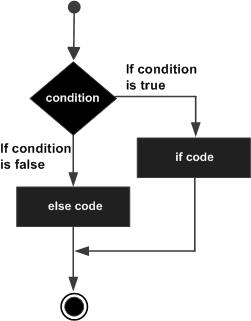
{

/\* statement(s) will execute if the boolean expression is false \*/

}

If the boolean expression evaluates to **true**, then the **if block** of code will be executed, otherwise **else block** of code will be executed.

C programming language assumes any **non-zero** and **non-null** values as **true**, and if it is either **zero** or**null**, then it is assumed as **false** value.



Example:

#include <stdio.h>

void main ()

{

/\* local variable definition \*/

int a = 100;

/\* check the boolean condition \*/

if( a < 20 )

{

/\* if condition is true then print the following \*/

printf("a is less than 20\n" );

}

else

{

/\* if condition is false then print the following \*/

printf("a is not less than 20\n" );

}

printf("value of a is : %d\n", a);

}

When the above code is compiled and executed, it produces the following result:

a is not less than 20;

value of a is : 100

The if...else if...else Statement

An **if** statement can be followed by an optional **else if...else** statement, which is very useful to test various conditions using single if...else if statement.

When using if , else if , else statements there are few points to keep in mind:

* An if can have zero or one else's and it must come after any else if's.
* An if can have zero to many else if's and they must come before the else.
* Once an else if succeeds, none of the remaining else if's or else's will be tested.

Syntax:

The syntax of an **if...else if...else** statement in C programming language is:

if(boolean\_expression 1)

{

/\* Executes when the boolean expression 1 is true \*/

}

else if( boolean\_expression 2)

{

/\* Executes when the boolean expression 2 is true \*/

}

else if( boolean\_expression 3)

{

/\* Executes when the boolean expression 3 is true \*/

}

else

{

/\* executes when the none of the above condition is true \*/

}

Example:

#include <stdio.h>

int main ()

{

/\* local variable definition \*/

int a = 100;

/\* check the boolean condition \*/

if( a == 10 )

{

/\* if condition is true then print the following \*/

printf("Value of a is 10\n" );

}

else if( a == 20 )

{

/\* if else if condition is true \*/

printf("Value of a is 20\n" );

}

else if( a == 30 )

{

/\* if else if condition is true \*/

printf("Value of a is 30\n" );

}

else

{

/\* if none of the conditions is true \*/

printf("None of the values is matching\n" );

}

printf("Exact value of a is: %d\n", a );

return 0;

}

When the above code is compiled and executed, it produces the following result:

None of the values is matching

Exact value of a is: 100

**GOTO Statements :**

A **goto** statement in C programming language provides an unconditional jump from the goto to a labeled statement in the same function.

**NOTE:** Use of **goto** statement is highly discouraged in any programming language because it makes difficult to trace the control flow of a program, making the program hard to understand and hard to modify. Any program that uses a goto can be rewritten so that it doesn't need the goto.

Syntax:

The syntax for a **goto** statement in C is as follows:

goto label;

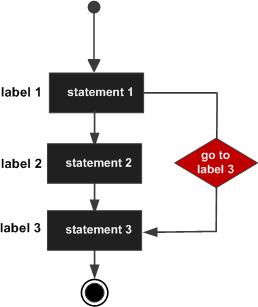
..

.

label: statement;

Here **label** can be any plain text except C keyword and it can be set anywhere in the C program above or below to **goto** statement.

Flow Diagram:



Example:

#include <stdio.h>

int main ()

{

/\* local variable definition \*/

int a = 10;

/\* do loop execution \*/

LOOP:do

{

if( a == 15)

{

/\* skip the iteration \*/

a = a + 1;

goto LOOP;

}

printf("value of a: %d\n", a);

a++;

}while( a < 20 );

return 0;

}

When the above code is compiled and executed, it produces the following result:

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 16

value of a: 17

value of a: 18

value of a: 19

**switch statement**

A **switch** statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each **switch case**.

Syntax:

The syntax for a **switch** statement in C programming language is as follows:

switch(expression){

case constant-expression :

statement(s);

break; /\* optional \*/

case constant-expression :

statement(s);

break; /\* optional \*/

/\* you can have any number of case statements \*/

default : /\* Optional \*/

statement(s);

}

The following rules apply to a **switch** statement:

* The **expression** used in a **switch** statement must have an integral or enumerated type, or be of a class type in which the class has a single conversion function to an integral or enumerated type.
* You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
* The **constant-expression** for a case must be the same data type as the variable in the switch, and it must be a constant or a literal.
* When the variable being switched on is equal to a case, the statements following that case will execute until a **break** statement is reached.
* When a **break** statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
* Not every case needs to contain a **break**. If no **break** appears, the flow of control will *fall through*to subsequent cases until a break is reached.
* A **switch** statement can have an optional **default** case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No**break** is needed in the default case.

Flow Diagram:



Example:

#include <stdio.h>

int main ()

{

/\* local variable definition \*/

char grade = 'B';

switch(grade)

{

case 'A' :

printf("Excellent!\n" );

break;

case 'B' :

case 'C' :

printf("Well done\n" );

break;

case 'D' :

printf("You passed\n" );

break;

case 'F' :

printf("Better try again\n" );

break;

default :

printf("Invalid grade\n" );

}

printf("Your grade is %c\n", grade );

return 0;

}

When the above code is compiled and executed, it produces the following result:

Well done

Your grade is B

**PART B**

(PART B: TO BE COMPLETED BY STUDENTS)

**(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)**

|  |  |
| --- | --- |
| Roll No. | Name: |
| Program: | Division: |
| Semester: | Batch : |
| Date of Experiment: | Date of Submission: |
| Grade : |  |

B.1 Algorithm

1.

2.

3.

B.2 Flow Chart

1.

2.

3.

B.3 Program Code

1.

2.

3.

B.4 Input-Output

1.

2.

3.

B.5 Conclusion:

*(****Students must write the conclusion as per the attainment of individual outcome listed above and learning/observation noted in section B.3)***