COMPUTER PROGRAMMING AND UTILISATION
UNIT 3 – DECISION MAKING STRUCTURES

Introduction

- In real life problems are not simple and linear. Some of the actions are to be performed on fulfilment of certain conditions. Some of the actions are to be repeated for certain number of times or till a condition is satisfied.
- 'C' language provides different control structures for decision making- like if statement, if…else statement, else-if ladder, switch statement, goto statement and break statement.

if statement

- If statement checks a condition. If the condition is true then the statements are executed, otherwise they are not executed.
- The syntax of if statement is:

```
if      (condition)
        statement(s);
```

- If there are more than one statements to be executed then they are put within { } symbols.

if else statement

- In certain cases, we want to execute one set of statements, if condition is satisfied, and other set of statements if condition is false. In that case we can use if… else statement.
- The syntax is :

```
if(condition)
        statement(s)1;
else
        statement(s)2;
```

- If condition is TRUE, then statement(s)1 are executed, otherwise statement(s)2 are executed.

Program – if else

```
/* Write a program to find larger of two numbers using if statement */
    #include<stdio.h>
    #include<conio.h>
    main()
    {
            float num1,num2;
            clrscr();
            printf("Enter one number\n");
            scanf("%f",&num1);
            printf("Enter second number\n");
            scanf("%f",&num2);
            if (num1 > num2)
                        printf("%f is larger\n",num1);
            else
                        printf("%f is larger\n",num2);
    }
```

Nested if else

- As the name suggest, if is used inside if statement. The nesting can be upto any level.

```
/* Write a program to find maximum of three  numbers */
        #include<stdio.h>
        #include<conio.h>
        main()
        {
                int num1,num2,num3;
                int max;
                clrscr();
                printf("Enter three numbers\n");
                scanf("%d%d%d",&num1,&num2,&num3);
                if (num1 > num2)
                {
                        if (num1 > num3)
                                max = num1;   /* num1 largest */
                        else
                                max = num3;  /* num3 largest */
                }
                else
                {
                        if (num2 > num3)
                                max = num2;  /* num2 largest */
                        else
                                max = num3;  /* num3 largest */
                }
                printf("maximum of %d, %d and %d    is = %d\n",num1,num2,num3,max);
        }
```

Multiple if else statement

- The two level choice provided by if else statement is not sufficient when we have multiple choices. 'C' provides multiple if statement for that. The syntax is:
- if (condition1)

```
                statement(s)1;
else if (condition2)
                statement(s)2;
else if (condition3)
                statement(s)3;
        .
        .
        .
else if(conditionN)
                statement(s)N;
else
                default_statement(s);
```

```
/* Write a program which asks day number and prints corresponding  day name i.e if input is 5, it will print the fifth day of week which is Thursday.
   Sunday being the first day. */
   #include<stdio.h>
   #include<conio.h>
   main()
   {
        int day;
        clrscr();
        printf("Enter day number between (1-7)\n");
        scanf("%d",&day);
        if(day==1)
                printf("Sunday\n");
                /* equality checked with == (= symbol twice)  and not single = */
        else if(day==2)
                printf("Monday\n");
        else if(day==3)
                printf("Tuesday\n");
        else if(day==4)
                printf("Wednesday\n");
        else if(day==5)
                printf("Thursday\n");
        else if(day==6)
                printf("Friday\n");
        else if(day==7)
                printf("Saturday\n");
        else
                printf("Wrong input\n");
   }
```

Program – find character type digit, uppercase, lowercase etc

/* Write a program to check the category of given character. Digit, Upper Case, Lower Case or other symbol */

```
#include<stdio.h>
#include<conio.h>
main()
{
        char ch;
        clrscr();
        printf("Enter one character\n");
        scanf("%c",&ch);
        if (( ch >= '0') && (ch <= '9'))
                printf("The character %c is digit\n",ch);
        else if (( ch >= 'a') && (ch <= 'z'))
                printf("The character %c is Lower Case\n",ch);
        else if (( ch >= 'A') && (ch <= 'Z'))
                printf("The character %c is Upper Case\n",ch);
        else
                printf("The character %c is other symbol\n",ch);
}
```

Program – roots of quadratic equation

```
/* Write a program to find the roots of quadratic equation ax2 +bx +c =0.
    if b2–4ac =0, only one root,  if b2–4ac >0, two roots,  if b2–4ac <0, no roots. formula is x = –b ± (2a)  */
    #include <stdio.h>
    #include <conio.h>
    #include<math.h> /* sqrt() function used. sqrt()  finds square root of  number*/
    main()
    {
    float a,b,c,d;
    float r1,r2,img; /* img stores imaginary part  if b*b-4*a*c <0 */
    clrscr();
    printf("Give values of co-efficients a,b and c\n");
    scanf("%f%f%f", &a,&b,&c);
    if ((a == 0) && (b !=0))  /* If a=0 then equation is bx +c =0 so, x = -c/b */
                {
                printf("Single root = %f\n", -c/b);
                }
      else
                {
                d = b*b - 4*a*c; /* calculate d */
```

Program – roots of quadratic equation (cont)

```
        if (d>0)       /* real and distinct roots */
                        {
                        r1 = (-b + sqrt (d))/(2*a);
                        r2 = (-b - sqrt (d))/(2*a);
                        printf("Real and distinct roots are:  %7.2f %7.2f\n",r1,r2);
                        }
        else if( d==0) /* real and equal i.e only one  root */
                        {
                        r1 = -b/(2*a);
                        printf("Real and equal roots are: %7.2f %7.2f\n",r1,r1);
                        }
        else           /* imaginary roots */
                        {
                        r1 = -b/(2*a);
                        img = sqrt(-d)/(2*a);  /* sqrt of negative not defined */
                        printf("Imaginary roots\n");
                        printf("Roots are: %7.2f  + %7.2fi\n",r1,img);
                        printf("Roots are: %7.2f  - %7.2fi\n",r1,img);
                        }
        }
}
```

switch statement

- Multiple if statement is difficult to manage if there are many choices to handle. 'C' provides a better way for handling multiple choices by switch statement. The syntax is :
- switch (variablename or expression)

```
{
case value1 :        statement(s)1;
                                  break;
case value2 :        statement(s)2;
                                  break;
            .
            .
            .
case valueN :        statement(s)N;
                                  break;
default  : default_statement(s);
}
```

switch statement (cont)

- Writing the default value and its corresponding default_statement(s) is optional.
- The value of expression or variable written after the switch statement must be either character or integer value.
- In multiple if…else statement, we have many conditions, while in switch statement, we have only one expression.

Program week day number to name using switch statement

/* Write a program which asks day number and prints corresponding day name. */

```
#include<stdio.h>
#include<conio.h>
main()
{
        int day;
        clrscr();
        printf("Enter day number between (1-7)\n");
        scanf("%d",&day);
        switch(day)  /* day variable has only integer value*/
                {
                case 1:
                        printf("Sunday\n");
                        break;          /* break statement here*/
                case 2:
                        printf("Monday\n");
                        break;          /* break statement here*/
                case 3:
                        printf("Tuesday\n");
                        break;          /* break statement here*/
```

Program week day number to name using switch statement (cont)

```
        case 4:
                printf("Wednesday\n");
                break;      /* break statement here*/
        case 5:
                printf("Thursday\n");
                break;      /* break statement here*/
        case 6:
                printf("Friday\n");
                break;      /* break statement here*/
        case 7:
                printf("Saturday\n");
                break;      /* break statement here*/
        default:
                /* case where value is outside range 1 to 7 */
                printf("Wrong input\n");
        }
}
```

Program – month number to number of days

```
#include<stdio.h>
#include<conio.h>
main()
{
        int month;
        clrscr();
        printf("Give month number between (1-12)\n");
        scanf("%d",&month);
        switch(month)
                {
                        /*      number of days in month 1,3,5,7,8,10 and
                                12=31 so same case for all  */
                case 1:  case 3: case 5:  case 7:  case 8:  case 10:  case 12:
                        printf("Number of days in month
                                        number %d is = 31\n");
                        break;
                case 4:      /* number of days in month 4,6,9 and 11=30
                    so same case for all */
```

Program – month number to number of days (cont)

```
case 6: case 9:  case 11:
                        printf("Number of days in month
                                number %d is = 30\n");
                        break;
                case 2:
                        printf("Number of days in month
                                number %d is = 28 or 29\n");
                        break;
                default:
                        printf("Wrong input\n");
                        break;
                }
        }
```

break statement

- It causes an exit from switch body.
- If we do not write break statement after case statement, then next switch statements are executed thus giving wrong answer.

default statement

- If any of the case statements in switch do not match, then the statements written in default part are executed.
- This statement is optional.
- Normally, it is written in switch statement after all cases are over.

