

B.Tech.
First Semester Examination
Fundamentals of Computers & Programming in C
(CSE-101F)

Note : Attempt any five questions.

Q. 1. (a) What is the relation between address bus and memory capacity of a computer system.

Ans. Memory is used for storing the data & information. After processing every data or information are stored in memory. Memories are of two types:

1. Main memory
2. Secondary Memory.
1. Main memory is temporary memory device which is used for temporarily storage.
2. Secondary memory is permanent memory device which is used for permanent storage like, floppy disk, CD-ROM etc.

Computer memories have some capacity of a computer system.

$$\text{Memory capacity} = 2^{\text{address line}}$$

There are another fastest memory which is used for storing the information or data i.e. cache memory.

Q. 1. (b) Briefly describe any one printer.

Ans. Printer : Printer is the most important output device, which is used to print information on paper.

Types of Printers: Printers are mainly classified into two different types of printer:

1. Impact printers
2. Non-Impact printers

Impact Printers: The printers that print the character by striking against the ribbon & onto the paper, are called impact printers. These printers are of two types :

- (i) Character & (ii) Line printers.

Non-Impact Printers : The printers that print the characters without striking against the ribbon & onto the paper are called non-impact printers, like (i) Laser printers, (ii) Inkjet printer (iii) Thermal printer.

Impact Printers : DOT MATRIX PRINTER (DMP): One of the most **popular** category of printers in market, mainly because of their ease of printing feature & economical price is Dot Matrix printer. Each character printed is in form of pattern of DOTs & head consists of a Matrix of pins of size (5 x 7, 7 x 9, 9 x 7 or 9 x 9) which comes out of form a character, that is why it is called Dot-Matrix Printer. Printer head moves on a carriage. Paper is sandwiched between ribbon & head. Say 'A' is to be printed, then pins corresponding to 'A' moves out. Strike the ribbon & 'A' is printed. For printing, the whole line a buffer (temporary storage) of size equal to the size of paper is used. A paper of width 80 columns, a buffer of 80 characters is used. The way of printing, always in one direction from left to right is called unidirectional. New printers print bidirectionally which is achieved as : first line is printed starting from first position of buffer storage, as soon as first character is printed, 80th character of next line is stored in it & so on. After printing one, line, head starts printing from right to left using buffer positions from 80th to first again. Thus, it prints in both direction.

DMP are available in two sizes as 80 column & 132 column Printing speed varies from 200 to 360 cps character per second.

Advantages:

1. Inexpensive
2. Widely used
3. Other language characters can be printed.

Disadvantages:

1. Slow speed
2. Poor quality

Q. 1. (c) Perform the following conversions :

- (i) $(0.1102)_2 \rightarrow (?)_{10}$ (ii) $(89)_{10} \rightarrow (?)_8$
 (iii) $(140)_{10} \rightarrow (?)_{16}$ (iv) $(901)_8 \rightarrow (?)_{16}$
 (v) $(101101)_2 \rightarrow (?)_{10}$

Ans. (i) $(0.1101)_2 \rightarrow 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$

$$= \frac{1}{2} + \frac{1}{4} + \frac{1}{16} = \frac{8+4+1}{16} = \frac{13}{16} = (0.8125)_{10}$$

(ii) $(89)_{10} \rightarrow (131)_8$

8	89		
8	11	1	$\Rightarrow (131)_8$
8	1	3	
0	1		

(iii) $(140)_{10} \rightarrow (8C)_{16}$

16	140		
16	8	12 = C	$\Rightarrow (8C)_{16}$
0	8		

(iv) $(901)_8 \rightarrow (385)_{16}$

16	901		
16	56	5	$\Rightarrow (385)_{16}$
16	3	8	
0	3		

$$(v) (101101)_2 \rightarrow (45)_{10}$$

$$= 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$= 32 + 8 + 4 + 1 = (45)_{10}$$

Q. 2. (a) Define the terms:

(i) Click (ii) Double click

Ans. A mouse is commonly used input device in field of computers. Mouse was **initially developed** by Apple computers for their Macintosh computers, as a painting device. It could move an arrow like cursor on screen for selecting different options by a click.

