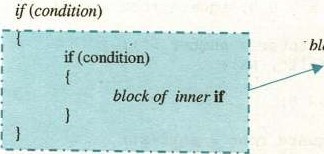
11.23 Nested if Statement

Nested if statenrnt nrans an if statenrnt inside another if statenrnt. Nesting can be done up to any level. The programnrr may use as many if statements inside another if statenrnt as (s)he wants. However, the increase in the level of nesting also increases the complexity of the •nested if statenrnt. The general form of nested if statenrnt is as follows:

- block of outer if

The else statement is optional, it may or rmy not be used with outer or inner if statement. However, if used, its block can also contain offrr if statements.

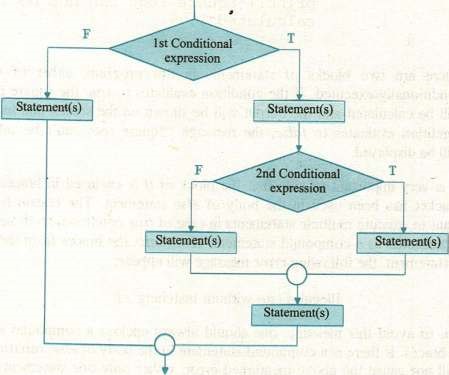


Fig. I I .3 Flow chart of nested if statement

A program that accepts three numbers from the Example 2 user and displays the largest number.

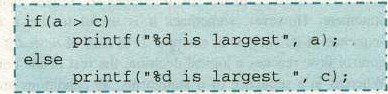
#include <stdi0.

void main (void)

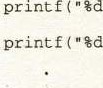
int a, b, c; 

printf ( "Enter three numbers ;

if (a > b)



else

if (b > c) is largest'• , else is largest", c)

This is a simple program that compares three numbers to find the largest one. The body of if and body of else, both contains other if statements (nested ifs are boxed). This sort of arrangement of multiple if Statenrnts is called nested if.

The execution of the nested if proceeds as follows: The first condition (a > b) is tested, if it is true then tir second condition (a > c) is tested, if it is also true. the rest of the conditions will skipl\*d and we conclude that a is the largest number. If the second condition is false, this nrans a is greater than b but smaller than c. Therefore c is Iaqgest number.

If the first condition (a > b) is false, the body of if is skipped and the flow of control is transferred to tir body of else where the condition (b > c) is tested. If it is true then the second condition (b > c) is tested, if it is also true, this means b is greater than a (from the first condition) and b is greater than c (from the Second condition), therefore we conclude that b is the largest numlEr. If the second condition in the body of else is false, it means b is greater than a (from the first condition), but b is snuller than c, therefore we conclude that c is the largest number.

In this way, we can implenvnt decision making in tir program using nested if statement.

Comparison of Nested if and Sequence of ifs

It is due to the complexity of nested if statenrnt that beginners usually prefer to use a sequence of if statenrnts rather than a single nested if statement. However, sonrtirnes it is useful to use a nested if instead of sequence of ifs such as in example 2. In case of nested if statenrnt, when the control now reaches a logical decision, the rest of the conditions are skipped. Whereas in a sequence of if statements, all conditions are tested in any case. This can be understood by the following example.



Cha

11

Example 3 A program that inputs a number from the user and determines whether it is positive, negative or zero.

#include <stdio.h> void main(void) int num;

printf('Enter a number scanf("%d", &num);

if (num > O) printfCThe number is positive"); if (num < O)  printfCThe number is negative"); if (num O) printf(The number is zero");

This program implenrnts the solution using a sequence Of if statenrnts. Suppose, the user enters a positive number, the answer is decided in the first comparison i.e. the number is positive. So there is no need to check the number for its being negative or zero as it is impossible. But, this solution suggests doing the last two comparisons unnecessarily, wasting tir CPU tine This situation may be avoided by using sone other form of if statement such as if-else statenrnt.

11.2.5 if-else if Statement

Nested if staterrrnts can YEconr quite complex, if there are more than three alternatives and indentation is not consistent, it may be difficult to determine the logical structure of the if statement. In such situations, if statement with multiple alternatives (if-else if) can be a good option. The form of if-else if statenrnt is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| A Textbook | Science |  |  |

if (condition1) statement/ •, else if (condition2) statement2;

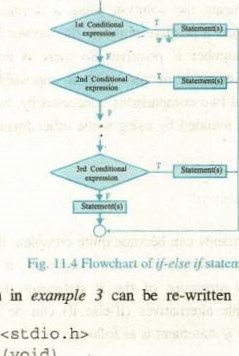
else if (condition.)



 statementk•,

The test conditions in if statement with multiple alternatives are executed in sequence until a true condition is reached. If a condition is true, the staternent(s) following it is executed, and the rest of the alternatives are skipped. If a condition is false, the statenrnt following it is skipped and the next condition is tested. If all conditions are falls, then staterrntk following the last else is executed. TIE following flowchart shows the execution flow of the program through if-else if statenrnt. 

re-written using if-else if structure as follows: \*include void main (void)



slatement

The

program

int num;

"Enter a number 9 ) ; scant ; if (num > O) printf ("The number is positive") ; else if (num < O) number is negative") ; else number is zero") ;

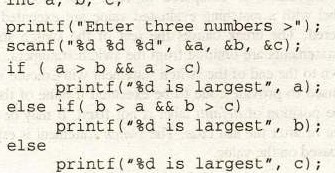
It can be seen that the else-if construct has greatly simplified the program, while preserving the efficiency as well. If a positive number is entered, we will reach the answer in the first comparison and rest of the conditions will be

11.3 USE OF LOGICAL OPERATORS

We have studied the three logical operators (AND, OR, and NOT represented by Il, and respectively) in chapter 2. These operators play an important role in constructing certain compound conditions to be used with ifstatement. Until now, we have been using simple conditions with if statenrnt and its variations. In this section, We shall observe how complex program logic can sirnplified using logical Operators.

To grasp the concept, we re-write the program in example 2 using logical AND operator. This will demonstrate how much the task is simplified.

#include void main (void)



int

a,

b,

c;

There may be situations where the use of logical operators may simplify the program logic. We just need to concentrate the underlying problem. We shall see more examples of using logical operators in next chapters.

11.4 SWITCH STATEMENT

The switch staterrrnt can also be used in C to select one of many alternatives. Like if statement, it is also a conditional staterrnt that compares the value of an expression against a list of cases. The case label can an integral or character constant. Similarly, the value of the expression must be an integer or a character; it must not be a double or float value. Here's the syntax of switch statenent.

switch(expression)

 case vall: statements",

case va12: statements\*



case vain:

statementsn•, break;

# statementsk;

The value of expression is compared to each case latrl. The statenrnts following the first label with a matching evaluated value are executed until a break statement is encountered. The break causes the rest of the switch statenrnt to skipped. If all break statenrnts are omitted from the switch staterrrnt, the code from the first true case down to the end of the switch statenE•nt will execute sequentially. A default label may be used to provide code to be executed if none of the case label is hutched. However the position of default label is not fixed. It may be placed before the first case statement or after the last case. The switch staternent is especially useful when the selection is based on value.

Example 4 Write a program that inputs a character from the user and checks whether it is a vowel or a consonant.

\*include void main (void)

char ch;

printf ( "Enter an alphabet" ) ; ch = getche() ; 

swi tch ( ch)

case •a' :

case :

printf a vowel") break ; case case 'E' :



:

a vowel") ; break ; case 'i' :

case 'I' :

a vowel") ; break; case 'o' : case

Of Class 12



break; case •u' :

case a vowel") ; break;



default :

 is not a vowel") ; break;

In this program, we have used two case labels, one after the other, without having any other staternent between them. This sort of arrangenrnt of case labels works equivalent to the logical OR operator i.e„ whether the value of trr variable ch is 'a' or 'A', the code following the labels case 'a and case 'A' will execute.

* The value of expression in switch statenrnt must be of type int or char, but not of typefloat or double.
* If the value of expression in mitch statenrnt is of type float or double„ the compiler will generate the following error nyssage:

Switch selection expression must be of integral type

11.5 CONDITIONAL OPERATOR

Conditional operator is used as an alternative to the if-else state11Ent. It is a ternary operator (requires three operand). Its general form is:

conditional expression? true-case statement : false-case statement;

The expression may be a relational or logical expression. If the expression is true then the true-case statement will executed otherwise the false-case statenrnt is executed. e.g:,

#include

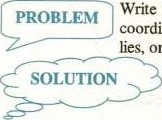
#include <conio void main()

int a, b; printf ( aplease enter two numbers: "

a > b ? is larger", printf ( is larger", b) ;



11.6 CASE STUDY: Locating a point in the coordinate plane

Write a program that input x- and y-coordinates of a point in the coordinate plane. Based on these values, it then determines wtrre it lies, on x- or y-axis, or in any of the four quadrants.

The first step towards solution of any problem (simple or complex) is to understand it. Think on the problem from all aspects; identifr input and output requirenrnts and different types of restrictions on the data. After analyzing the problem, try to develop a simple solution. Don't indulge yourself in unnecessary details. You can trace the solution on the paper in the form of an algorithm. An algorithm is a step-by-step procedure to solve any problem in finite number of steps. Let's work out problem described above:

|  |  |
| --- | --- |
| 294 quad | quad |
| 3rd quad | quad |

It is clear from the figure 11.1 that each point will lay on either of the two axes or in any of the four

Input Values: The program will input the values of x-coordinate and y-coordinate. These values will be Of type int.

Output: The program will output a nrssage describing the position of the point in the coordinate plane.

Fig. 11.1 -Co«dinate plane What do we know?

* If x- and y-coordinates are zero, point will be at the origin
* If x-coordinate is zero and y-coordinate is non-zero, then tlr point will be on y-axis. 
* If y-coordinate is zero and x-coordinate is non-zero, then the point will be on x-axis.
* If both x- and y-coordinates are positive ( > 0), the point will lie in the 1 quadrant.
* If x-coordinate is negative ( < O ) and y-coordinate is positive ( > 0 ), the point will lie in the 2nd quadrant.
* If x- and y-coordinates are negative ( < O), the point will lie in 3rd quadrant.
* If x-coordinate is positive ( > 0 ) and y-coordinate is negative ( < 0 ), the point will lie in the 4th quadrant. 



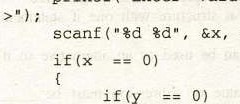
of

NOW, keeping in view all this information, let's try to write a C program to solve this problem:



# include void main (void)

int x, Y; printf ( "Enter value forand Y—coordinates



print f ( "Thelie on the origin") ; else printf ( "The point lie on y—axis") ;

else if(x

if (Y printf ( "The point lie on x—axis") ;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| else if(y > 0) |  |  |  |  |
| printf ( "The quadrant ) else | point |  | in |  |
| p;-intf ( "The quadrant " ) ;  if (Y | point | 1 les | in | 4th |

printf ( "The point lies on x—axis") ; else if (y > O)  printf ( "The point lies in 2nd quadrant ) ; else 



printf ( "The point lies in 3rd quadrant " ) ;



Fill in the blanks:

control the flow of execution in a progranu is a group statenrnts encloséd in braces.

(iii) if staterrrnt is used to select a pathflow in a program based on a

is a pictorial representation of a program refers to a structure with one if statenrnt inside another if staterrnt.

1. statenrnt can used as an alternative to if-else if statement.
2. In switch statenrnt the value Of e\*pression must
3. The —StatenEnt switches the control outside the block in which it is used.
4. The purpose of latrl in switch-case statenrnt is the sarne as that of else in if-else statenrnt.

|  |  |  |
| --- | --- | --- |
| Decision Constructs |  | 11 |

A condition is an expression that is either 



1. Write T for true and F for false staterrrnt.

An arithnrtic expression can not be used as condition in if statenrnt. The conditional operator is a ternary operator.

* 1. Logical operators are used to compare values.

A switch statenrnt can not be used within the blockof an if staterrrnt. The break statenrnt stops the execution of a program for a 1mrTEnt and then resunrs.

* 1. In sequence structure, statenrnts are executed in the order in which they appear in the program.
  2. A false condition always evaluates to zero.

(yiii) Use of the statenrnt terminator at the end of an if statenrnt causes a syntax error.

(ix) The switch expression can not of float or double C is an unstructured programming language.

1. What is a control structure? Briefly describe the basic control structures for writing programs.
2. How many selection statenrnts are available in C? Discuss trr difference trtween them.
3. Write the general form of the following statenrnts: if statenrnt with one alternative if statement with two alternatives
   1. if statement with multiple alternatives (iv) switch statenrnt
4. Rewrite the program given in the example 4 using if statenrnt.



Attempt the following parts: 

Assuming x is 10.0 and y is 15.0, what are the values of following conditions:

a)

b)

c)

d)

 [Hint: The value of a true condition is 1 and a false condition is O]

Write an expression to test each of the following relationships.

* + 1. age is from 18 to 25.

|  |  |
| --- | --- |
| Textbook ofComputer Science |  |

* + 1. temperature is less than 40.0 and greater than 25.0.
    2. year is divisible by 4 (Hint: use
    3. speed is not greater than 80.
    4. y is greater than x and less than z.

w is either equal to 6 or not greater than 3.

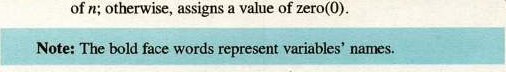
Note: The bold face words represent variables' nanrs.

* 1. Write assignnrnt Staten-Ent for the following:

a) Assigns a value of I (one) to the variable test if k is in the range— m through inclusive. Otlrrwise, assigns a value of zero.

 Assigns a value of one (l) to the variable lowercase if ch is a lowercase letter; otlrrwise, assigns a value of zero (O).

c) (1) to the variable divisor if m is a divisor



Assigns

a

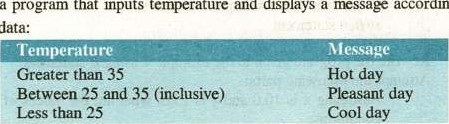
value

of

one

Writé an interactive program that contains an if statenrnt that may be used to computer the area of a square (area = side2) or a triangle (area = h x base x height) after prompting the user to type the first character of the figure nanr (S or T).

1. A year is a leap year if it is divisible four, except that any year divisible by 100 is a leap year only if it is divisible by 400. Write a program that inputs a year such as 1996, 1800, and 2010, and displays 'Leap year" if it is a leap year, otherwise displays 'Not a leap year".
2. Write a the following data:



program

that

inputs

temperature

and

displays

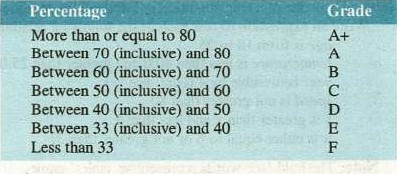
a

according

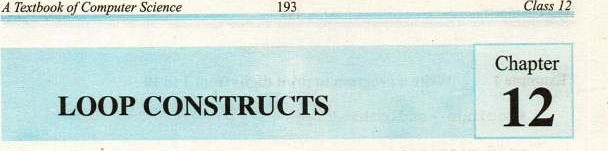
to

|  |  |  |
| --- | --- | --- |
| Decision Constructs | c | n |

Write a program that inputs obtained marks Of a student, calculates percentage (assume total marks are 1100), and displays his/her grade. The grade should be calculated according to the following rules:



1. Write a program that inputs two numbers and asks for the choice of the user, if user enters 1 then displays the sum of numbers, if user enters 2 then displays result of subtraction of the numlrrs, if user enters 3 then displays the result of the multiplication of the numbers and if user enters 4 then displays result of the division Of the numbers (divide the larger numtrr by the srnaller number), otherwise displays the nwssage 'Wrong Choice". [Use a switch statenrnt to implenrnt the solution]



* 1. OVER VIEW

 We often encounter problems whose solution may require executing a statement or a set of statements repeatedly. In such situations, we need a structure that would allow repeating a set of statements up to fixed number of times or until a certain criterion is satisfied. In C, Loop statenrnts fulfill this requirement. This chapter will provide the basis for writing iterative solutions to certain problerns. Here, we shall introduce different loop constructs available in C.

Iteration is the third type of program control structure (sequence, selection, iteration), and the repetition of statements in a program is called a loop. There are three loop control statements in C, these are: 

 while  do-u.'hile  for

* 1. WHILE

The while loop keeps repeating associated statements until the specified condition becomes false. This is useful where the  does not know in advance how many times the loop will be traversed. The syntax of the while statement is:

while (condition)  statement(s);

The condition in the while loop controls the loop iteration. The Statements, which are executed when the given condition is true, form the body of the loop. If the condition is true, the body of the loop is executed. As Soon as it beconrs false, the loop terminates immediately.

Loop Constructs Chapter

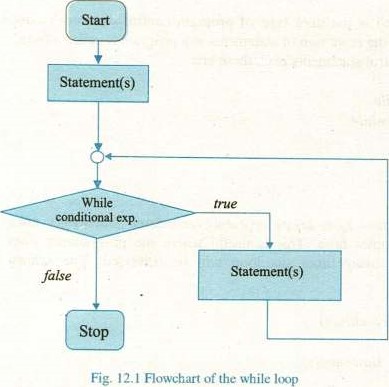
'xample i Write a program to print digits from I to 10.

#include void main (void)

int count ;

count = 1; while( count 10)

printf count) ; count = count + 1;



This is a simple program which demands iterative solution. It does not make sense to use ten printf StatenEnts to print ten digits; if so, what if we have to print digits from I to Should we write one thousand printf staterrents to accomplish the task? Certainly not, the right way to cone up to the solution is to use a loop,



which would execute ten tirnes. Each tinr the loop executes, a number is printed



which is incremented by one for every iteration until the required list of numbers is printed,

In this program, We use a variable count which is initialized to 1. The condition (count 10) depends on the value of this variable. Until the condition is true, the control will enter the body of the loop, and as soon as it twomes false, the control will exit from the loop and will jump to the next statement to the body of the loop. First time, when the condition is checked, it is found true as the value of count which is one, is less than ten. The control enters the body of the loop and the number "1" is printed. The next line of code incrernents the value of count by one, which becomes '2". After that, the control will immediately jump to the while staternent where again the condition is tested which is still found true, as 2 is less than 10. The control again enters the body of the loop, and the number '2" is printed. The value of the variable count again increases by one and beconrs The control again transfer to the while staternent. This process continues until the value of count becornes "11 making the condition false. When the condition beconrs false, the control will exit • from the loop.

The count is the loop control variable. A variable whose value controls the number of iterations is known as loop control variable. The compound statenrnt, which is enclosed in braces, is the body of the loop.

In whilé loop, the loop eonti•öl variable always initialized Gütside thé bbd) the loop and is incremented or decrerrrnted inside the loop body

12.3 DO-WHILE LOOP

This is very similar to the while loop except that the test occurs at the end pf the loop body. This guarantees that the loop is executed at least once. This loop is frequently used where data is to be read; the test then verifies the data, and loops back to read again if it was unacceptable. The syntax of the do-while statenrnt is: 

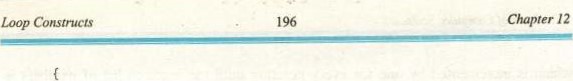
do

 statement(s);

)while (condition);

The important point about this loop is that unlike while loop, it ends with a semicolon. Omitting the semicolon will cause a syntax error. Let us re-write the prpgram in example 1 using do-while loop.

#include void main(void)



int count ; count —

count) ;

count = count + 1;

) while( count IQ) ;

Here, we achieve the same objective as in previOus example but in a different Way. The keyword do let the program flow to move into the body of the loop without checking any test condition. It nrans, whatever is written in the loop body always will be executed at least once. At the completion of execution of the body of the loop, the test condition is checked. If it is found true, the control is transferred to the first statement in the body of the loop, and if the condition is evaluated to false, the loop terminates immediately and the control moves to the very next instruction outside tir

The do-while loop is of great importance in situations where we need to execute certain statements at least once.

Example 2 Your telephone connection may be in any of two states i.e.. working or dead. Write a program that reads the current state of the telephone line; the user should enter w ror working state and d for dead state. Any input, other than w or d, will be considered invalid. We want to force the user to enter a valid input This could be achieved by using a du-white loop. Let us consider the following program:

# include void main( void )

char state; do



print f (u \nP1ease enter the current state of the telephone line (enter \ for working and \ 'd\' for dead) ;

&state) ;

)while (state •w' state

This program demonstrates a scenario where an invalid input is not processed.

Until the user enters a valid input (d or w), the program repeatedly shows him (or her) the message for the valid input to be entered.

 Here, the key point is the correct understanding of the test condition (state != 'w' && state 'd'). It is a compound condition which is comprised of two sub conditions i.e. state 'w' and state 'd'. It should be noted that if two or more conditions are combined using logical AND operator to form a compound condition, the compound condition will be true only if all the sub conditions are true and if any of the sub conditions is false, the compound condition evaluates to false. Therefore, when the user enters an invalid input (suppose e), the first sub condition state 'w' evaluates to true (becuause e is not equal to w), similarly the second sub condition state != 'd' also evaluates to true. Since both the sub conditions are true, therefore the compound condition also evaluates to true and the control flow returns back to the print/ staternent in the body of the loop. This process continues until the user enters a w or d. When the user enters a d or a w, one of the sub conditions evaluate to false causing the compound condition to be evaluated to false and the control flow exit the loop.

while Vs do-while

In while loop, the body of the loop may or may not execute depending on the evaluation of the test condition.

In do-while loop, first the body of the, loop is executed and then the test conditionis checked. Hence it always executed at least once,

The for statement is another way of implementing loops in C. Because of its flexibility, Imst programnrrs prefer the for statement to implement loops. The syntax of thefor loop is as follows: for (initialization expression; test condition: increment/decrement expression) statement(S);



C

r

There are three expressions infor loop statenrnt, these are

* Initialization of the 10op control variable
* Test condition
* Change (incrernent or decrernent) of the 100p control variable

The initialization expression is executed in only the first iteration. Then the loop condition is tested, If it is true, the statenrnts in the body of the loop are executed. After execution of the body of the loop, the incrernent/decrenrnt expression is evaluated. It is very important to note that the initialization expression is only executed for the first iteration. For second and next iterations, the loop condition is tested, if it is true then the body of the loop is executed and then the increment/decrernent expression is evaluated. After evaluation of the increment/decrement expression, the test condition is checked again and if it is true then the body of the loop executed. This process continues as long as the loop condition is true. When this condition is found to be false, the for loop is terminated, and the control transfers to the next statenrnt following the for loop. Usually, we increment or decrement the loop control variable in tir increnrnt/decrenrnt expression. Let us re-write the program in example] usingfor loop.

#include void main (void)

int count ;

for (count = I; count 10; count++) printf( , count) ;

There are three expressions in for loop statenrnt, separated by semicolons. Two Of the expressions i.e. initialization expression and increment/decrement expression are optional. We may omit these expressions. In this case, the for Statement will be written as follows:

for ( ; count 10; )

The loop condition is mandatory. This can not omitted. In this case we must have to initialize the loop control variable outside the for statement and it should be incremented or decremented inside the loop body.

12.4 NESTED LOOP

Nested loop nrans a loop inside the body of another loop. Nesting can done up to any level. But, as the level of nesting increases, the complexity of the



of

nested loop also increases. There is no restriction on the type Of loops (while, dowhile, or for) that may placed in tir body of other loops. For example, we can place one or rmre while or do-while loops in tir mdy offor loop. Similarly, One or for loops can placed in the body of while or do-while loop.

3  Write a program that will print asterisks according to the

Example

- pattern shown in the fig. 12.2.

# include

void main (void)

int inner;

 for (int outer=7;  outer— — )  12.2 asterisks

inner = 1; while ( inner Outer)

inner +4;

printf ;

In this program. a while loop is used inside the mdy offor loop, which shows a nested loop. The outer loop is controlled by tir loop control variable outer. TIE outer loop is executed seven tinrs. For each iteration of the outer loop, the inner loop executes until tlr value of tir inner loop control variable i.e„ inner is Ess than or equal to value of variable outer. It should noted that each a new iteration for the outer loop starts, variables used in tlr inner loop are re-initialized and re-processed.

For the first iteration of the outer loop, variable 'outer' is initialized to 7, and all next iterations it is decrenrnted by 1. This process continues until the value of the variable is greater than or equal to 1. For first iteration of the outer loop, tir  loop executes seven tines, and for tie 2nd iteration it executes six tinrs, similarly for the last seventh iteration, the inner loop executes just one tine Each

LOOP Chapter

time when the inner loop is terminated, the statement printf "\n" ) Imves the cursor to the start of the new line.

Note:

Many programs require a list of items to entered by the user. Often, we don't know how many items the list will have. For example, to find the average marks of a class, we have to input the marks of every student of the class. Similarly to calculate the sum Of a series, we have to input the list of numbers in the series. There are so many other situations where the solution demands to enter a list of iterrus to process. Loops are very useful to develop solutions for such problenß. Each tine the loop body is repeated, one or more data items are input. But, often we don't know how many data items will be input by the user. Therefore, we must find sorr way to signal the program to stop reading and processing new data.

One way to do this is to instruct the user to enter a unique data value, called a sentinel value, after the last data item. The loop condition tests each data item and causes loop exit when the sentinel value is read. Choose the sentinel value carefully, it must be a value that could not normally occur as data. The general form of a sentinelcontrolled loop is:

1. Get the first line of data
2. While the sentinel value has not been encountered
3. Process the data line
4. Get another line of data

Sentinel Value is an end marker that follows the last item in a list of items

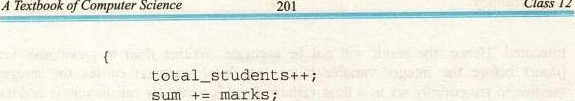
Example 4  a program to 

#include void main (troid)

int sum O, marks, total\_students - o;— float average;

printf( "Enter. marks of the student (or any —ve number to quit) > " ) ; scant ( "\*d", &marks) ;

if (marks O)



Class

marks

) while (marks Y— if (total\_sLudents > 0)

average = sum / (float) total\_students; printi "The average marks of the class are:

\n", average) ; 

•printi ( "Please enter the marks of at least one student to calculate average\n" ) ;

This program demonstrates a typical implenrntation of sentinel loop. Size of the class does not nutter, whatever it is, the average will be calculated in the sane way. Here any negative number may act as the sentinel value because no student can have negative marks. However, zero would not be a wise option because there can a student with zero marks.

The program reads the marks until the user enters a negative number. For every valid input (zero and •f•ve numtErs) the control switches to the body of the while loop. In the loop body, the total\_students is incremented by one and the sum is accumulated. As soon as a negative nurnlrr is entered, the sentinel while loop is terminated. The next line to the end of the while loop is an if staternent, which checks the count for, total students i.e„ total\_students to ensure that the marks of at least one student have been entered. Omitting this if statement may crash the program. It is because of the formula for calculating average where the sum is divided by total\_students. When marks of any students are not entered, the value of total\_students is zero and calculating average for zero students will result in a runtinw error of division by zero. So, to avoid this possible error first the value of the variable total\_students is cheked; if its greater than zero then the average is calculated otherwise a message is shown to the user to enter the marks of at least one student.

Now, notice the average formula i.e.,

average = sum / (float) totalStudents;

We have used the keyword float in parenthesis the variable total\_students. The reason is that both the variables sum and total\_students are integers. So, their division will be integral division in which the fractional part is

|  |  |  |  |
| --- | --- | --- | --- |
|  | Constructs | C | Ter 12 |

truncated. Hence, the result will not accurate. •Writing float in parenthesis i.e. (float) before the integer vanable narr (i.e. total\_students) causes the integer variable to temporarily act as a float variable for this particular calculation. It is done to preserve the fractional part in the result. The integer variable (total\_students) will act as an integer for all other calculations. The effect of this change is strictly associated with that particular calculation. This phenonrnon is known as type casting.

12.5 GOTO STATEMENT

The goto statement performs an unconditional transfer of control to the nanEl label. The label must be in the sanr function. A label is nraningful only to a got) statement; in any other context, the labeled statenrnt is executed without regard the label.

The general form of the goto statenrnt is as follows: goto label;



label: statement

5 Write a program to calculate the square root of a positive number.

Example

 Also handle negative numiv•s properly.

#include <math . # include



.

void main ( )

float num;

positive: 

printf( "Please Enter a positive number: 

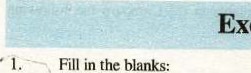


if (num 0) goto positive; else root Of 2f is .2f", num sqrt (num) ) ;

If the user enters a negative number, the control transfers to the label positive. 

of

|  |  |
| --- | --- |
|  | Class n |

Exercise 12c

Fill in the blanks:

1. There are types of loop in C.
2. The loop condition controls the loop
3. In loop, first the body of the loop is executed and then the test condition is checked.
4. nuns a loop within the body of another loop.
5. The statement performs an unconditional transfer of control to the named label.
6. Repetition of statenynts in a program is called
7. There areexpressions in for loop statenrnt. (viii) The body of the while loop executes only if the specified condition is
8. Increase in the level of nesting increases theof the nested loop.
9. is meaningful only to a goto statement.
10. Write T for true and F for false statenrnt.
    1. There is no difference between while and do-while loop.
    2. The body of a while loop may or may not execute.
    3. The do-while loop always executes at least once.
    4. var++is an example of prefix increment.
    5. The condition of an infinite loop never beconw true.

 Initialization expression is optional infor loop.

(vii)for( i = 1; i 10; i++) ; isaninfiniteloop.

* 1. Loop is a decision rnaking construct.
  2. A while loop can not usedin body of afor loop.

(S)In type casting, a variable of one type t\*haves as the variable of . another type temporarily.



1. Define a loop. How many loops are available in C? Compare the following

while loop and do-while loop  while loop and for loop

1. What is a sentinel controlled loop and how it is implenrnted? Discuss sone Of tir situations wirre it can useful.

Write the output of the following program fragnrnts:

while (k



5)

%3dW, k, 10 -k)

Trace the output of tlr following of code.

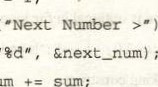
(int i = l; i 5; ++i)



6. Correct the following code segrrrnts according to the given instructions: Insert where they are needed and correct errors if any. Ttr corrected code should accept five integers and should display tlrir

count O; while (count 5) ; count +2 1;

; scanf ( next\_num numbers were added; \n•, count) ; sum is \n", sum) ;

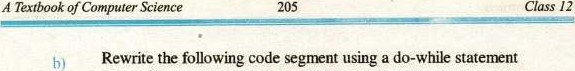


print

f

(

sum;



sum = O; for (odd = 1; odd < n; odd = odd + 2) sum = sum + odd;

 Of positive odd numbers less than %d is n,

Trace tir output of ttr following program segrents, assuming m is 3 and n is 5:

a) 

for (j = O; j < k; 4—Fj) printf("\*");

b) for (k = n; k > O; —k)

for (j = m: j > O; —j)



printf('W);

Re-write the program in example3 by replacing the inner while loop with afor loop  a do-while loop

Re-write the program inexample3 by replacing the outer for loop

 a bhile loop  a do-while loop 



Chapter

12

206

1. Write a program that inputs a nurntrr and displays nrssage "Prinr number" if it is a prirrr numixr, otherwise displays 'Not a prinr numirr"
2. Write a program that displays the first 15 even numbers.

l l . Write a program that inputs a number, and displays its table according to the following format:

Suppose the number entered is 5, the output will be as follows:



10=50

1. Write a program using do-while loop that repeatedly prompts for and takes input until a value in the range O through 15 inclusive is input, The program should add all the values t\*fore exiting the loop and displays their sum at the
2. Write a program that produces following output:

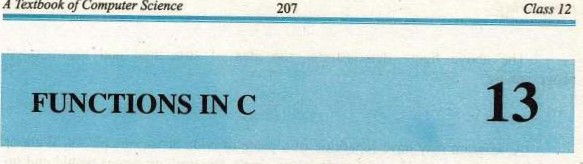
0

o 1 2 3 o 1 2 3 4 o 1 2 3 4 5

1. Write a program the produces the following output:

 2

* 1. 4
  2. 8
  3. 16
  4. 32
  5. 64



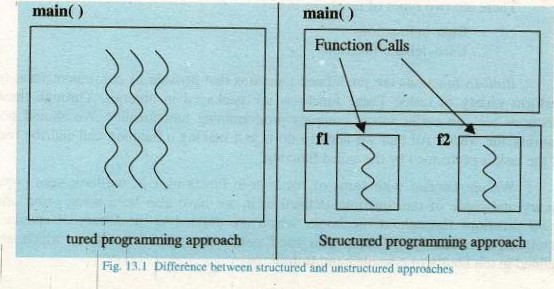
of

13.1'

OVERVIEW

The idea of modular programming is the result of inspiration from the hardware manufacturing where replicable components of different items are available. If a component of an item gets out of order, it is replaced with a newer one. Many different components from different manufacturers can be combined together to form a hardware device such as computers, cars, and washing machines. Functions are the building blocks of C programs. They encapsulate pieces of code to perform specific operations. Functions allow us to accomplish the similar kinds of tasks over and over again without being forced to keep adding the same code into the program. Functions perform tasks that may need to be repeated many tinys.

In programs we have seen so far, the whole program logic was contained in a single main function. This style of writing programs is known as unstructured programming. Recall chapter 8 where we have discussed the difference between unstructured and structured programming. Here we shall discuss structured programming approach. It is a rm)dular way of writing programs. The whole program logic is divided into numirr of smaller modules or functions. The main function calls these functions where they are needed. A function is a self-contained piece of code with a specific purpose.



Functions in C 208 Chapter 13



The above figure demonstrates the idea of structured and • unstructured programming.

13.2 IMPORTANCE OF FUNCTIONS

A program may have repetition of a piece of code at various places. Without the ability to package a block of code into a single function, programs would end up being much larger. But the real reason to have functions is to break up a program into easily manageable chunks. The use of functions provides several benefits. Some of them are:

 They make programs significantly easier to understand and maintain. The main program can consist of a series of function calls rather than countless lines of code.

* Functions increase reusability of the code. Well written functions may be reused in multiple programs. The C standard library is an example of the reuse of functions.
* Different programnrrs working on one large project can divide the workload by writing different functions, hence ensuring the parallel development of the software.
* Functions can be executed as many times as necessary from different places in the program
* When an error arises, rather than examining the Whole program, the infected function is debugged only. TYPES OF FUNCTIONS

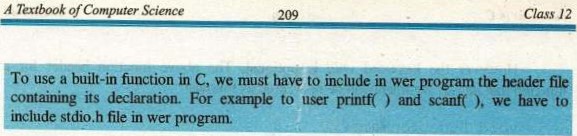
There are two types of functions in C:

# Built-in functions

(iii) User-defined functions

Built-in functions are predefined functions that provide us convenient ways to perform variety of tasks. These functions are packaged in libraries. Through these functions we can easily access complex programming functionality. We should not reinvent the wheel. All that we need to do is just making a function call and the rest of the task is performed by the called function.

We are familiar with sorr of the built-in functions, e.g„ we have seen ctype library and some of the functions defined in it, we have also been using prinrf and scanf functions throughout the book, which are defined in the library of standard input/output, similarly we have also used getch and getche functions which are defined in the library of console input/output.



Built-in functions are not sufficient for solving every kind of problem. A programmer rnay need to write his/her own functions depending on the nature of problem being solved. Such functions are called user-definedfunctions.

13.4 WRITING FUNCTIONS IN C

We are familiar With the main( ) function, which is the mandatory part of every C program. In chapter, we have introduced the structure of the main function. Every function in C has almost the basic structure. A function in C consists of a function header which identifies the function followed by the body of the function between curly braces containing the executable code for the function. Every function in Cas written according to the following general form:

returen\_type •FunctionName  )

Executable Statement (s) return expression;

13.4.1 Function Header

The first line of function definition is called the function header i.e.

return\_cype Funct ionName (parameter\_list)

It consists of three parts:

* The type of the return value
* The narne ofthe function
* The parameters of the function enclosed in parentheses

The return\_type can be any valid data type. If the function does not return a value, the return type is specified by the keyword void. A function that has no paranEter specifies the keyword void as its parameter list. Hence, a function that has no parameter and does not return any value to the calling function will have the

void FunctionNarne (void)

|  |  |
| --- | --- |
| Functions in C | Chapter 13 |

However the keyword void is optional. TIE above function header for a ftnction that has no argument can be re-written as follows:

void FunctionName( )

13.4.2 The Function Body

Variables declaration and the program logic are implemented in the function body. Function body makes use of the arguments passed to the function. It is enclosed in curly braces. A function can called in the body ofanother function.

13.4.3 The return Statement

The return statenrnt is used to specify the value retuned by a function. The general form of return statenrnt is:

return [expression) ; 

When the return statement is executed, expression is evaluated and returned as the value of the function. Execution of the function stops when the retum statement is executed, even if there are other statements still remaining in the function body. If the type of the return value has been specified as void in the function header then there is no need to use a return Statement.

13.5 FUNCTION PROTOTYPE

The compiler must know functions used in the program% That's why we include corresponding header files in the source program before using built-in functions such as stdio.h and •conio.h etc. A header file contains the prototypes of the functions provided by the library. The compiler actually needs enough information to be able to identify the function that we are using. Afunction prototype is a statement that provides the basic information that the compüer needs to check and use a function correctly. It specifies the parameters to passed to the function, the function name, and the type of the return value. The general form of the function prototype is as follows:

return\_type FunctionNarne' (parameter\_list) ;



We might be surprising at the above statenrnt. It looks like the function header, yes it is, but with a semicolon at the end.





The prototype for a function which is called from another function must appear before the function call statenrnt. Functions prototypes are usually placed at the beginning of the source file just before tir function header of the main function.

13.6 CALLING A FUNCTION

Function call is a nechanism that is used to invoke a function to perform a specific task. A function call can invoked at any point in the program. In C the function narne, the arguments required and the statenrnt terminator ( ; ) are specified to invoke a function call.

When function call statement is executed, it transfers control to the function that is called. The memory is allocated to variables declared in the function and then the statements in the function body are executed. After last staterrnt in the function is executed, control returns to the calling function.

13.7 LOCAL VARIABLES AND THEIR SCOPE

 When the program executes, all variables are created in nrrnory for a limited time period. They come into existence from the place where they are declared and then they are destroyed. The duration in which a variable exists in nrmory is called lifetime of the variable.

Note: Operating system nunages the allocation and de-allocation of memory for åil variables in wer programs. So, by destroying a variable we mean returning the —wry allocated to a variable back to the operating system for other programs,

The scope of a variable refers to the region of a program in which it is accessible. The name of a variable is only valid within its scope. So a variable can not be referred outside its scope. Any attempt to do so will cause a compiler error.

All variables that We have declared so far have been declared within a block — that is, within the extent of a pair of curly braces. These are called local variables and have local scope. The scope Of a local variable is from the point in the program where it is declared until the end of the block containing its declaration.



LE—k1L..,

#include void main ( )

int nCount = O; if (nCount 0)

int chk; chk 10;

chk) ;

We have used two variables in this program; these are nCount, and chk. Both of these are local variables. But, they have different scope. The scope of the variable nCoun1 is the block of main( ) function from its point of declaration to the end of the main( ) function. Whereas the scope of the variable chk is the block of ifstatenrnt

i.e„ from its point of declaration until the end of the block of if statenrnt.

These variables can only be referenced within their respective scopes. Any reference made to them outside of their scopes would illegal, thus the program causes the following compiler error.

'chk' : undeclared identifier

This is because in the last printf( ) statenrnt of the program, tir variable chk is referenced outside of if block i.e„ out of its scope, which is illegal. The lifetime of local variables is the duration in which the program control rernains in -the block in which they are declared. As soon as the control moves outside of their scope, these variables are destroyed.

13.8 GLOBAL VARIABLES AND THEIR SCOPES

The variables which are declared outside all blocks i.e. outside the rnain( ) and all other functions •gre called global variables and have global scope. They are accessible from the point where they are declared until end of the file containing them. It means they are visible throughout all the functions in the file, following their point of declaration. The lifetirne of global variable is until the termination of the program. They exist in rnemory from the start to the end of the program.

o r

|  |  |  |
| --- | --- | --- |
|  | 213 | Class 12 |

Note: It is very important to understand that every tine the block of st containing a declaration for a local variable is executed, the variable is created anew/ and if we specify an initial value for the local variable, it will be reinitialized each tinp it is created.

Example:

#include void main() int nCount = 1

clrgcr() ;

While(nCount 10)

int chk 10; chk) ;

chk chk + I; nCount++ ;

In this program, each repetition of the loop prints the value of the variable chk The addition to the value of chk will have no effect, trcause at the end of execution of the body of the loop, the control moves outside the loop body (which is also tir scope of chk variable) and returns to the while statenrnt•, this causes the chk variable to be destroyed in each repetition. The chk variable is again created in the next repetition and gets destroyed at the end of the repetition. This process continues until the loop condition is true.

Output:

10 10 10 10 10 10 10 10 10 10

#include void Counter (void) ; int nCount = 0; void main()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | rer 13 |

for (int n = O;

Counter() ;

printf ("nCount nCount) ;

void Counter (void)

nCount++ ;

This is a simple program which demonstrates the use of global variable. Here, we have declared a global variable i.e., nCount outside the main and the Counter functions. This is not contained in any block. TIE global variable nCOunt, the function main, and the function Counter all are defined in the sarne file. Because, the variable nCount is declared on top of the two functions, therefore it is visible within them. The function Counter, increnrnts the value of nCount by one each tinr it is called. The main() executes a loop six tinrs and call the function Counter to increment the value of nCount. The value of the variable nCount is printed as the final output of the program i.e.,

Output:

nCount

Note: The point to be noted here is that the variable nCount is declared outside functions main() and Counter(), but they manipulate it as if it was declared within them. The nCount is created in nrmory before the start ofexecution of the main() and texists until the execution of the program ends.

13.9 FUNCTIONS WITHOUT ARGUMENTS

The simplest type of function is one that returns no value and no arguments are passed to them. The return type of such functions is void and the ParanEter\_List may either be empty or containing the keyword void. Lets consider the following

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A Textbook | C | r Science | 215 | Class 12 |
|

•ample 3Write a function named Print\_Asterisks that will print

asterisks according to the pattern shown in the fig. 13.2. and invoke a function call from the function main to print the asterisks.

#include

void print\_Asterisks (void) ; / / function

 / / prototype void main (void)

|  |  |
| --- | --- |
| // Function call print\_Asterisks ( ) ; | Fig. 13.2 osterisks pattern |
| void print\_Asterisks (void) | // function header |

int inner ;

for (int outer=7;  outer-—)

inner 1; while( inner Outer)

inner++ ; printf ( ø\n");

We have discussed this program in the previous chapter, but here we have followed a different approach. The next line to the #include directive is the prototype for the function Print\_Asterisks( It tells the compiler about the function, its return\_type and number of paranrters (void) in this case. Our main function consists ofjust one line of code i.e., print\_Asterisks ( ) ;

It represents afunction call to the function Print\_Asterisks( ). We can think of a function as a worker who takes necessary steps to accomplish the task assigned to

Similarly, the function print\_Asterisks ( ) is capable of printing asterisks in a specific order. When the function call statenrnt is executed, the control is immediately transferred to the Print\_Asterisks function. Memory is allocated to the variables inner and outer. Then conrs the for and while loops, which print asterisks, When the task is completed, the control is transferred to the function main from the function Print\_Asterisks, and the memory allocated to the variables inner and outer is returned to the operating system again. Then, the control is transferred to the next statement to the function call staternent in the calling function i.e., rnain(). As there is no statement in the main function other than the function call, so the program will terminate

13.10 FUNCTIONS THAT RETURN A VALUE AND ACCEPT

ARGUMENTS

So far, we have discussed simple functions that return no value to the calling function. However, we may need a function that could return a value and arguments could be passed to it. In previous chapters, we have seen a number of such built-in functions e.g„ sqrt(), toupper(). tolower() etc. Here, wc shall learn to write these tylxs of functions in C. Let's consider the general form of function header:

 return\_type FunetionNarne  )

The return\_type specifies the data type of the value that the function returns.

Parameter\_list is a coma separated list which specifies the data type and the name of

each parameter in the list.



#incJude int Add (int: El, void maln()

 int a, b; int sum; clrscr() ;

int n2) ;

/ / clears the previous output from / /t:he screen values for ' a', and 'b' \* ) ;



sum

Add(a,

b)

;

A Textbook 0 Science

|  |  |  |  |
| --- | --- | --- | --- |
| co | ter | 217 | Class 12 |

printf ("%d + = ,  sum) ;

int Add (int n1, int n2) return ril + n2;

Suppose the user enters 12 and 15 for a and b respectively then the output of the ofthe program will be:

12 + 15 - 27

The line of code in the main function is a function call to the Add( ) function: The Add( ) requires two paranrters of type int to be passed to it. In the function call, we have passed two variables i.e. a and b Of type int to the function. These arguments (i.e„ variables a and b) are called actual arguments or actual parameters of the function. These are local variables and their scope is the body of main function. Whereas the paranrters specified in the function header nl and n2) are called formal arguments or formal parameters of the function and their scope is the body ofAdd function. These are also called dummy arguments.

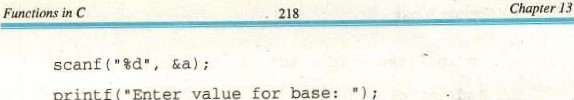
When parameters are passed to a function, the value of actual paranrters is copied in the formal parameters of the functions. The function uses its formal parameters for processing data passed to it. Any change rnade to the value of formal parameters does not affect the value of actual paranrters. Here, the values of a and b are copied in n] and n2  The function Add returns the sum of the two values to the main function which is then assigned to the variable sum.



#include float (int base, int altitude) ; void main ( )

int a, b; float area;

printf( "Enter value for altitude: 



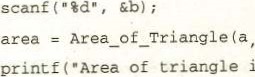
value

for

printf(

"Enter

b) ; area) ; float int altitude)



;

is

%

.2f",



(int

base,

return

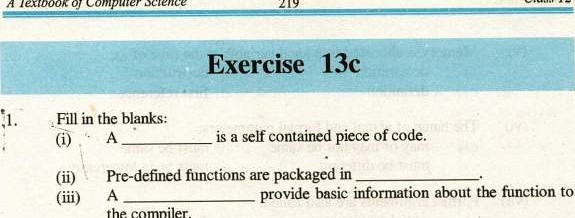
;

Output

Suppose the user enters 25 and 45 for altitude and base respectively, then the output of the program will be:

Area of triangle is S62 .50

the compiler.



A

Textbook

of

Computer

Science

Class

12

(iv) The duration for which a variable exists in nemory is called its

 of a variable refers to the region of the program where it

can be referenced.

(vi)variables are declared outside all blocks.

1. A function can not return Imre than value(s) through return

statement.

1. The paranrters specified in the function header are called

parameter.

1. The paranrters passed to a function in tlE function call are called parameters.

(X) Functions help to achieveprogramming.

1. Choose the correct option:



(i) Function prototypes for built-in functions are séecified in:



 a) source files header files

* + 1. object files d) image files

Global variables are created in:

 b) ROM

* 1. Which of the following is true about a function call?



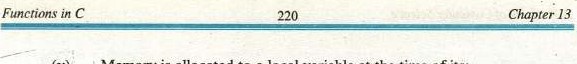
* + 1. Stops the execution of the program
    2. Transfers control to the called function Transfers control to the main function

d) Resu11Es the execution of program

* 1. Which of thefollowing looks for the prototypes of functions used in a

 programi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | a) | h) | | c) conviler | d) | |  |



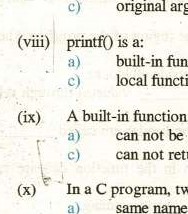
* 1. Memory is allocated to a local variable at the titne of its:

 a) declaration b) destniction

* + 1.  d) first reference
  1. The name of actual and formal pararneters:

|  |  |
| --- | --- |
| a) may or may not sanr | h) must sanr |
| c) must be different  (vii) Formal argunrnts are also called: | d) must be in lowercase |
| a) actual argurrrnts | b) dummy argunrnts |

original arguments d) referenced arguments



function

function

function:

a)

* + - 1. user-defined function keyword 

can not be redefined b) can redefined can not return a value d) should be redefined

In a C program, two functions can have:

same pararrrters

* + - 1. san-r nanr and same pararneters
      2. same nanr but different parameters

1. Write T for true and F for false statenrnt.
   1. In C, argunrnts can be passed to a function only by value.
   2. There can be multiple main functions in a C program.
   3. A function can be called anywhere in the program.
   4. In C, every function must return a value. 
   5. A user-defined function can not be called in another user-defined function.

(vii A function can called only once in a program

(vii) Scope of a local variable is the block in which it is defined.

(viiii Global variables exist in nrmory till the execution of the program.

(W, An unstructured program is more difficult to debug than a structured program.

Function body is an optional part of the fimction.

1. What is a function? How many types of functions are used in C? Discuss the difference between them.
2. Differentiate the following:

Function Definition and Function Declaration Global and Local variables

* 1. Scope and Lifetinr of a variable
  2. Function prototype and Function header

Formal parameters and Actual paranwters of a function

1. How is a function call made in a C program? Discuss briefly.
2. Answer the following:

What is the purpose of a function argument?

How many (maximum) values can a function return using return statement?

* 1. When is a function executed, and where should a function prototype and function definition appear in a source program? (iv) Write three advantages of functions.

|  |  |  |  |
| --- | --- | --- | --- |
| A Textbook o co | er Science | 221 | Class 12 |

1. Write a program that call two functions Draw\_Horizontal and Draw\_Vertical to construct a rectangle. Also write functions Draw\_Horizontal to draw two parallel horizontal lines, and the function Draw\_Vertical to draw two parallel vertical lines.
2. Write a program that prompts the user for the Cartesian coordinates of two points (Xl, Y') and (x2, Y2) and displays the distance between them. To compute the distance, write a function nanrd Distance( with four input parameters. The function Distance( ) uses the following distance formula to compute the distance and return the result to the calling function:



1. Write a program that prompts the user to enter a number and then reverse it. Write a function Reverse to reverse the For example, if the user enters 2765, the function should reverse it so that it becomes 5672. The function should accept the numlrr us an input paranrter and return the reversed number.
2. Write a function nanrd Draw\_Asterisks that will print asterisks according to the pattern shown in the following and make a function call from the function main to print the asterisks pattern.



1. Write a function Is\_Prirne that has an input paranrter i.e num, and returns a value of I if num is prinr, otherwise returns a value of O.
2. Write a complete C program that inputs two integers and then prompts user to enter his/her choice. If the user enters I the numbers are added, for the choice of 2 the are divided, for the choice of 3 the numtrrs are multiplied, for the choice of 4 the numbers are divide (divide the larger number by the Smaller number, if the denominator is zero display an error message), and for the choice of 5 the program should exit, Write four functions Add(), Subtract(), Multiply() and Divide() to complete the task.

Write a program that prompts the user to enter a number and calls a function Factorial() to compute its factorial. Write the function Factorial() that has one input paranEter and returns the factorial of the number passed to it.

15. Write a function GCD that has two input paranrters and returns the greatest common divisor of the two numbers passed to it. Write a complete C program that inputs two nurnlrrs and call the function GCD to compute the greatest common divisor of the numbers entered.