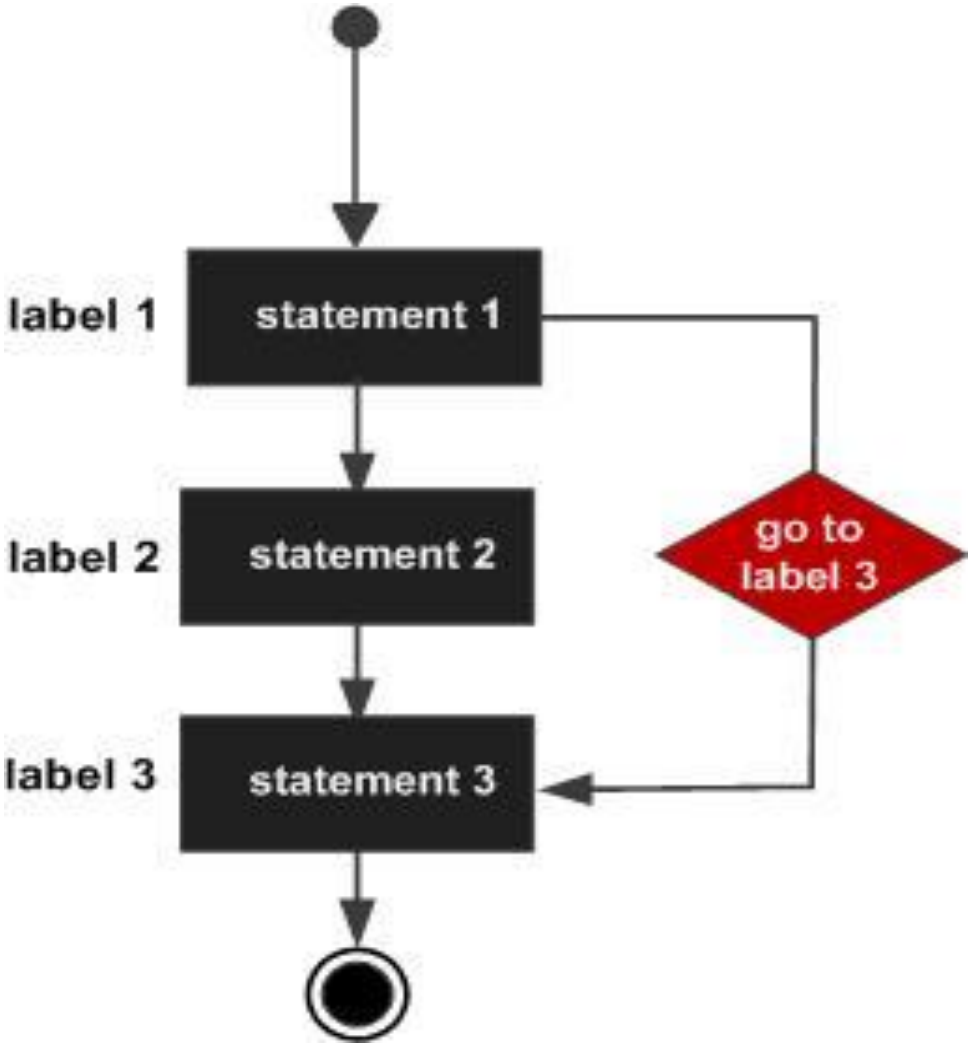
/* Write a program to check the entered character is vowel or not */

```
#include<stdio.h>
#include <conio.h>
main()
{
        char ch;
        clrscr();
        printf("Enter one character\n");
        scanf("%c",&ch);
        if( ((ch >= 'a') && (ch <='z')) || ((ch>='A')  && (ch<='Z')))
        {
        switch(ch)
                {
                case 'a':   case 'A':   case 'e':   case 'E':   case 'i':   case 'I':
                case 'o':   case 'O':   case 'u':   case 'U':
                        printf("Entered character %c is vowel\n",ch);
                        break;
                default:
                        printf("Entered character %c is not vowel\n");
                }
        }
        else
                printf("Entered character %c is not from alphabet\n",ch);
}
```

```
// Performs addition, subtraction, multiplication or division depending
the input from user

# include <stdio.h>

int main() {

    char operator;
    double firstNumber,secondNumber;

    printf("Enter an operator (+, -, *,): ");
    scanf("%c", &operator);

    printf("Enter two operands: ");
    scanf("%lf %lf",&firstNumber, &secondNumber);

    switch(operator)
    {
        case '+':
            printf("%.1lf + %.1lf = %.1lf",firstNumber, secondNumber,
firstNumber + secondNumber);
            break;

        case '-':
            printf("%.1lf - %.1lf = %.1lf",firstNumber, secondNumber,
firstNumber - secondNumber);
            break;

        case '*':
            printf("%.1lf * %.1lf = %.1lf",firstNumber, secondNumber,
firstNumber * secondNumber);
            break;

        case '/':
            printf("%.1lf / %.1lf = %.1lf",firstNumber, secondNumber,
firstNumber / secondNumber);
            break;

        // operator doesn't match any case constant (+, -, *, /)
        default:
            printf("Error! operator is not correct");
    }

    return 0;
}
```

goto syntax and flow

```
goto label;
... .. ...
... .. ...

label:
... .. ...
... .. ...
```



Example - goto

```
int main()
{
  int age;
  Vote:
     printf("you are eligible for voting");
  NoVote:
     printf("you are not eligible to vote");
  printf("Enter you age:");
  scanf("%d", &age);
  if(age>=18)
  goto Vote;
  else
  goto NoVote;
  return 0;
}
```

Example

```
#include<stdio.h>
Void main()
{
    double x, y;
    read:
    scanf("%lf", &x);
    if (x<0) goto read;
    y = sqrt(x);
    printf("%lf %lf", x, y);
    goto read;
    }
```

COMPUTER PROGRAMMING AND UTILISATION
UNIT 3 – LOOPING CONTROL STRUCTURES

Introduction

- When we have some steps to be performed repeatedly, we need to use looping control structures in our program.
- Looping control structures help our program make more readable, compact and more structured.
- The looping structures provided in 'C' language are:

  while loop,

  do…while loop

  for loop.
- Which type of looping statement to use depends on the situation.

Introduction (Cont)

- The loop is controlled by a condition. The condition decides how many times the statements will be executed.
- If there are more than one sentence (which will normally be the case) in the loop, then those sentences are put in { } symbols.
- The statements within { } symbols are called body of the loop. We can say that there are two parts of loop : **condition** and **body**.

Types of loops

- Depending on where the condition is checked, we can have two types of loop structures:
    1. **Entry Control**
    2. **Exit Control**
- In entry control loop, the condition is written first and then the body of statements.
- While in exit control loop, the body of statements is written first and then condition is written.
- This means that body of statements in exit control loop will be executed at least once.

While loop

- While loop is helpful, when we do not know in advance, how many times the statements are to be repeated. The syntax of while loop is:

```
while (condition)
{
        statement(s);
}
```

  The statements within the { } symbols are known as body part of the loop.
- Here, the condition is checked first and then only the statements are executed. So, while loop is **entry controlled** loop.
- The statements in the body part will be executed till the condition is TRUE. As soon as the condition becomes false, the control will come out of the while loop.
- If the condition remains TRUE always then, it will lead to infinite loop. So, writing a valid condition is important.

Program – First N integer numbers

```
#include<stdio.h>
#include<conio.h>
main()
{
        int n,sum=0;
        int i=1;
        clrscr();
        printf("Give Integer number:  ");
        scanf("%d", &n);
        while(i<=n)
        {
                sum = sum + i;
                i++;
        }
        printf("Sum of first %d numbers = %d\n", n, sum);
}
```

Program – Separate the digits of a given number

```
#include<stdio.h>
#include<conio.h>
main()
{
        int n,i;
        clrscr();
        printf("Give Integer number:  ");
        scanf("%d", &n);
        while(n != 0)
        {
                i = n %10;              /* separate the digit */
                printf(" The digit is = %d \n",i);
                n = n/10;  /* new value of n after separation of digit */
        }
}
```

Program – Reverse a number

```
#include<stdio.h>
#include<conio.h>
main()
{
        int num, i;
        clrscr();
        printf("Give integer number\n");
        scanf("%d", &num);
        printf("reverse of %d is =", num);
        while(num != 0)
        {
                i = num%10;
                printf("%d", i);
                num = num/10;
        }
}
```

Program – Armstrong number

- Armstrong number is equal to summation of cubes of its individual digits.
- Here, we will use the logic of previous program to separate the individual digits and then sum the cubes of individual digits.

```
#include<stdio.h>
#include<conio.h>
main()
{
        int n,sum =0;              /* sum initialized to 0 */
        int i,num;    /* separated digit stored in i */
        clrscr();
        printf("Give Integer number:  ");
        scanf("%d",&n);
        num =n;     /* n stored in num, because n
                                   will change in loop */
```

Program – Armstrong number (cont)

```
while(n!=0)
{
    i = n %10;              /* separate the digit */
    sum = sum + i*i*i; /* find cube of digit and add to sum */
    n = n/10;  /* new value of n after separation of digit */
}
if(sum == num)
                printf("Given number %d is armstrong\n",num);
else
                printf("Given number %d not armstrong\n",num);
}
```

Program – first N prime numbers

- Prime number is a number which is divisible by 1 and itself only.
- Logic for checking prime, we need only check the divisibility of that number by upto square root of that number. First prime number is 2, remaining prime numbers can be odd only.
- For example, 2, 3, 5, 7, 11 etc are prime number

```
#include<stdio.h>
#include<conio.h>
#include <math.h>
main()
{
        int i, n, count, temp;
        int num;
        /* count stores the numbers displayed, num stores the current number checked
for prime, temp is a temporary variable stores sqrt(num). i starts from 2 goes maximum upto
temp */
        clrscr();
        printf("How many prime numbers?\n");
        scanf("%d", &n);
        printf("The prime numbers are\n");
```

```
        num =2;
        printf("%4d",num);
        count =1;
        num++;
        while(count < n)
        {
                temp = sqrt(num);
                i = 2;
                while (i <= temp)
                {
                        if (num % temp == 0)
                                break;
                        i++;
                }
                if (i >temp)
                {
                        printf("%4d",num);
                        count++;
                }
                num = num +2;
        }
}
```

Fibonacci Numbers

- Fibonacci number is a number which is a summation of previous 2 numbers in series.
- 0 and 1 are first two Fibonacci numbers.
- For example, 0, 1, 1, 2, 3, 5, 8, 13 etc is Fibonacci series
- This program can also be written using while loop.

Program – sum of series

```
/* Write a program to compute the sum of following series
     1- 1/2 + 1/3 - ....+ 1/n  */
   #include<stdio.h>
   #include<conio.h>
   #include<math.h>            /* use of pow() function */
   main()
   {
         int n;
         float sum =1.0;       /* sum initialized to 1 */
         int i=2;
         clrscr();
         printf("Give Integer number:  ");
         scanf("%d",&n);
```

Program – sum of series (cont)

```
        while(i <= n)
        {
        /* pow(x,y) function calculates x^y.    In, pow(-1,i) will be
             positive for even value of i and will be negative for
             odd value of i */
                  sum = sum - pow(-1,i)/i;
                  i++;
        }
        printf("Summation of given series = %7.2f\n", sum);
}
```

do while loop

- do…while is an exit control loop, where the condition is checked later. At least once the body of the loop will be executed. Whenever you want to make sure that loop statements must execute at least one time, you should use do…while loop instead of while loop.
- The syntax is :

```
do
{
        statement(s);
} while (condition);
```

- As soon as the condition evaluates to FALSE, the loop terminates and the next statement after do…while loop is executed.
- We can always use do…while loop wherever we have used while loop. It means that all the program written in previous section using while loop can always be written using do…while loop.

Program – sum of series

```
/* Write a program to compute the sum of following series 1– 1/2 + 1/3 - ....+ 1/n */
    #include<stdio.h>
    #include<conio.h>
    #include<math.h>    /* use of pow() function */
    main()
    {
            int n;
            float sum =1.0;             /* sum initialized to 1 */
            int i=2;
            clrscr();
            printf("Give Integer number:  ");
            scanf("%d",&n);
            do
            {
                    sum = sum - pow(-1,i)/i;   /* pow(x,y) function calculates x^y */
                    i++;
            }
            while(i <=n);
            printf("Summation of given series = %7.2f\n", sum);
    }
```

Program – sum of digits of a number

```
/* Write a program to sum the individual digits of a given positive number.  For example, sum of
   digits of 1234 is 1+2+3+4 = 10  */
   #include<stdio.h>
   #include<conio.h>
   #include<math.h>
   main()
   {
        int n,sum=0;
        int digit;
        clrscr();
        printf("Give Positive Integer number:  ");
        scanf("%d",&n);
        printf("Given number = %d\n",n);
        if (n <0 )
                printf("Please give positive number\n");
```

Program – sum of digits of a number (cont)

```
        else
        {
                do
                {
                        digit = n %10;
                        sum = sum + digit;
                        n = n/10;
                } while (n >0);
                printf("Summation of individual digits = %d\n", sum);
        }
}
```

Program – reverse a number and check for palindrom

- Palindrome is a string which is same after reversal also.
- For example, 121 ,222 is palindrome, while 123 is not palindrome.

```
#include<stdio.h>
#include<conio.h>
main()
{
        int num,temp;
        int i,rev=0;
        clrscr();
        printf("Give integer number\n");
        scanf("%d",&num);
        printf("reverse of %d is =",num);
        temp = num;
```

Program – reverse a number and check for palindrom (cont)

```
        while(num != 0)
        {
                i = num%10;
                rev= rev*10 + i ;
                num = num/10;
        }
        printf("%d\n",rev);
        if (temp == rev)
                printf("Given number %d is palindrome\n",temp);
        else
                printf("Given number %d is not palindrome\n",temp);
}
```

for loop

- When we do not know number of iterations required, we can use while or do…while loop. But, in certain cases, we need to execute some steps certain number of times. In such situations where we know the number of iterations in advance, then we can use for loop.
- The syntax of for loop is :

```
for(initialization; condition; increment/decrement)
        {
                statement(s);
        }
```

- We require one variable which is called as control variable. The control variable is initialized in initialization expression. This statement executes only once.
- The condition expression actually checks the value of control variable. If the condition expression is TRUE, then the statements written are executed. After every execution of statements, the last expression increment/decrement is executed. Then again the condition is checked and body is executed if the condition evaluates to TRUE. Loop terminates when the condition evaluates to FALSE.

for loop (cont)

- Following for loop prints 1 to 10 each on separate line.
- for(i=1;i<=10,i++)

  printf("%d\n",i);
- What is the output of Sample program?
- main()

  {

  int n=6, t=1;

  for (; n<10; n = n+2)

  printf("%d %d\n" ,n, ++t);

  }

Program- square and cube of numbers

```
#include<stdio.h>
#include<conio.h>
main()
{
        int i;
        int sq, cu;
        clrscr();
        printf("Number\tSquare\tCube\n");
        printf("======\t=====\t====\n");
        for(i=1;i<=10;i++)
                {
                sq = i *i;
                cu = sq *i;
                printf("%d\t%d\t%d\n",i,sq,cu);
                }
}
```

Program - factorial

```
/* Write a program to find factorial of a given number */
    #include<stdio.h>
    #include<conio.h>
    main()
    {
            int i,n;
            long int fact;
            clrscr();
            printf("Give positive number for which factorial is to be found\n");
            scanf("%d",&n);
            if ( n<0)
                    printf("Factorial of -ve number not defined\n");
```

Program – factorial (cont)

```
        else
        {
                fact =1;
                for(i=1;i<=n;i++)
                        fact = fact *i;  /* only one statement,    so { } not required */
                printf("Factorial of %d= %ld\n",n,fact);
        }
}
```

Program – sum of series

- Calculate sum of following series

$$\frac{1}{x} - \frac{2}{x2} + \frac{3}{x3} - \frac{4}{x4} \ldots\ldots$$

Here, alternate terms have different sign. The next denominator is found by multiplying previous value with x, while numerator is the term number. For example, numerator in 3rd term is 3, for 5th term it is 5.

```
#include<stdio.h>
    #include<conio.h>
    main()
    {
        int i,n;
        float sum, term, x;
        clrscr();
        sum=0;                    /* initialize */
        term =1;
        printf("Give value of x\n");
```

Program – sum of series (cont)

```
    scanf("%f",&x);
    printf("Give value of n\n");
    scanf("%d",&n);
    for (i=1;i<=n;i++)
    {
            term = term *1/x;     /* get denominator */
            sum = sum + i *term;                    /* i value denotes numerator */
            term = -term;         /* alternate +/- */
    }
    printf("Sum = %f\n",sum);
}
```

Nesting of loops

- Using loop inside a loop is called nesting of loops. The nesting can be for any number of levels.
- The outer loop should not end between the starting of inner loop and ending of inner loop.

  Following example shows nesting up to 2 levels.

```
for(i=1;i<=3;i++)
{
        for(j=1;j<=3;j++)
        {
                        .
        }
}
```

- The outer loop control variable is I, inner loop control variable is j. For one value of i, all the values of j are used, then only i gets its next value.

Program – print pattern

- Print following pattern
- 1

  12

  123

  1234

  12345

  …..

  …..

Program – print pattern (cont)

```
#include<stdio.h>
#include<conio.h>
main()
{
        int i,j,n;
        clrscr();
        printf(“How many lines?\n”);
        scanf(“%d”,&n);
        for (i=1;i<=n;i++)
        {
                for (j=1;j<=i;j++)
                        printf("%d“,j);
                printf("\n“);
        }
}
```

Program – print pattern

- Print pattern
- 1

  1 3

  1 3 5

  1 3 5 7

  .....

  Here, from the pattern, it is clear that number of numbers in line i is = i

  So, this example can be solved using nesting of loops. The outside loop will count number of lines and provide newline character, while inner loop will count number of numbers in current line.

Program – print pattern (cont)

```
#include <stdio.h>
main()
{
        int i,j,n;
        clrscr();
        printf("How many lines?\n");
        scanf("%d",&n);
        for (i=1;i<=n;i++)
                {
                for (j=1;j<=i;j++)
                        printf("%d", 2*j-1);
                printf(„\n");
                }
}
```

Program- pythogarian triplet

- Write a program to generate pythagorian triplet between 1 and 25

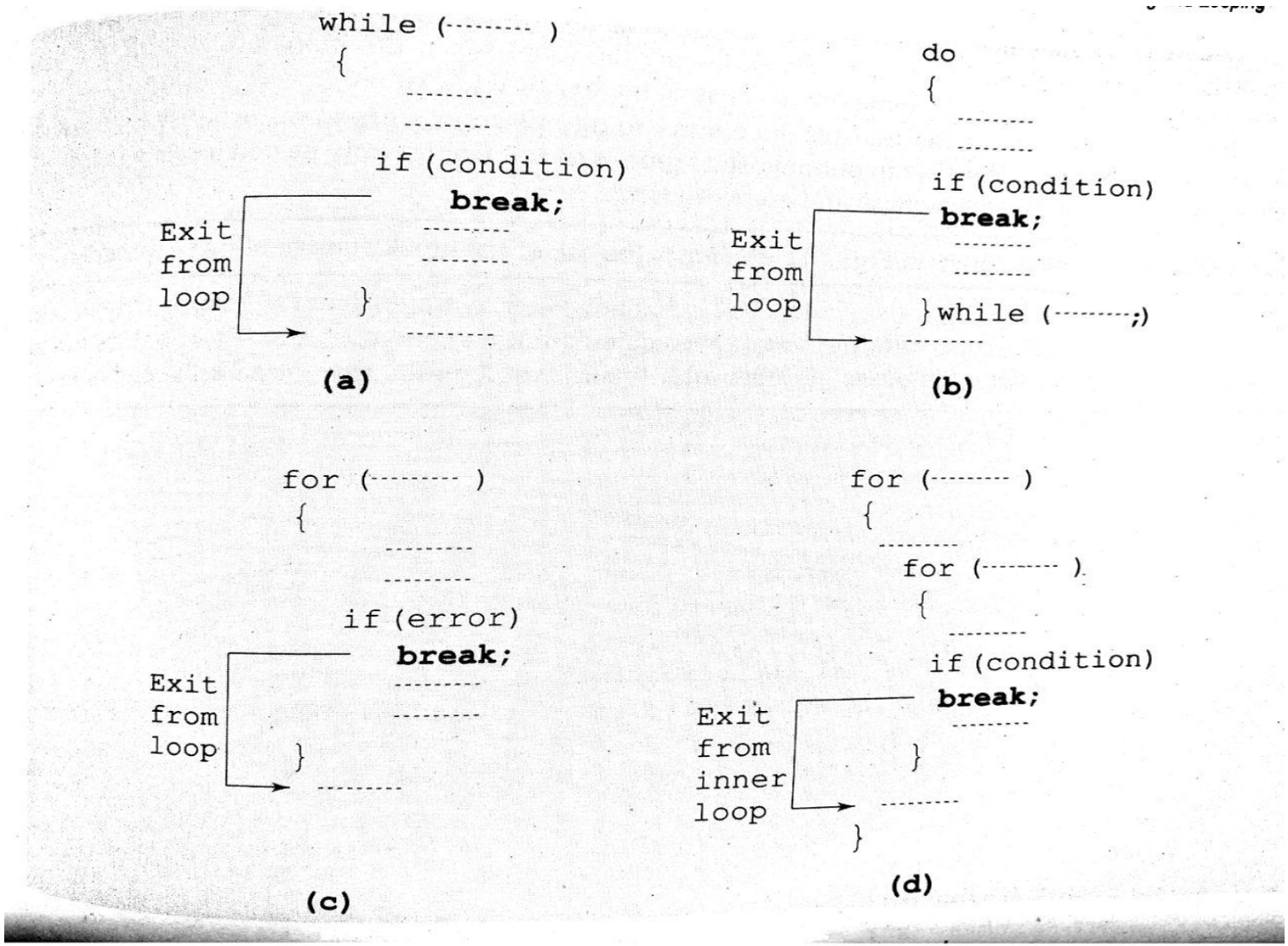
  for example, 3 4 5 is pythagorian triplet, where 3*3 + 4*4 = 5*5

```
#include<stdio.h>
#include<conio.h>
main()
{
        int i,j,k;
        clrscr();
        for (i=1;i<=23;i++)
        {
                for(j=i+1;j<=24;j++)
                {
                        for(k=j+1;k<=25;k++)
                        {
                                if (i*i + j*j == k*k)
                                        printf("%2d %2d %2d\n",i,j,k);
                        }
                }
        }
}
```

Break statement

- We can also use break statement in loops also. Whenever, break statement is encountered in a loop, the loop is terminated and control transfers to the statement immediately after the body of loop. The break statement will be executed if some condition is satisfied. When break statement executes, all the statements after that up to the end of body are skipped and loop terminates.
- In case of nested loops, if break statement is used inside the inner loop, then the inner loop terminates and the next statement following inner loop executes.
- If the break statement is used inside outer loop, then the outer loop terminates and control is transferred to the statement following the outer loop.

Break statement (cont)

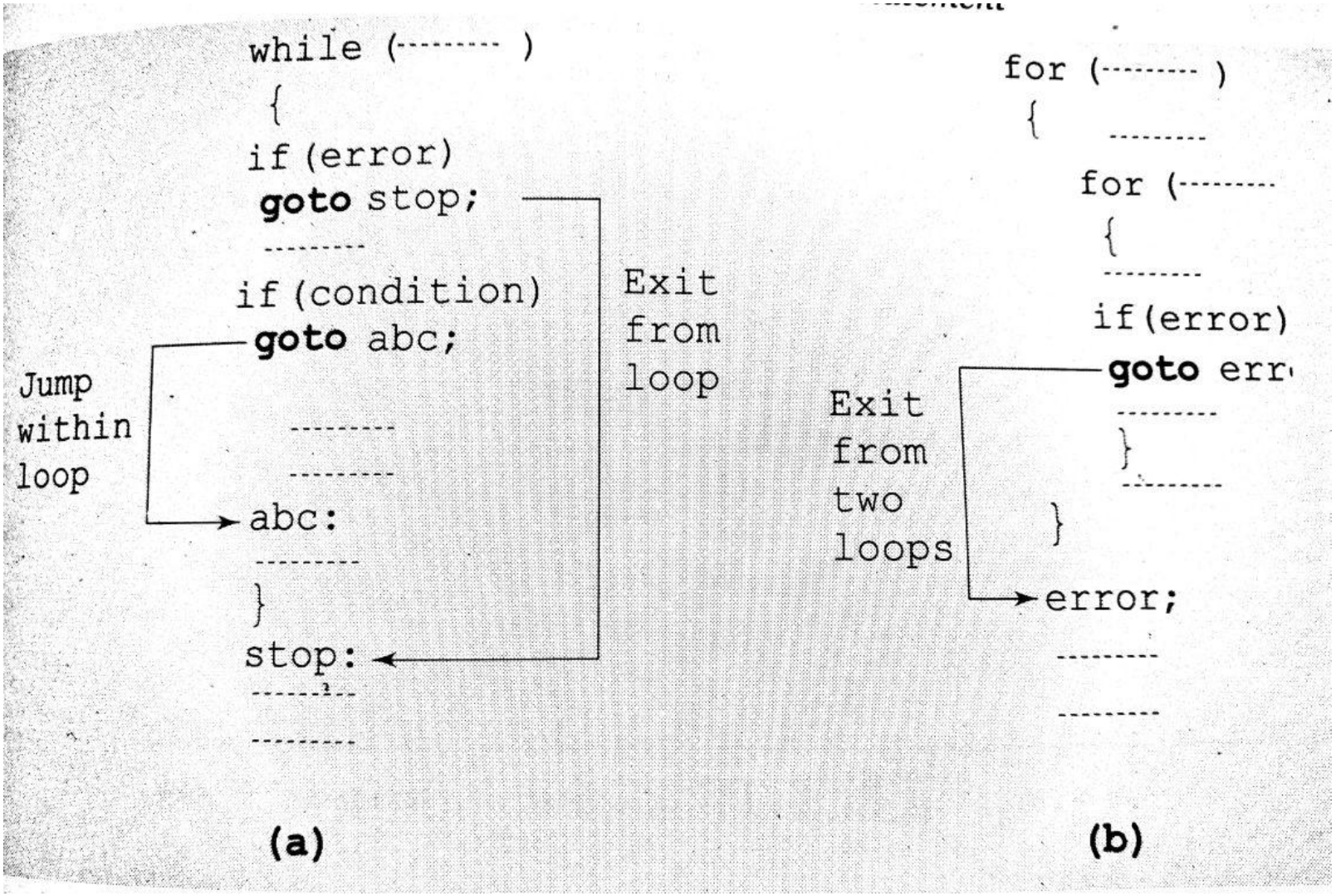


Break statement (cont)

Following example shows the break inside outer loop, so control goes to the statement outside outer loop.

```
for(  ; ;   )
{
     for(  ;   ;    )
     {
      .
      .
      }   /* inner loop over */
      .
     if (condition)
         break;  /* break outside inner loop */
      .
} /* outer loop over */
. /*control transferred outside outer loop */
```

Goto statement

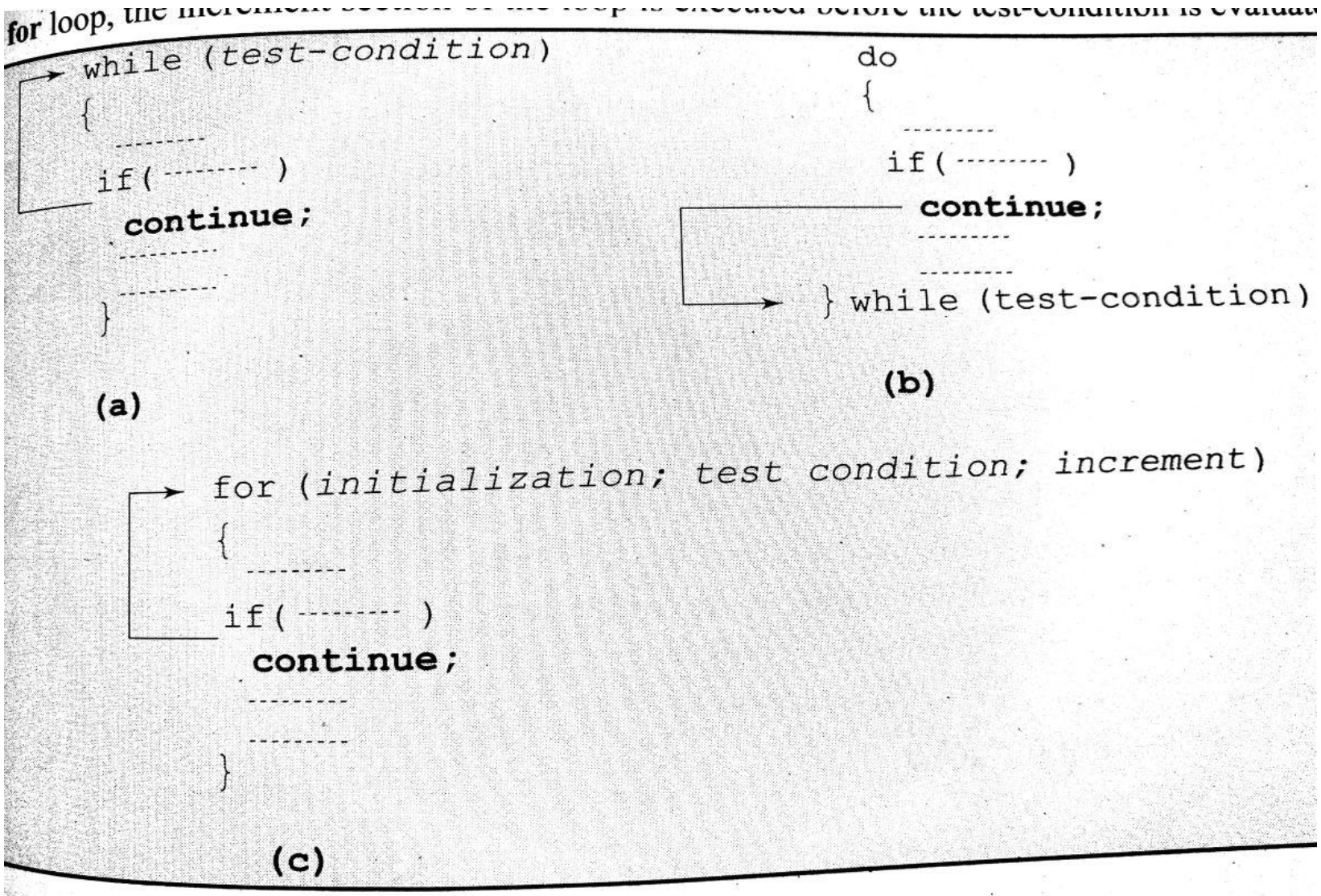


Continue statement

- Continue is also a flow breaking statement like break statement. But it works differently.
- When continue statement executes, it skips the remaining statements of current iteration and next iteration starts. Normally, continue statement is used inside if statement within the body of the loop.
- Following example shows working of continue statement.

```
for(  ;  ;    )
{
        .
        .
        if (condition)
                continue;
        .
         /* when continue executes the control comes here i.e next iteration starts */
}
```

Continue statement (cont...)



Program – to print prime numbers in given range

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
    int start, end,i,d,flag;
    clrscr();
    printf("Give start and end range numbers\n");
    scanf("%d%d",&start,&end);
    printf("Prime numbers are :\n");
```

Program – to print prime numbers in given range (cont)

```
        for (i=start; i<=end;i++)
        {
                flag =1;
                for (d=2;d<sqrt(i);d++)
                        if ( i%d == 0)
                        {
                                flag =0;
                                break;
                        }
                if(flag ==1)
                        printf("%d ",i);
        }
}
```

Program – print numbers multiple of N from given range

- For example, if N=5 and range is [17, 45] it prints 20,25,30,35,40,45.

```
#include<stdio.h>
main()
{
        int i,n;
        int start,end;
        int temp,num;
        printf("Give value of n\n");
        scanf("%d",&n);
        printf("give start and end\n");
        scanf("%d %d",&start, &end);
        temp = start/n;
        num = temp * n;
        if (num< start)
                num = num +n;
```

Program – print numbers multiple of N from given range (cont)

```
        while ( num <= end)
        {
                printf("%d ", num);
                num = num +n;
        }
}
```