A mouse is used along with a mouse pad. Place the mouse pad on **a flat surface & place the mouse on it**. Move the mouse on pad & the pointer moves in the **direction** of the **movement of mouse**. Various terms related to use of mouse are :

(i) Click
(ii) Doubleclick

Click : When the left button of mouse is pressed & released quickly **then this is known** as **clicking the mouse**. Before clicking a mouse the pointer is always brought to the object to be selected & a **click selects it**.

Double Click: The double clicking is used to initiate some **action on the selected item**. Basically it selects & initiates the action. For this the Enter key is pressed & released quickly twice.

Q. 2. (b) What is an operating system? Why it is necessary for a Computer System?

Ans. An operating system is a collection of programs that governs **the control of resources**, such as processor, main memory, secondary storage, Input/output device & files, so that we **can use the computer** system conveniently & efficiently. In the allocation/reallocation of the resources, **the operating systems keeps** track of the status of each resource. An operating system decides who gets a resource **for how long & when**. In other aspect one can say that an operating system is a program that acts as **an interface between user & systems hardware**.

Need of Operating System : Initially the computers were introduced as base machines. **The programs for** them were developed by manually translating squeezes of **instructions into binary or some other code** where base is usually an integer power of 2. Then the instructions & **data were entered into computer by means of** console switches or through a hexadecimal keyboard. Then address **of first instruction was loaded into** program counter to start execution of the program. Result were obtained by examining **the contents of the relevant** registers & memory locations. There was no concept of any supporting system software at all. User has to do mostly all the subcontrolling jobs manually or through program. The need was of a user who **know each & every thing** about machine. A general user was not at all able to work on computer. At that time the following needs gave birth to system softwares & development of operating system.

1. For Fast & Automatic Working: As the manual entry & working in absence of system **software result** into the slow speed of processing so a need of such a software was **felt** which could **do not of the intermediate** job automatically. So that user has to least bother about machines working, **resulting into fast & automatic** working of the machine. The Operating System automates the sequencing of operations involved in **program execution**.

2. For Better Utilization of System Resources: The base machine **approach** was **not very efficient**. With this approach, running if computer system may require **frequent manual** loading of **program & data**. This **results** in low utilization of system resources. To overcome this problem operating system **with multi user environment** were developed.

3. To realize the resource utilization potential of batch processing, a mounted batch of jobs must be executed automatically, without slow human intervention & their results into the development of operating system.

4. To provide a human a machine interface which could present a communication mode for both of them, the operating system was developed.

5. To match the speed of fast processors & comparatively slow input/output devices the need of operating systems was felt.

Q. 2. (c) Write ten features of Internet.

Ans. "Internet" refers to the global information system that:

- (i) It is logically linked together by a globally unique address space based on the internet protocol (IP) or its subsequent extensions/follow-ons;
- (ii) It is able to support-communications using the transmission control protocol.
- (iii) Provides, uses or makes accessible, either publicly or privately, high level services layered on the communication & related infrastructure.

Internet Requirements : At a very basic level, a user need the following to access the internet:

- * An internet access account from an ISP
- * A computer
- * A modem connected to a telephone line
- * Necessary software for connecting to the internet & accessing information.

There are large number of features of Internet:

1. Career Advancement: Search job listings, post resumes, interview online.

2. Actions : Sell old "tuff, acquire more stuff, with online actions.

3. Distance Learning : Attend online lectures, have discussions, research papers.

4. Download Files: Get software, music & documents such as e-books.

5. Entertainment: Amuse yourself with internet games, music & videos.

6. E-mail & Discussions Groups: Stay in touch world wide through electronic mail & online chat rooms.

7. News : Stay current on politics, weather, entertainment, sports & financial news.

8. Research & Information : Find information on any subject, using browser & search tools.

9. Telephony and Conferencing: Make inexpensive phone calls; have online meetings.

10. E-business : Connect with coworkers, buy supplies, support customers.

Q. 3. (a) What do you mean by a translator? Discuss any two translators briefly.

Ans. Language translators also called language processor. Main function of language translators are :

- 1. Language translators perform the translation of high-level language or assembly language into machine language.
- 2. They check for & identify syntax errors that may be present in the program being translated.

There are three types of translators programs : Assemblers, compilers & interpreters.

Assemblers : Assemblers convert the assembly language program into machine language program. Assemblers are of two types :

(i) Resident Assembler : An assembler which resides & runs on the same computer for which it is producing object code is called self assembler or resident assembler.

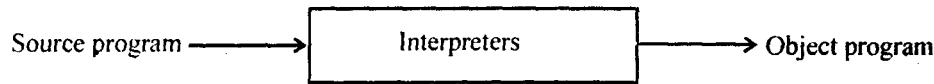
(ii) Cross Assembler : Some times the computer for which the object code is to be produced is a cheaper & less powerful. In such situations, a faster & powerful computer can be used for programming. But the object

is to be created according to the small computer. Therefore a cross assembler **run on the computer other than** its computer for which it has to produce the object program. Cross assemblers are of two types :

(a) **One-pass Assembler** : One pass assembler reads the complete **program just once, then** assign the addresses to the labels used in the source program & translates the program in machine code.

(b) **Two-pass Assembler**: Two pass assemblers read the assembly language program twice. In first pass, assembler reads the assembly source languages program & identifies **all the labels. Then these labels are** assigned the addresses with respect to their starting address. In second pass, **the assembler translates each** instruction of program to machine code & assign address to that.

Interpreter : Interpreter is another type of translator used for **translating High Level Language into** machine code. It takes one statements of a High Level Language & translates **it into a machine instruction** which is immediately executed. So in this case no object code is saved. In other words, an interpreter translates one instruction & the control unit executes the resulting machine code, the next instruction is translated & the control unit executes the machine code instruction & so on.



Each time the program is executed, every line is checked for syntax error & then converted to **equivalent** machine code directly. It requires less main memory. It is good for fast debugging & testing. **Interpreters** doesn't provide any security for source code.

Q. 3. (b) Write any five characteristics of a good algorithm.

Ans. Algorithm is step by step procedure to solve a specific problem.

There are few characteristics of a good algorithm.

(i) **Finiteness** : An algorithm should be terminated after a finite numbers of steps.

(ii) **Defeniteness** : Each step of the algorithm is precisely defined (eg. if $x > 50000$ then goto 5 this is well defined if x is very large than goto 5 this is not define.

(iii) **Generality (Completeness)**: The rule of the algorithm are complete so that it can solve **all problems** of a particular type (for any input data) for which it is designed.

(iv) **Effectiveness** : All the operations used in the algorithm are basic & are capable of being **performed** mechanically. For example, the operation A divided by B is effective if both A & B have reasonable magnitudes. In case A & B pare real numbers expressed as an infinite sequence of digits **then the operations of division may** not be effectiveness.

(v) **Input-Output**: An algorithm has certain precise inputs or initial data & the outputs are generated in the intermediate as well as the final steps of the algorithm.

Q. 3. (c) Define system software and application software.

Ans. Software is a set of computer programs which are designed & developed to perform specific task desired by the user or by the computer itself. So software makes the computer to do work. Once it is put to work it becomes possible to manipulate or process. There are two types of software, (i) System software (ii) Application software.

System Software : The system software is collection of polymers designed to operate, control & extend the processing capabilities of the computer itself. System software are generally prepared by computer manufacturers. These softwares perform a variety of functions like file editing, storage management, resource accounting, input/output management etc.

Types of System Software:

1. System Control Programs: They control the execution of programs, manage the storage & processing

resource of the computer & perform other management & monitoring functions. Example is Operating Systems."

2. System Support Programs : They provide routine service functions to other computer programs & computer users. Example is utility programs.

3. System Development Programs : They assist in the creation of publication programs. Examples are language translators like Interpreters, compilers etc.

Application Software: Application software is the program or set of programs used to perform specific & general applications. Examples of application software are pay roll software, student record software, inventory control software, spread sheets, dBASE etc.

Types of Application Software:

(i) General Purpose Application Software: Example of general purpose application software are: Word processors, spread sheets, database etc.

(ii) Special Purpose Application Software: Example of special purpose application software are : Accounting, Inventory, Production management etc.

Q. 4. (a) What are the basic data types supported by C language? What are their storage requirements?

Ans. Various quantities such as constants, variable etc. occurring in C program must have a type associated with them. There can be one & only one type associated with an entity. C supports a very large number of data types. The type of an entity establishes the following information about it.

1. Its meaning.
2. Constraints application to it.
3. Possible value of the entity.
4. Possible operations applicable on the entity.
5. Function that can be used with it.

A data type is defined as a finite set of values along with set of rules for permissible operations.

Basic Data Type Supported by C Language:

- (a) Integer(int)
- (b) Floating point (float)
- (c) Character (char)

Basic Data Types : Each types of data may be represented differently in the computers memory. Consequently, memory requirement of these data types will also be different. Various data types with the size according to 16-bits machine are given in table.

Integer Type (int):

- * int (16 bits)
- * short int (16 bits)
- * long int (32 bits)
- * unsigned int (16 bit)
- * unsigned short int (16 bits)
- * unsigned long int (32 bits)

Character Type (char)

- * char (8 bits)
- * undersigned char (8 bits)

Floating Point Type (Float)

- * float (32 bits)
- * double (64 bits)
- * long double (80 bits)

C support a number of qualifiers that can be applied to the basic data types. They are :

- Short
- Long
- Signed
- Unsigned

Also double stores double precision floating point number. An unsigned number is always +ve or zero.

With signed modifier, first bit is reserved for the sign (+ve) or (-ve) of the variable.

Range of a signed data types can be calculated according to the following formula,

$$-2^{\text{size}-1} \text{ to } +2^{\text{size}-1} - 1$$

For Example: Size of the int data type is 16. so range is -2^{15} to $+2^{15} - 1$ i.e. -32768 to +32767

Range of a unsigned data type can be calculated according to the following formula,

$$0 \text{ to } 2^{\text{size}} - 1$$

e.g., size of unsigned int datatype is 16. So range is 0 to $2^{16} - 1$ i.e. to 65535.

The use of qualifier signed on integers or characters assumes a signed data types. Hence 'int' or 'signed int' have the same meaning, size & range.

Q. 4. (b) Write a program in C that reads three integers and prints largest of them.

Ans. Program in C that reads three integers & prints largest of them.

```
#include <stdio.h>

main()
{
    int a, b, c, big;
    printf("Enter three numbers\n");
    scanf ("%d %d %d", &a, &b, &c);
    if(a>b)
    {
        if(a>c)
            big = a;
        else big = c;
    }
    else
    {
        if
        if(b>c)
            big = b;
```

```

    }
    printf ("Largest number = %d", big);
}
RUN : Enter three numbers
76    54    67
Largest number = 76

```

Q. 5. What do you mean by derived data type? Why they are needed? Write a program in C to search an element in an array using binary search? The program should be well indented and self explanatory.

Ans. Data Types: Various quantities such as constants, variable etc. occurring in a C program must have a type associated with them. There can be one & only type associated with an entity. Supports a very large number of data types. The type of an entity establishes the following information about it.

- (a) Its meaning
- (b) Constraints applicable to it.
- (c) Possible value of the entity.
- (d) Possible operations applicable on the entity.
- (e) Function that can be used with it.

A data type is defined as a finite set of values along with set of rules for permissible operations.

Derived Data Type : Derived data types are those data types whose derived from fundamental data types.

There are some derived data type :

1. Array
2. Function
3. Pointer etc.

Array : An array is a set of elements of the same data type represented by a single variable name. Each individual array element can be referred to by specifying the array name, followed by a subscript, enclosed in square brackets. Arrays may be one-dimensional or multi-dimensional depending upon whether each element of the array is identified by one or more subscripts. Thus, if makes is a one-dimensional array containing n elements, the individual array elements will be marks [0], marks [1].....marks [n-1]. Since only one subscript or index follow the variable name marks, therefore, this array is called one-dimensional array.

Figure shows the structure of a one-dimensional array.

```

Marks [0] = 40;
Marks [1] = 57;
Marks [2] = 20;

```

```

Marks [n-1 ] = 60;

```

Array **Declaration :** The general form of declaration of one-dimensional array is:

```

data type variable-name [size];

```

Data type indicates the type of values that are going to be stored in array elements, that is, it specifies the type of array elements. Size indicates the maximum number of elements that can be stored inside the array.

```

eg. float a [10];

```

The declaration defines a to be an array consisting of 10 real elements that are designated as a [0], a[1],a.[a].

Input and Output of Arrays : The entire array cannot be input/output using a single input/output state-

ment. To read/write an array, the elements of an array have to be read/write individually. The for do loop is generally used to input/output array elements.

The program given below reads the elements of an array & writes them.

```
# include (stdio.h>
main ()
{
    inti;
    float a [5];
    for (i = 0; i < 5; i ++)//* reading values into array */
    scanf("%f",&a[i]);
    for (i = 0; i < 5; i ++)//* writing of values into array elements */
    printf("%f",a[i]);
}
```

Function : A function is a self-contained block of code that performs a particular task. C function can be classified into two categories, namely, library function & user-defined functions.

1. Library functions are not required to be written by users examples of library functions are scanf, printf, sqrt, cos, etc.
2. User-defined functions have to be developed by the user.

The use of user-defined functions allows a large program to be broken down into a number of smaller-self contained components; each of which has some unique, identifiable purpose. Thus a c program can be modularized through the intelligent use of such functions.

Advantages of User-Defined Functions:

1. Repetition : A function may be used to avoid rewriting the same set of statement within the same program.
2. Universal Use : One function can be used in more than one program.
3. Debugging : Debugging becomes very simple after using function.
4. Memory: Function make efficient use of computer memory.
5. Team Work : Large program can be divided into different subprogram which can be written by the group of students or users.
6. Readability : Separating the code into function's also makes the program easier to design & understand.

The general form of all function is

```
f type f name (argument list)
argument declaration;
{
    local variable declaration;
    statement-1;
    statement-2;

    return (expression);
}
```

Pointer : A pointer is nothing but a variable that contains an address which is a location of another

variable in memory. The advantages of the pointer variables.

- * The storage size can be adjusted dynamically as desired by the program.
- * Storage may be shared among several variables.
- * Complex data structures like linked list, stack etc.

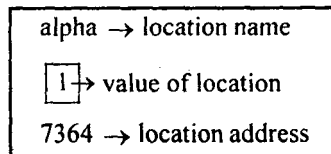
THE "ADDRESS OF" AND THE "INDIRECTION" operators.

The declaration statement,

```
int alpha = 1;
```

Tells the compiler to

1. Reserve just enough space in memory to hold an integer value.
2. Associate the name alpha with this space.
3. Store the value 1 there.



Program to search an element from list of element using binary search technique.

```
# include <stdio.h>
# include <conio.h>
void main ()
{
    int a[100], i, loc, mid, beg, end, n, flag = 0, item;
    clrscr ();
    printf ("How many elements");
    scanf ("%d", &n);
    printf ("Enter the element of the array\n");
    for(i = 0; i <= n - 1; i++)
    {
        scanf ("%d", &a[i]);
    }
    printf ("Enter the element to be search \n");
    scanf ("%d", &item);
    loc = 0;
    beg = 0;
    end = n - 1;
    while ((1 beg <= end) && (item != a[mid]))
    [
        mid = ((beg + end)/2);
        if (item == a[mid])
        {
            printf ("Search is successfull\n");
```

```

        loc = mid;
        printf ("Position of the item %d \n", loc + 1);
        flag = flag + 1;
    }

    else if (item < a [mid])
        end = mid - 1;
    else
        beg = mid + 1;
    }

    if(flag==0)
    {
        printf ("search is not successfully");
    }
    getch ();
}

```

Q. 6. (a) Differentiate between structure and union.

Ans. Structure : The structure is composed of data items that may be of different data types. The data items of a structure are called fields. Each field has an identifier (i.e. field name) & a data type.

The general format of a structure definition is

```

struct sname
{
    datatype member 1;

    datatype member n;
};

```

Where struct is a keyword also called tag.sname is the name, given to the structure data type.

Member-1.... member n are elements of the structure being defined.

The portion of the structure definition enclosed in braces is known as structure template.

Unions : Unions like structures, contain members whose individual data types may differ from **one** another. However, the members that compose a union all share the same storage area within the **computer's** memory; whereas each member within a structure is assigned its own unique storage area. Thus, unions **are** used to conserve memory. They are infect for application involving multiple members, where values need **not** be assigned to all of the members at any one time.

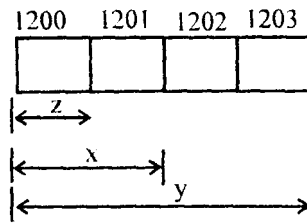
The syntax for declaring a union, declaring variables of union type, accessing elements of a union, is identical to that of structures, except for storage of members of union & their initialization.

eg.: Union sample

```

{
    intx;
    float y;
    char z;
};
union sample code ;

```



Q. 6. (b) Write a program that uses a union to maintain examination scores of a student as per following criteria:

If student has passed examination : store percentage of marks secured by student else store 'C' for compartment, 'F' for fail, 'A' for absent.

Ans. / * accept student data & print student's performance using structure */

```
# include <stdio.h>
```

```
main ()
```

```
{
```

```
    union student
```

```
    {
```

```
        int rollno;
```

```
        char name [20]
```

```
        int marks [5];
```

```
    };
```

```
    union student stud;
```

```
    char grade;
```

```
    int i, total = 0;
```

```
    float avg;
```

```
    clrscr( );
```

```
    printf("Enter the student's record An");
```

```
    printf("\n Roll no :");
```

```
    scanf("%d", & stud. roll-no);
```

```
    fflush (stdin);
```

```
    printf("\n name ;");
```

```
    gets (stud, name);
```

```
    printf("\n Enter students marks : \n");
```

```
    for(i = 0; i <= 4; i++)
```

```
{
```

```
    printf("\n subject_% d: " i + 1);
```

```
    scanf("%d", & stud.marks [i]);
```

```
    total t = stud.marks [i];
```

```
}
```

```
    avg = total/5.0;
```

```
    if(avg >= 80
```

```

        grade = 'A';
    else
    {
        if(avg >= 60 && avg < 80)
            grade = 'B';
        else
        {
            if(avg >= 40 && avg < 60)
                grade = 'C';
            else
                grade = 'D';
        }
    }
    clrscr();
    printf ("\n student's performance evaluation is : \n");
    printf ("\n Roll no : %d", stud.name);
    printf ("\n Total marks : %d", total);
    printf ("\n Average : %2f", avg);
    printf ("\n Grade : % C", grade);
    getch ();
}

```

Q. 7. (a) What do you mean by recursion? Write a recursive program to calculate factorial of a number.

Ans. Recursion refers to the process in which function call itself. Recursion is a powerful technique using which some problems, can be expressed in a form which is very close to their natural statement. Although, the method is simple to understand & the function is easy to create when the problem is inherently recursive yet the information of recursive functions for other problems is difficult.

Example: Program to find the factorial of a number, Recursive definition of n factorial is

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n * (n-1)! & \text{if } n > 0 \end{cases}$$

The evaluation of 4! by this definition is shown below :

$$\begin{aligned}
 4! &= 4 * 3! \\
 &= 4 * 3 * 2! \\
 &= 4 * 3 * 2 * 1! \\
 &= 4 * 3 * 2 * 1 * 0! \\
 &= 4 * 3 * 2 * 1 * 1 \\
 &= 24
 \end{aligned}$$

```

#include <stdio.h>
main ()
{
    int n;

```

```

        printf ("\n Enter a number:");
        scanf ("%d",&n);
        if (n < 0)
            printf ("Invalid number \n")
        else
            printf ("%d! = %d", n, fact (n));
    }

    fact(int x)
    {
        if(x == 0)
            return 1;
        else
            return (x * fact (x - 1));
    }

```

Q. 7. (b) What are the storage classes of C variables? Briefly explain each.

Ans. Every variable in C possesses a characteristic called is storage class. To fully define a variable we need to define not only its type but also its storage class.

By using the variables with the appropriate storage classes, we can write programs that use memory more efficiently, run faster & are less prone to programming errors.

There are four Storage Classes in C :

1. Auto (automatic) storage class
2. Static storage class
3. Extern (external) storage class
4. Register storage class

By default, storage class is auto. Therefore, if we do not specify the storage class of a variable then it is assumed of type auto.

The storage class defines the following characteristics of the variable.

1. Storage : We can store a variable either in the memory or the CPU registers. A storage class of a variable tells we where the variable should be stored.

2. Default Initial Value: If the initial value of a variable is not specified in the declaration statement, then storage class tells we what will be the initial value of the variable.

3. Lifetime : The lifetime of a variable is the length of time it retains a particular value.

4. Scope : The scope or visibility of a variable refers to those parts of a program which will be able to recognize it.

Auto Storage Class : We can define a local variable by auto specifier. However this specifier is seldom used because a local variable is by default auto.

The variables declared with auto storage class have the following features :

- 1. Storage:** Stored in memory.
- 2. Default Initial Value:** If not initialized in the declaration statement, their initial value is unpredictable, which is often called garbage value.
- 3. Local (Visible)** to the block in which the variable is declared.
- 4.** It retains its value till the control remains in the block in which the variable is declared.

Example:

```
# include <stdio.h>
main ()
{
    auto int i;
    printf ("value of i = %d, i);
}
```

Run: Value of i = 689

This example shows the fact that if the variable is not initialized, it contains a garbage value. Most of the times, for un-initialized variables, we will obtain value zero.

Static Storage Class : If one desires that the local variables should retain its value even after the execution of the function in which it is declared, then the variable must be declared as static. The value of a static variable is retained & is available to the user even if the control returns back to the function again.

The variable declared with static storage class have the following features :

- Stored in memory.
- If not initialized in the declaration statement, their initial value is zero.
- Local (visible) to the block in which the variable is declared.
- It retains its value between different function calls. That is, when for the first time a function is called, a static variable is created with initial value zero, & in subsequent calls it retains its present value.

3. Extern Storage Class: The scope of external variable is global. External variables are declared outside all function i.e. in the beginning of the program file. These variables are visible to all functions in a program file. However, if we have written our functions in separate program files, then we have to declare that variable again with extern storage class to tell the compiler that this variable or these variables are external variables & are defined in other source files.

The variables declared with extern storage class have the following features :

1. Stored in memory.
2. If not initialized in the declaration statement, their initial value is zero.
3. Global i.e. visible to all functions.
4. Retains its value till the program execution terminates.

Register Storage Class: We know that a variable is nothing but a memory location in main memory of the computer of the CPU has to access the main memory to manipulate the variable. This activity slows down the execution of the program because main memory is about 10 times slower than the CPU. Since, a number of registers are available in CPU itself & all of them are not used all the time, a few of them can be used to store some frequently used variables so that the execution becomes faster.

The variables declared with register storage class have the following features :

1. Stored in processor registers, if a register is available. If no register is available the variable is stored in memory, & works as if its storage class is auto.
2. If not initialized in the declaration statement, their initial value is zero.
3. Local to the block in which the variable is declared.
4. It retains its value till the control remains in the block in which the variable is declared.

Q. 8. (a) What do you mean by a library? How they are useful for the programmers?

Ans. A library is a collection of program or functions which can be used by other programs. Like all other high level languages, C provides function libraries containing built-in functions. These standard functions are

For more study material Log on to <http://www.ululu.in/>

extremely useful to the programmers in the sense that if some of the available functions are called in a program then they are combined with the program code by the linker at the time of linking. The advantages of library functions are listed below :

1. Reusability of Code: The program development activity requires three major activities: coding, testing & debugging. If a function, developed previously by self or some body else, is available as a standard library function then the effort of rede velopment can be avoided.

2. Faster Development : Since many useful functions are available to the programmer, the program development process becomes faster & the productivity increases.

3. Easier Software Maintenance : Since a function available in the library is shared by other users, the changes or modifications in the function can be done at library level. Thus, the software maintenance becomes easier.

There are two types of libraries provided by C : Standard C libraries & user libraries.

Standard C Libraries: The declarations of various library function are maintained by C in the form of the header files. In other words we can say that each function library has a corresponding header file with extension, h. For instance, stdio.h is a header file.

The required header files are included in the program with the help of angle brackets '<' and '>'. Infact, the angle brackets make the c computer to look for the header file into its induce directory.

Standard Library Functions: C supports a rich library of standard functions. The access to a function is allowed by including its corresponding header file at the beginning of the program.

For Example : getchar (), putchar (), gets (), puts (), strcpy (), fflush () etc.

Some more useful functions are given below :

Strcpy () : This function copies one string to another. The function prototype of this function is contained in string.h header file. The general form of this function call is given below :

strcpy (dest, source)

Strcat () : It appends a source string to the end of dest string so that the length of the resulting string is equal to total of both the string.

Strlen () : This is also a string.h related function. The argument to this function is string. It calculates the length of that string i.e. it returns the number of characters, in the string except the null character.

sin()
sqrt()
pow()
cos()
log() etc.

Q. 8. (b) