**Chapter 3**

**CONTROL STATEMENTS**

***LEARNING OBJECTIVES***

*After going through this chapter, the readers will be able to:*

*Learn different types of statements in C.* 

*Develop algorithms and flow charts for simple engineering and mathematical problem solving.*

*Write programs using if, if-else, nested if, nested if-else and else-if ladder.* 

*Write programs using the looping structures while, do-while & for.*

*Learn usage of Jumping Statements:* ***break, continue & goto*** *in the programs.*

**3.1 INTRODUCTION**

Control Statements determine the “flow of control” in a program and enable to specify the order in which the various instructions in a program are to be executed by the computer. Normally high level procedural programming languages require three basic control instructions.

Sequential control instructions 

Decision and Selection control instructions

Repetition or Loop or Iterative control instructions 

Jumping statements

The common thing shared by all the programs written so far is that in each of the programs, all the statements from the first statement till the last statement get executed without fail in a serial manner. That is, one after the other. This kind of execution of statements in a program is called **Sequential Execution**.

The programming circumstances require selecting some statements for execution if some condition is satisfied; and skipping the block of statements if the condition is not satisfied, thereby, resulting in a change in the order of execution of statements. To be precise, selection of some statements for execution depends on whether a condition is true or false. This kind of execution of statements is called

**Conditional Execution or Selection**.

Suppose, if there is a need for a group of statements to be executed repeatedly until some logical condition is satisfied, then looping is required in the program. This can be carried out using various L**oop or Repetitive or Iterative** control statements.

The **Jumping statements** are used to exit from loop or to go to the next repetition of the loop after skipping the remaining statements of the loop or to transfer control from one part of the program to another part of the program unconditionally.

The sequential control instruction ensures that the instructions are executed in the same order in which they appear in the program. Decision and selection control instructions allow the computer to take decision as to which instruction is to be executed next. The loop control instruction helps computer to execute a group of statements repeatedly.

Decision and Selection control instructions allows a program to take different courses of action depending on different conditions. C provides two selection structures.

**if** 

**switch**

***Control Statements* 51**

Repetition or loop control instruction allows a program to execute the same section of code more than once. A section of code may either be executed a fixed number of times, or while some condition is true. C provides three looping statements.

**while**

**for** 

**do-while**

C provides three jumping statements

**break**

**continue** 

**goto**

**3.2 SIMPLE & COMPUND STATEMENTS**

**3.2.1 Simple statements**

A single statement is called a simple statement.

**Assignment**: A = A + 1; 

**Input**: scanf(“%d”,&x); 

**goto**: goto begin; 

The C language uses semicolons as statement terminators. A semicolon follows every simple (non compound) statement.

**Example3.1:** if(x==y)

printf(“x and y are equal”);

If the values of the variables x and y are same, then only the printf() statement gets executed. Otherwise, it is skipped.

**3.2.2 Compound statements**

A set of simple or compound statements enclosed within a pair of opening and closing braces is called a Compound Statement.

The terms compound statement and block both refer to a collection of statements that are enclosed in braces to form a single unit. Compound statements have the form

**Example3.2:** {

printf(“….”);

scanf(“%d”,&a);

if(a>max)

max=a;

}

**Example3.3:** if(flag==1)

{

printf(“…”);

scanf(“%d”,&b);

}

***Control Statements* 52**

**3.3 NULL & EXPRESSIONS STATEMENTS**

**3.3.1 Null Statement**

A "null statement" is an executable statement containing only a semicolon; it can appear wherever a statement is expected. Nothing happens when a null statement is executed. The null statement is:

Statements such as **do**, **for**, **if**, and **while** require that an executable statement appear as the statement body. The null statement satisfies the syntax requirement in cases that do not need a substantial statement body.

**;**

This example illustrates the null statement:

**Example3.4:** for(i=0;i<10;line[i++]=0) ;

In this example, the loop expression of the **for** statement line[i++] = 0 initializes the first 10 elements of line to 0.

**3.3.2 Expression-statement**

**expression;**

All side effects from the expression evaluation are completed before the next statement is executed. The following are the expression statements.

**Example3.5:** x=(y+3) /\* x is assigned the value of y+3 \*/

x++; /\* x is incremented \*/

x=y=0; /\* both a and y are initialized to 0 \*/

proc(arg1,arg2); /\* function call returning void \*/

y=z=(f(x)+3); /\* a function-call expression \*/

**3.4 DECISION-MAKING AND BRANCHING (SELECTION)**

i) **if-statement**

ii) **switch-statement**

**3.4.1 if-statements**

The different forms of if- Statements are

i) The simple-if statement.

ii) The if-else Statement.

iii) The Nested if- else Statement.

iv) The else-if Ladder.

**3.4.1.1 The Simple if statement**

It is used to execute an instruction or block of instructions only if a condition is fulfilled.

***Control Statements* 53**

The syntax is as follows:

Condition

**False**

**if(condition)** ?

**True**

**statement;**

Statement

**Exit**

The above flow chart reflects the logical flow of the if statement. Where condition is the expression that is to be evaluated .If this condition is true (non-zero), statement is executed. If it is false(zero), statement is not executed and the program continues to the next instruction after the conditional statement.

**Example3.6:** if(a==b)

printf(“a and b are equal”);

if the values a and b are equal, then only the printf() statement gets executed. Otherwise, it is skipped. **Example3.7:** if((a>b)&&(a>c))

printf(“a is the largest”);

If the value of the variable a is greater than the values of both b & c, then only the printf () statement gets executed. Otherwise, the statement is skipped.

**Example3.8:** if(n%2)

printf(“n is odd”);

If the value of the variable n is not divisible by 2, i.e. if the remainder after division of n by 2 is 1, then only printf () statement gets executed. Otherwise, the statement gets skipped.

**Example3.9:** if(a>b)

{

t=a;

a=b;

b=t;

}

Only when the value of a is greater than the value of b the statements within the pair of braces gets executed. Otherwise, the statements are skipped.

**3.4.1.2 The if-else statement**

If-else structure acts as a selector of one block of statements out of two blocks.

The syntax of the if-else statement is as follows:

***Control Statements* 54**

if ( test-expression) statement 1;

else

statement 2;

statement x;

True

False True

**Test**

**Expression**

Statement2 Statement1 Statements

The above flow chart reflects the logical flow of the if-else statement.

If test-expression evaluates to true, statement 1 will be executed, otherwise, statement2 will be executed. Then the control is transferred to statement x.

The statement1 is called if-block and the statement2 flowing else is called else-block statement.

**Example3.10:** if(a>b)

printf(“a is larger than b”);

else

printf(“b is larger than a”);

Here if a is greater than b the message “a is larger than b“is displayed on the screen. Otherwise,” b is larger than a” is displayed.

**Example3.11:** if(basic>5000)

{

da=basic\*65;

hra=basic\*15;

cca=200;

}

else

{

da=basic\*57;

hra=basic\*12;

cca=100;

}

Here, if the basic is greater than Rs.5000 then da and hra are calculated to be 65% and 15% of basic respectively and the cca is assigned Rs.200. otherwise, i.e. the basic is less than or equal to 5000 than da and hra are calculated to be 57% and 12% of basic respectively and cca is assigned Rs.150.

**Program3.1Program to illustrate if-else statement.**

#include<stdio.h>

#include<conio.h>

int main(void)

{

int year, r4, r100,r400;

printf (“Enter a year\n”);

scanf (“%d”,&year);

***Control Statements* 55**

r4=year % 4;

r100=year % 100;

r400=year % 400;

if (( r4 == 0) && (r100!=0) || (r400 == 0)) printf (“%d is a leap year” ,year); else

printf (“%d is not a leap year” ,year); }

**3.4.1.3 The Nested if-else Statement**

Input-Output:

Enter a year

4000

4000 is a leap year Enter a year

2011

2011 is not a leap year

We know that if execution of a block of statements is conditional, we can use one if statement. What if the conditional execution of the block of statements is itself conditional? Here, we need to enclose one if statement within another if statement. On the similar lines, we can even enclose an if-else within another if-else. If one if-else is enclosed within another if-else, the resulting structure is called nested if – else.

The syntax of nested if-else is as follows.

if (test-expression 1 ) {

if(test-expression 2) {

statement-1;

}

else

{

statement-2;

}

**False True**

**Test-expr**

**False**

**Statement Test Expression**

**True**

}

else

{

statement -3; }

**Statement Statement**

Here, one if-else is enclosed within another if-else. The if structure, which encloses another is called outer-if. The if structure, which is enclosed within another is called inner-if. Statement-1, statement-2 and statement-3 may be simple or block of statements. First, test-expression1 is checked. If it evaluates to true, test-expression2 is checked. If test-expression2 also evaluates to true, then else-block of the inner-if, statement-1 would get executed otherwise, statement-2 would be evaluated. If test-expression1 itself evaluates to false, then else-block of the outer-if, statement-3 would get executed. Thus, this variation acts as a selector of one out of three blocks of statements.

**Example3.12:**The following program segment compares a and b.

if(a>=b)

{

if(a>b)

***Control Statements* 56**

printf(“a is larger than b”);

else

printf(“a and b are equal”);

}

else

printf(“a is less than b”);

**Example3.13:** The following segment compares three values a ,b and c and collects the largest of them in the variable largest.

if( a>=b)

{

if(a>c)

largest = a;

else

largest = c;

}

else

{

if(b>c)

largest = b;

else

largest = c;

}

**3.4.1.4 The else-if Ladder**

The else-if ladder helps select one out of many alternatives blocks of statements for execution depending on the mutually exclusive conditions. The syntax of the else-if ladder is as follows.

**False**

if(test-expression 1) Statement-1;

else if (test-expression2)

**True**

**T**

**T**

**False**

Statement-2;

else if(test-expression3) Statement-3;

:

**T** - Test expression

:

else if (test-expression n) Statement -n;

else

**statements**

**True False**

**T**

**statements**

**True**

**statements**

**statements**

Here test-expression1, test-expression2… test-expression-n are mutually exclusive. That is only one test

Statement

expression of all will be true. There are n+1 of statements. Initially, test-expression1 is checked. If it is n+1;

statement x

***Control Statements* 57**

true, then statement-1 would get executed; all the other blocks of statements would be skipped; then the statement x gets executed. If test-expression1 is false, test-expression2 is checked. If it is true, then statements-2 would be executed; all the other statements would be skipped. If test-expressions 2 is false, test-expression3 is checked. If it is true, statements-3 would get executed; all the other statements would be skipped. This is continued. In general, ith test-expressions is checked only when the first i-1 test

expressions evaluate to false. If none of the expressions is found to be true, the last statement would get executed. Thus, else-if ladder act as a selector of one out of n+1 blocks of statement.

**Example3.14*:*** printf (“Enter a number\n”);

scanf(“%d”,&n);

if(n<=10)

printf(“ n is less than or equal to 10”);

else if (n <=20)

printf(“ n lies between 11 and 20”);

else if (n<=30)

printf (“ n lies between 21 and 30”);

else

printf(“ n is greater than30”);

**Program3.2** To find whether a number is positive ,negative or 0.

void main()

{

int n;

printf (“Enter a number\n‟); scanf (“%d”,&n);

if(n>0)

printf (“%d is positive”,n); else if (number >0)

printf (“%d is negative”,n); else

printf (“Zero”);

getch();

}

**3.4.2 The Switch Statement**

Input-Output

Enter a number

78

78 is positive number.

Switch statements provide a non-iterative choice between any numbers of paths based on specified conditions. They compare an expression to a set of constant values. Selected statements are then executed depending on which value, if any, matches the expression. Switch statements have the form

Here, expression is an integer or character expression. It evaluates to either integer or a character.Value1, value 2…. Value N is the case values. The expression of the switch statement can take any of the values. If it matches with value1, statement-1 will get executed and the break statement following it causes the skipping of the switch structure. If the expression matches with value2, then statement2 will get executed. If the value of expression matches with none of the case values then default option will get executed.

***Control Statements* 58**

switch ( expression)

{

case value1:statement-1;

break;

case value2:statement -2;

break;

.

.

case value n:statement -n;

break;

default :default statement;

break;

}

**Points to Remember:**

**Test-expr**

**Statements1 Case1**

**Statements1 Case2**

**Statements1 Case3**

**Statements1 default**

i) The case values should not be float values or Boolean expression.

ii) The case values value1,value2,… value n should be distinct.

iii) The order of their presence is immaterial.

iv) default case is optional.

**Example 3.15:** scanf (“%d”,&n);

switch (n)

{

case 1:printf (“one”);

break;

case 2:printf (“two”);

break;

default : printf (“Other than one or two \n”);

}

If the value of n is 1, then the string “one” is displayed. If the value of n is 2,then the string “two” is displayed.

**Program3.3** Program to accept a digit and display it in words.

void main()

{

int digit;

printf (“Enter a number\n”); scanf (“%d”,&digit);

switch (digit)

{

case 0 : Printf( “Zero”); break; case 1 : Printf( “One”); break; case 2 : Printf( “Two”); break; case 3 : Printf( “Three”); break;

case 4 : Printf( “Four”); break; case 5 : Printf( “Five”); break; case 6 : Printf( “Six”); break; case 7 : Printf( “seven”); break; case 8 : Printf( “Eight”); break;

Input-Output Enter a digit 7

seven

***Control Statements* 59**

case 9 : Printf( “Nine”); break;

default : Printf( “Not a digit ”); break;

}

getch();

}

3.5 **Looping or Iterative or Repetitive Statements in C**

The repetition of execution of a block of statements as long as some condition is true is called Looping. Looping is also called as Iteration.

There are three kinds of iterative statements.

i) **while-loop**

ii) **for-loop**

iii) **do-while loop**

**3.5.1 The while-loop**

When in a program a single statement or a certain group of statements are to be executed repeatedly depending upon certain test condition, then while statement is used

The general form of**while loop** is as follows:

while (expression)

Statement;

The statement may be simple or compound statement. The keyword “while” is followed by expression enclosed within a pair of parenthesis. The expression can be an arithmetic expression, relational expression or a logical expression. The expression is then followed by a set of one or more statements, which are expected to be repeatedly executed.

The execution sequence effected by while looping construct is best illustrated by the following flow chart segment.

**Initialization**

**False**

**Test**

**Expression**

**True**

**Statements**

**Example3.16:** i=1;

sum =0;

while (i<=10)

{

sum+=i;

i++;

}

***Control Statements* 60**

The purpose of the code is to find the sum of the first 10 natural numbers

**Example3.17:**while ((ch=getchar())!=‟\n‟)

{

printf (“%c\n”,toupper(ch));

}

The segment code enables us to accept characters of a line, one at a time till the new line character is entered. Each character entered is converted to upper case and displayed.

After getting familiarized with the syntax and examples of the usage of the while loop, we will now write some programs which make use of it.

**Program3.4 Program to find the sum of the digits of an integer**

void main()

{

int rem,sum;

long int n,temp;

printf (“Enter an integer”);

scanf (“%ld”,&n);

temp=n;

sum=0;

while (n>0)

{

rem =n%10;

sum+=rem;

n/=10;

}

printf (“sum of the digits of ld=%d\n”,temp,sum); getch();

}

**3.5.2 The for loop**

The syntax of for loop is as follows:

Input – Output:

Enter an integer

123

Sum of the digits of 123= 6.

for (expr1; expr2; expr3)

statement;

The statement may be simple or compound statement:

i) expr1 is initialization expression

ii) expr2 is test-expression or conditional expression

iii) expr3 is increment/decrement expression.

Statement is executed repeatedly until the value of expr2 is 0. Before the first iteration, expr1 is evaluated and then expr2 is evaluated. If expr2 is non-zero then only control enter the loop .expr1 is usually used to initialize variables for the loop. After each iteration of the loop, expr3 is evaluated. This is usually used to increment a loop counter. In fact, the for-loop is absolutely equivalent to the following sequence of

statements:

expr1;

while (expr2) {

statement; expr3;

}

***Control Statements* 61**

That's why expr1 and expr3 must contain side effects, else they are useless.

**Example3.18:** for (i=0; i<100; i++)

printf(“%d\t”,i);

All the expressions are optional. If expr2 is left out, it is assumed to be 1. Statement may be a compound statement as well.

The execution sequence effected by for looping construct is best illustrated by the following flow chart segment.

Expression 1

False

Expression 2

TRUE

Statement

Expression 3

**The execution sequence effected by for loop is as follows:**

i) Initialization expression is executed.

ii) Test-expression is evaluated.

iii) If the test-expression evaluates to true, the statements in the body of the loop would get executed. iv) Control goes to the increment/decrement expression, and it is executed.

v) Test-expression is again evaluated.

vi) If it evaluates to true, again the statements in the body of the loop would get executed. vii) Control once again goes back to the increment/decrement expression, which updates the looping variable.

viii) If the test-expression evaluates to false execution of the body of the loop is terminated. **Example3.19:** for (j=1;j<=10;j++)

printf(“%d\n”,j);

here the for loop prints the first 10 natural numbers.

**Program3.5** Program to find largest of n numbers

#include<stdio.h>

void main()

{

int n,i;

float x,large;

printf ( “Enter the number of values “);

scanf(“%d”,&n);

printf(“Enter the first number”);

***Control Statements* 62**

scanf(“%f”,&x);

large=x;

printf(“Enter the reaming numbers”);

for ( i=1; i <n; ++i)

{

scanf(“%f”,&x);

if(x>large)

large=x;

}

printf(“Largest of given numbers =%f\n”,large); getch();

}

Input-output:

Enter the number of values 6 Enter the first number 10 Enter the reaming numbers -3

23

78

-220

5

Largest of given numbers=78

**Program3.6*:*** Program to generate multiplication table of a number #include<stdio.h>

#include<conio.h>

void main()

{

int num,i,product;

printf( “\n Enter a number:”);

scanf(“%d”,&num);

printf(“The multiplication table of %d is:”,num); for( i=1;i<=10; i++)

{

product= num\*i;

printf( “%d \* %d = %d \n”,num,i,product); }

getch();

}

Input –Output:

Enter a number: 6

The multiplication of 6 is: 6 \* 1 =6

6 \* 2=12

.

.

.

6 \* 10 = 60

As mentioned earlier a for loop can be written without any one of the expressions or two expressions or all the three.

**Example3.20:**for(sum=0;n>0;)

{

digit=n/10;

sum+=digit;

n/=10;

}

Here the segment is to find the sum of digits of a given integer. The loop is written without expression3. **Program3.7** Program to find gcd of two integers using Euclidian algorithm

#include<stdio.h>

void main()

{

int a,b,rem,temp1,temp2; printf(“Enter two integers”); scanf(“%d%d”,&a,&b); temp1=a; temp2=b;

for(;(r=a%b)>0;)

{

a=b;

Input-output:

Enter two integers 36 24 gcd(36,24)=12

***Control Statements* 63**

b=r;

}

printf(“gcd(%d,%d)=%d\n”,temp1,temp2,b);

}

In the above program the loop is written without expression1 and expression3.

The for loop can be written without any expressions as like

for( ; ; )

{

statements block;

}

Would cause that statement block to be executed indefinitely.

**3.5.3 do-while Loop**

The while and for loops test the termination condition at the top. By contrast, the third loop in C, the do-while, tests at the end after making each pass through the loop body. Hence the body is always executed at least once.

Thus while & for loops are entry-controlled loops, whereas do-while is exit controlled loop. The general form of do*-while* loop:

Initialization

**do {**

**statements;**

Statements

**}while (expression); false**

Test

Expression

True

the statement block is executed, and then expression is evaluated. If it is true, statement block is evaluated again, and so on. When the expression becomes false, the loop terminates.

**Example3.21:** i=1;

sum=0;

do

{

sum+=i;

i++;

}while ( i<= 10);

***Control Statements* 64**

The statements in the body of the loop get repeatedly executed as long as i< =10 and it finds the sum of the first 10 natural numbers.

**Example3.22:** do

{

scanf(“%d”, &n);

printf(“Do you want to continue? Enter Y or N”);

ch = getchar();

} while ( ch ==‟Y‟);

The above segment of code enables us to keep accepting integer value into the variable n as long as we enter „Y‟ to the variable ch. Once we input „n‟ response to the statement ch = getchar (); the loop is exited.

**Program3.8** Program to print first 5 natural numbers using do-while.

#include<stdio.h>

void main()

{

int x=5;

int i=0;

//using do while loop statement do{

i++;

printf (“%d\n”,i);

}while(i<x);}

**Comparison of the three loops**

Output: 1

2

3

4

5

| while | for | do - - while |
| --- | --- | --- |
| n=1;  while (n<=10)  {  - - -  - - -  n++;  } | for(n=1; n<=10; n++)  {  - - -  - - -  } | n=1;  do  {  - - -  - - -  n++;  }  while (n<=10); |

**3.6 Nested control structures**

Like the method of if-else structures, loops also can be nested. In nested loops one loop should be completely embedded within the other. There can be no overlap. The inner and outer loop structures can be different.

***Control Statements* 65**

for(i=1;i<n;++i) {

. . . . .

. . . . .

while( expression) {

……..

……….

}

for(i=1;i<n;++i) {

. . . . . . .

. . . . . . .

for(j=1;j<n;j++) {

. . . . .

. . . . .

}

. . . .

}

**Program 3.9** Program to print all prime numbers between two limits. #include<stdio.h>

#include<math.h>

void main()

{

int ll,ul,k,n,r;

printf(“Enter the lower and upper limits\n”);

scanf (“%d%d”,&ll,&ul);

printf (“The prime number between %d and %d are\n”,ll,ul); for( n=ll;n<=ul;++n)

{

k=sqrt (n);

for (i=2;i<=k;++i)

{

r=n%i;

if(r==0)

break;

}//end of inner for loop

if (i>k)

printf (“%d\n”,n);

}//end of outer loop

}// end of main

Input-output:

Enter the lower and upper limits10 20

The prime numbers between 10 and 20 are

11

13

17

19

Inner for loop is written to check n is prime or not. Outer loop is written to execute this loop repeatedly for n=ll to ul.

**3.7 JUMP STATEMENTS**

Jump statements transfer control unconditionally from one part of the program to another part of the program.

i) goto identifier

ii) continue

iii) break

**3.7.1 goto statement**

So far we have discussed ways of controlling the flow of execution based on certain specified conditions. Like many other languages, C supports goto statement to branch unconditionally from one point to another in the program. Although it may not be essential to use goto statement in a highly structured language like C, there may be occasions when the use of goto might be desirable.

***Control Statements* 66**

The goto require a label in order to identify the place where the branch is to be made. A label is any valid variable name, and must be followed by a colon. The label is placed immediately before the statement where the control is to be transferred. The general forms of goto and label statements are shown below.

goto label; - - - - - - - - - - - - - - - - - - - - - - - -

label :

statement; - - - - - -

- - - - - -

- - - - - -

goto label :

Forward jump Backward jump

label :

statement;

**The label c**an be anywhere in the program either before or after the goto label; statement. During running of a program when a statement like

**goto begin;**

is met, the flow of control will jump to the statement immediately following the label begin:. This happens unconditionally.

Note that a goto breaks the normal sequential execution of the program. If the label: is before the statement go to label; a loop will be formed and some statements will be executed repeatedly. Such a jump is known as a backward jump. On the other hand, if the label: is placed after the go to label; some statements will be skipped and the jump is known as forward jump.

A goto is often used at the end of a program to direct the control to go to the input statement, to read further data.

**Example3.23:**double x, y;

read :

scanf(“%d”, &x);

if (x<0)

goto read;

y=sqrt(x);

printf(“%f %f”, x,y);

goto read;

The above example evaluates the square root of a series of numbers read from the terminal. The program uses two goto statements, one at the end, after printing the results to transfer the control back to the input statement and other to skip any further computation when the number is negative.

Due to the unconditional goto statement at the end, the control is always transferred back to the input statement. In fact, this program puts the computer in a permanent loop known as an infinite loop. The

***Control Statements* 67**

computer goes round and round until we take some special steps to terminate the loop-such infinite loops should be avoided.

**3.7.2 Break Statement**

Sometimes, it is required to jump out of a loop irrespective of the conditional test value. Break statement is used inside any loop to allow the control jump to the immediate statement following the loop. The syntax is as follows:

break;

When nested loops are used, then break jumps the control from the loop where it has been used. Break statement can be used inside any loop i.e., while, do-while, for and also in switch statement.

Let us consider a program to illustrate break statement.

**Program3.10** Program to calculate the first smallest non-trivial divisor of a number. #include<stdio.h>

void main ( )

{

int div, nun, i;

printf (“Enter any number”); scanf (“%d”, &nun);

for (i=2; i<=nun; ++i)

{

if (nun%i) = = 0)

{

Input-output:

Enter any number: 9

Smallest divisor for the number 9 is 3

printf (“smallest divisor for the number %d is %d”, nun, i);

break;

}

}

}

In the above program, we divide the input number with the integer starting from 2 onwards, and print the smallest divisor as soon as remainder comes out to be zero. Since we are only interested in first smallest divisor and not all divisors of a given number, so we come out of the for loop using break statement without going further for the next iteration of for loop.

**3.7.3 Continue statement**

Unlike break statement, which is used to jump the control out of the loop, it is sometimes required to skip some part of the loop and to continue the execution with next loop iteration. Continue statement used inside the loop helps to bypass the section of a loop and passes the control to the beginning of the loop to continue the execution with the next loop iteration. The syntax is as follows:

**continue;**

Let us see the Program 3.1 given below to know the working of the continue statement.

**Program3.11** Program to print first 20 natural numbers skipping the numbers divisible by 5. #include<stdio.h>

void main ( )

{

int i ;

***Control Statements* 68**

for( i=1, i<=20; i++; {

if(i%5)= = 0)

continue;

printf(“%d”,i);

}

}

Output:

1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19

Here, the printf statement is by passed each time when value stored in i is divisible by 5.

**Program3.12** Program to find sum of all non-negative integers in a list of n integers entered through key board

void main()

{

int i,sum=0,n,x;

printf(“Enter number of values”); scanf(“%d”,&n);

printf(“Enter the values”);

for( i=1;i<=n;i++)

{

scanf (“%d”,&x) ;

if( n <0)

continue;

sum+ = n;

}

printf(“sm=%d\n”,sum);

}

**SUMMARY**

Input-output:

Enter the number of values 5

Enter the values

2

-12

5

0

-4

sum= 7

The program statements in C fall into three general types: assignment, input/output, and control. C has two types of control structures: selection (decision) and a repetition (loops). The decision control constructs are of two types: conditional and unconditional. The conditional control constructs are if, if else, if-else-if and switch. The unconditional control constructs are break, continue and goto. The loop control constructs are for, while and do-while. 

One-way decisions are handled with an if statement that either does some particular thing or does nothing at all. The decision is based on a test expression that either valuates to true or false. Two way decisions are handled with if-else statements that either do one particular thing or do another. Similar to one-way decisions, the decision is based on a test expression. Multi-way decision statements use if-else-if ladder or nested ifs, or switch statements. They are all used to evaluate a test expression that can have several possible values selecting different actions. 

In a program by using loop control statements while, for, do-while you can execute set of statements repeatedly, while and for are called entry-controlled loop which means that testing a condition is checked before starting a loop. Do-while is called as exit-controlled loop in which the testing a condition is checked after executing the loop. 

***Control Statements* 69**

Using break statement, we can leave a loop even if the test condition for its end is not fulfilled. It can be used to end an infinite loop, or to force it to end before its natural end. The continue statement causes the program to skip the rest of the loop in the present iteration as if the end of the statement block would have reached, causing it to jump to the next jump iteration. 

Using goto statement, we can make an absolute jumpto another part of the program. You should use this feature carefully since its execution ignores any type of nesting limitation. 

**EXERCISES**

**Multiple Choice Questions**

3.1 What would be the output of each of the following code segments?

i) void main ( )

{

int a=1, b=1;

if(a = = 1)

if(b = = 0)

printf(“HI”);

else

printf(“Bye”);

}

ii) void main ( )

{

int a,b=0;

if(a=b=1)

printf (“hello”);

else

printf (“world”);

}

iii) void main()

{

int var1,var2=2,num=100,a;

if(var1=var2%2)

num=2;

a=2;

printf(“%d %d”,num,var1);

}

3.2 What is the value of y in the following code?

void main()

{

int x=7,y=0;

if(x=6)

y=7;

else

y=1;

}

a)7 b)0 c)1 d)6

3.3 What is the value of z after the following statements executed?

void main()

{int x=2,y=2,z=1;

if(x=y%2)

***Control Statements* 70**

z+=10;

else

z+=20;

}

a)21 b)11 c)1 d)the expression x=y%2 is not allowed

3.4 Write the following expression using if statement.

x = ( y > 3 ) ? ( x < y ) ? y : x + 10 : ( y < 0 ) ? -y : y + 10;

3.5 What is the output of the following code?

main()

{

int a=500,b,c;

if(a>400)

{

b=300;

c=200;

}

c=100;

printf(“\n b=%d , c=%d \n”,b,c);

}

3.6 Which of the following is true?

int a=5,b=7;

if(a<b);

a++;

else

b++;

printf(“a=%d,b=%d \n”,a,b);

a)a=6,b=8 b)a=6,b=7 c)a=5,b=8 d)syntax error.

3.7 Which of the following is the correct output for the program given bellow? #include<stdio.h>

int main()

{

float a=0.7;

if(0.7>a)

printf(“hi \n”);

else

printf(“hello \n”);

return 0;

}

a)Hi b) hello c)hi hello d)None of the above

3.8 Which of the following statements are correct about the problem given below? #include<stdio.h>

main()

{

float a = 2.8, b = 2.87 ;

if (a = b)

printf(“ a and b are equal \n”);

else

printf(“a nd b are not equal \n”);

}

***Control Statements* 71**

a)The statement if (a = b) results in a compilation error.

b)floats cannot be compared using if

c)conditional operator should be used to compare floats

d)switch\_case should be used to compare floats.

3.9 What is the output when the following code is executed?

#include<stdio.h>

void main()

{

int a=0,b=1;

if(a<b || ++b)

a=5;

else

b=6;

printf(“a=%d ,b=%d \n” a,b);

}

3.10 What is the output when the following code is executed?

#include<stdio.h>

void main( )

{

int x=1,y=3,z=0;

switch(x)

{

case 0: x=2;

y=3;

break;

case 1: x=4;

break;

default : ++x;

}

printf (“x=%d,y=%d, z=%d”,x,y,z);

}

3.11 Originally x=4,y=3 and z=0 . What are values of x,y,z after executing the following code? switch(x)

{

case 0 : x=2;y=3;

case 1 : z=4; break;

default : ++x;

}

a)x=0,y=3,z=4 b)x=2,y=3,z=4 c)x=1,y=3,z=4 d)x=5,y=3,z=0

3.12 When is default statement executed in switch case construct?

a) When ever there is exactly one match.

b)When ever break statement is omitted in all case statements

c)When ever there is no match with case labels.

d)options b and c.

3.13 Consider the following program segment

int n sum=1;

switch(n)

{

case 2: sum=sum+2;

case 3: sum\*=2;

***Control Statements* 72**

break;

default : sum=0;

}

If n=2 ,What is the value of sum

a)0 b)6 c)3d)None of these

3.14 What will be printed by the following code?

#include< stdio.h>

main()

{

int x = 5;

switch(x);

{

case 5 : printf (“%d\t”, x \* x) ;

case 6 : printf(“%d\t”, x + 1 \* x+1);

default : break;

}

}

3.15 Omitting the break statement from a particular case in switch case construct a) leads a syntax error

b)causes execution of the program to terminate after the case c) causes execution to continue all subsequent cases. d) causes execution to branch to the statement after the switch statement 3.16 How many times of the while loop in the following program will be excuted. void main()

{

int j=1;

while(j<=255);

{

printf(“%d\n”,j);

j++;

}

}

3.17 What is the output when the following code is executed? void main( )

{

int i=3;

while(i)

{

int x=100;

printf(“\n%d %d”, i, x);

x=x+1;

i=i+1;

}

}

3.18 What is the output when the following code is executed? void main()

{

char j=1;

while(j<=255)

{

***Control Statements* 73**

printf(“%d\n”,j);

j=j+1;

}

}

3.19 What is the output when the following program is executed? #include<stdio.h>

main()

{

int i;

while(i<=10);

{

printf(“%d”,i);

++i;

}

}

a) First 10 natural numbers can be displayed b) 11 c) Natural numbers starting from 1 are displayed indefinitely d) no output. 3.20 What is the output when the following program is executed? #include<stdio.h>

main()

{

int a=5;

while(a-- >0)

printf(“%d\t”,a);

}

3.21 What is the output when the following program is executed? #include<stdio.h>

main()

{

int p=5;

while(p<8)

{

printf(“%d”,p);

p=9;

}

}

a)5 6 7 b)5 c)5 9 9 9 9 … d) no output

3.22 What is the output when the following program is executed? #include<stdio.h>

main()

{

int x=1;

while(1)

{

printf(“%d”,x);

++x;

}

}

3.23 What will be the output when the following code is executed? #include<stdio.h>

main()

***Control Statements* 74**

{

int a = 6;

while (a)

{

printf(“%d\t”, a);

a == 2;

}

}

3.24 What will be the output when the following code is executed? void main()

{

int i;

for(i=0; j<=10; i++, printf(“%d \t”, i));

}

3.25 What will be the output when the following code is executed? void main( )

{

int i;

for(i=0; i<3; i++)

printf(“%d\t”, i);

}

3.26 What will be the output when the following code is executed? void main()

{

int a;

for(a=1;a<=32767;a++)

printf(“%d”,a);

}

3.27 What will be the output when the following code is executed? void main()

{

int i=0;

for(;i++;)

printf(“%d”,i);

}

3.28 Which of the following statement is used to exit from a switch case statement?

a) exit b)continue c)goto d)break

3.29 How many times KLU is printed?

for(i=0;i>5;++i)

printf(“KLU\n”);

a)6 b)5c)0d)Indefinite no of times.

3.30 Write the syntax of for statement in C?

3.31 Which of the following is a loop construct?

a)if-else b)switch-case c)goto d)while

3.32 What is the output of the following program?

main()

{

unsigned int i;

for(i=10;i>=0;--i)

printf(“%d”,i);

***Control Statements* 75**

}

a) prints numbers from 10 to 0 b) prints numbers from 10 to 1 c) prints numbers from 10 to -1 d)An indefinite loop is formed 3.33 How many times x is printed?

for(i=0,j=10;i<j;++i,--j)

printf(“x”);;

a)10 b)5 c)4 d)none

3.34 what is the value of a after the following code is executed? void main()

{

int b,a=0;

for (b=0;b<10;b++)

{

switch(b)

{

case 0:

case 1:

case 2:

case 5:

++a;

case 3: break;

case 4: break;

default: break;

}

}

3.35 What is the output when the following program is executed? #include<stdio.h>

main()

{

int i;

for(i=1;i<10;++i)

printf(“%d”,i);

}

a) First 10 natural numbers can be displayed b) 11 c) Natural numbers starting from 1 are displayed indefinitely d) no output. 3.36 What is the output when the following program is executed? #include<stdio.h>

main()

{

int a=5;

for(a=5;a-- >0; )

printf(“%d\t”,a);

}

3.37 What is the output when the following program is executed? #include<stdio.h>

main()

{

int a;

for(a=10;! i;-- i)

printf(“%d”,i);

}

***Control Statements* 76**

a) First 10 natural numbers are printed in reverse order b)11 c) 10 d) no output.

What is the output when the following program is executed? #include<stdio.h>

main()

{

int b;

for(b=3;b<10;b=7)

printf(“%d”,b);

}

a)3 4 5 6 7 8 9 b)3 7 8 c) 3 7 7 7 7 … d)3

3.38 What is the output when the following program is executed? #include<stdio.h>

main()

{

int i;

for(i=10;++i;i-=5)

printf(“%d”,i);

}

3.39 What would be the output when the following program is executed? #include<stdio.h>

main()

{

int i=5;

for(--i;--i;--i)

printf(“%d”,i);

}

3.40 What would be the output when the following program is executed? #include<stdio.h>

main()

{

int i;

for(i=0;i--;i--)

printf(“%d”,i);

}

3.41 What would be the output when the following program is executed? #include<stdio.h>

main()

{

int i;

for( ;x>0;d=x%10,sum+=d,d/=10);

printf(“%d”,sum);

}

3.42 What is the output when the following code is executed? #include<stdio.h>

main()

{

int p;

for(p=-10 ; !p ; ++p)

printf(“\n%d”,--p);

}

***Control Statements* 77**

3.43 Identify the error in the following program?

#include<stdio.h>

main()

{

int x=10,y=0;

do{

y+=x;

--x;

}while(x)

printf(“\n%d ” , y);

}

3.44 What is the output when the following program is executed? #include<stdio.h>

main()

{

int a=10;

do

{

printf(“%d”,a);

--a;

}while(a<=1);

}

3.45 Which of the following is exit control loop? a)for b)switch case c)while d)do-while

3.46 What is the output when the following program is executed? void main( )

{

int i = 1;

do

{

printf(“%d”, i);

}while (i=i-1);

}

3.47 What is the output when the following program is executed? void main()

{

for(x=20;x>=1;x--)

{

for(y=x;y>=1;y--)

printf(“ “);

printf(“%d\n”,x);

}

}

3.48 What is the output when the following program is executed? #include<stdio.h>

main()

{

int i,j;

for(i=1;i<=3;++i)

{

***Control Statements* 78**

for(j=1;j<=3;++j)

printf(“%3d ”,i+j);

printf(“\n”);

}

}

3.49 What is the output when the following code is executed? {

for(i=1;i<=5;++i)

{

if(i<=4)

break;

printf(“%d”,i);

}

}

3.50 What is the output when the following program is executed? void main ( )

{

int num, sum;

for(num=2, sum=0; ;)

{

sum=sum+num;

if (num>10)

break ;

num = num+1;

}

printf(“%d”, sum);

}

3.51 What is the output when the following program is executed? void main( )

{

int i=5;

while (i)

{

i=i-1;

if(i = = 3)

continue;

printf(“\n hello”);

}

}

3.52 What is the output when the following program is executed? #include<stdio.h>

main()

{

int a=1;

while(a<=5)

{

if(a==2)

continue;

printf(“%d\t”,a);

++a;

}

***Control Statements* 79**

}

3.53 How many times KLU is displayed when the following code is executed? #include<stdio.h>

main()

{

int x;

for(x=-1;x<=5;++x)

{

if(x<3)

continue;

else

break;

printf(“KLU \n”);

}

3.54 What is the output when the following program is executed? void main ()

{

int i,j;

for(i=1;i<=5;i++)

{

if(i==3)

continue;

for(j=1;j<=i;j++)

{

if(j==3)

break;

printf(“%d”,i\*j);

printf(“\n”);

}

}

}

3.55 What is the output when the following program is executed? #include<stdio.h>

main ()

{

int x=1;

for( ; ; )

{

if(x==3)

break;

printf(“%d\t”,x);

break;

}

}

3.56 What is the output when the following program is executed? #include<stdio.h>

main()

{

int i;

for(i=1;i<=5;++i)

{

***Control Statements* 80**

if(i==3)

continue;

printf(“%d\t”,i);

}

}

3.57 What is the output when the following program is executed? #include<stdio.h>

main()

{

int i;

for(i=1;i<=10;++i)

{

if(i==3)

break;

printf(“%d \n”,i);

}

}

3.58 What is the output when the following program is executed? #include<stdio.h>

main()

{

int i=1;

for( ; ; )

printf(“\n %d”,++i);

if( i>5)

break;

}

3.59 Difference between while and do-while?

3.60 Distinguish between break and continue statements?

**Comprehensive Questions:**

3.1 Compare, in terms of their functions, the following pairs of statements: while and do---while

while and for

break and goto

break and continue

continue and goto

3.2 How would you decide the use of one of three loops in C for a given problem? 3.3 What is a null statement? Explain a typical use of it.

3.4 4.In a mechanical system the applied force p is expressed as a function of t in p(t) = 20 if x <= 3

4(++2) , if t > 3

3.5 A set of two linear equations with two unknowns x1 and x2 is given below ax1+bx2=m and cx1+dx2=n The set has a unique solution

*md bn*

x1= x2=

*na mc*

*ad cb*

*ad cb*

provided the denominator ad-cb is not equal to zero. Write a program that will read the values of x1 and x2. An appropriate message should be printed

if ad-cb=0

3.6 A function is defined as follows:

***Control Statements* 81**

f(x) = { 10x2 -5x+4, if x<=0

{ 10x2 +5x+4, if x>0

Write a program to find the value of the function for given value of x.

3.7 Write a C program to print the roots of a quadratic equation after reading the value of its coefficients a, b and c.

3.8 Write a program to convert the temperature from Fahrenheit to Centigrade and vice versa. 3.9 Write a program to accept the sides of a triangle and check whether a triangle is formed or not. If formed determine the nature of the triangle(scalene, isosceles, equilateral). 3.10 write a menu driven program to compute electricity bill taking different categories of users for different slabs.

3.11 In Mechanical System the applied force p is expressed as function of t in { 20 , if t<=3

p(t)= {4(t+2), if t>3

Calculate and print the force dependent on the time provided by the user

at the prompt.

3.12 A company sells five different products with prices shown in the following table. Product Number Retail Price Unit (in Rs)

1 15.50

2 19.25

3 14.75

4 25.00

5 12.50

Write a program using switch-case construct that reads the product number and number of units sold and display the amount to be paid.

3.13 The price for one copy of a software package is Rs 5000/-. The discount for the software depends on the volume as shown in the table below

Number of copies Discount

num < 5 No discount

num<10 10%

num < 50 15%

num >=50 20%

Write a program to calculate the cost with the number of copies of the softwarepackage provided by the user at the prompt.

3.14 A company sells five different products with prices shown in the following table: Product Number Retail Price Per unit (in Rs.)

1 15.50

2 19.25

3 14.75

4 25.00

5 12.50

***Control Statements* 82**

3.15 Write a program using switch-case construct that reads the product number and number of units sold and display the amount to be paid.

3.16 Write a program that determines a student‟s grade. It reads three tests scores (between 0 to 100) and calls a function that calculate and returns a student‟s grade based on the following rules: a)If the average score is 90% or more, the grade is A.

b)If the average scores are 70% or more and less than 90%, it checks the third score. If the third score is more than 90%, the grade is A; otherwise, the grade is B.

c)If the average score is 50% or more and less than 70%, it checks the average of the second and third scores. If the average of two is greater than 70%, the grade is C; otherwise, it is D. d)If the average score is less than 50%, then the grade F.

3.17 Write a program that accepts 0, 1 or 2. If 0 is entered by the user, accept the necessary parameters (radius, height etc.,) to calculate the volume of a cylinder. Inputs 1 and 2 correspond to cylinder and cone respectively. The process must go on until the user enters „q‟ to terminate the program.

3.18 Given an integer between 0 and 6, write a program that prints the corresponding day of the week. Assume that the first day of the week (O) is Sunday.

3.19 Write for loop to find the sum of squares of first N-natural numbers.

3.20 Write a program to find a factorial of a given integer.

3.21 Write a program to display the image of given number.

3.22 Write a program the prompts the user enter an integer ,n and then a floating point number as the numbers are read the program will calculate the average of the positive numbers. 3.23 Euler‟s number,e, is used as the base of natural logarithms. It can be approximated using the following formula;

e=1+1/1! +1/2! +1/3! +1/4! +1/5! +1/6! + . . . +1/ (n-1)! +1/n!

3.24 Write a program that approximates e using a loop that terminates when the difference between two successive vales of e differ by less than 0.0000001.

3.25 Write a program to evaluate the following investment equation v=P (1+r) n and print the tables which would give the value of v for various combinations of the following values of p,r and n. P: 1000, 2000, 3000. . . , 10000

r: 0.10, 0.11,0.12,. . . ,0.20

n: 1, 2,3,. . . 10

(Hint, P is the principal amount and V is the value of money at the end of n years. The equation can be recursively written as

V=P(1+r)

P=V

That is, the value of money at the end of first year becomes the principal amount for the next year and soon.)

3.26 Write a program to print the following triangle

\*

\* \*

\* \* \*

\* \* \* \*

- - - upto nth line

***Control Statements* 83**

\*

\* \*

\* \* \*

\* \*

\*

- - - upto nth line

1

1 2

1 2 3

1 2 3 4

- - - - upto nth line

3.27 The numbers in the sequence

1 2 3 5 8 13 21. . .

are called Fibonacci numbers. Write a program using a do-while loop to calculate and print the first m numbers.(Hint: After the first two numbers in the series, each numbers is the sum of the two Preceding numbers.)

3.28 Write a program that calculates and print the average of several integers. Assume that the last value read with scanf () is sentinel number 9999.

3.29 Caluclate the values of the function

f(x) = 2xsinx +tan-1(x)+ex for x in the range of -1<=x<=5 with a step size of 0.25

3.30 Write a program that reads an integer and determines if it is a prime number. If it is not a prime number, print out the smallest divisor. Otherwise, print out the prime number.

3.31 Write a program to read an integer number between 0 and 25. Print out the character corresponding to the input number of the ASCII value. Give an error message if the input number is outside the specified range. The program terminates when the input number is -1.

3.32 The cosine function can be expanded as a Taylor series as follows:

cosx = 1- x2/2! + x4/4! – x6/6! + ------ . Write a program to find the value of cosx to a given precision.

3.33 The sin function can be expanded as a Taylor series as follows:

sinx = x- x3/3! + x5/5! – x7/7! + … . Write a program to find the value of sinx to a given precision.

3.34 The Taylor series expansion of ex is

ex=1+x+x2/2!+x3/3!+… Write a program to find the value of ex to a given precision 3.35 Calculate the values of the function

f(x,y)= x2-y2+5xy for x in the range of -1 ≤ x ≤ 5 with a step of size 1 and for y in the range of 2 ≤ y ≤ 4 with a step of size 0.5.

3.36 Write a program to check given number is prime or not.

3.37 Write a program to display the image of a given number.

3.38 Write a program to print the values of the function

***Control Statements* 84**

f(x) = 3ex+ 4 sinx + 0.6 for x=0,0.1,0.2,………….1.0

3.39 Write a program to find maximum among a given list of numbers entered through keyboard Stop reading the data whenever zero is entered.

3.40 Write a program to find the average of a given list of numbers entered through keyboard .Stop reading the data whenever -1 is entered.

3.41 Write a program to find the sum of the first n terms of the series

Sin x = x – x3/3! + x5/5! - x7/7! + ---------

3.42 What are the various looping statements? Explain the syntax giving one example to each. 3.43 Write a program to read a particular number and to check whether it is a perfect number or not. 3.44 Write a program to print first n Fibonacci elements.

3.45 Write a program to check if a given number is an element of Fibonacci sequence. 3.46 Write a program to check if a given number is Armstrong or not.

3.47 Eulers formula e is used as the base of natural logarithm . It can be approximated using: e = 1 + 1/1! + 1 / 2! + 1/ 3! + ………………………. + 1/n! + …

Write a program that approximates e using a loop that will be terminated when the new term of the series is less than 0.000001.

3.48 To evaluate ∫ab f(x) dx,by numerical integration , divide the interval [a, b]into n sub-intervals, taking the equidistant points of subdivision as

x 0, x1, …, xn . The trepizoidal rule is

a∫bf(x) dx = (h/2) [y0 + 2(y1 + y2 + ….. + yn-1) + yn]

where h = (b – a)/n and yi = f(xi) , I = 0, 1, 2, .. , n

Write a program to evaluate 0∫1 exp.(x2) dx with n = 10 by trapezoidal rule.

3.49 Write a program to find the sum of the series

sum = 1+ x + x 2+ …………………. xn.

3.50 Write a program to find the sum of series

Sum = 1+ x + x2/2 + x3/3 + …………………………. x n-1/(n-1)

3.51 Write a program to find the sum of digits of a number.

3.52 Write a program to enter integer number and find the largest and smallest digit of the number. 3.53 For a certain electrical circuit with an inductance L and resistance R, the damped natural frequency is given by

1 2

*R*

Frequency=4 2

*LC*

*C*

It is desired to study the variation of this frequency with C(capacitance). Write a program to calculate the frequency for different values of C starting from 0.01 to 0.1 in steps of 0.01 3.54 Write a program to print the numbers that don‟t appear in the Fibonacci series. The numbers of such terms to be printed should be given by the user

3.55 Write a do-while loop that will display the following sequence of numbers 7,9,11,13 … 67.

3.56 Program to print all multiplication tables between two limits.

3.57 Write a program to display all prime numbers between two given limits. 3.58 Write a program to add all prime numbers between two given limits.

3.59 Write a program to display first n prime numbers.

3.60 Write a program to print all elements of Fibonacci sequence between two limits. 3.61 Write a program to print prime numbers between two limits.

3.62 Write a program to print the first n prime numbers.

3.63 Write a program to print all two digit perfect numbers.

***Control Statements* 85**

3.64 Write a program to print all values of the function f(x,y)= ex+y-2(x+y) for all combinations of x= -5,-3,-1,1,3,5 and y = 0, 0.5, 1.0,1.5,……………. 10.0.

3.65 Write a program to display each digit of a number as many times as the digit. 3.66 Write a program to check a given number is strong or not.

3.67 Given a set of integer numbers containing positive negative and zero values. Write a program to find average of only positive values using continue statement.

3.68 Create an infinite for loop check each value of the for loop .If the value is odd display it otherwise continue with iterations. Print even numbers from 1 to 100 use break statement to terminate the program.

***Control Statements* 86**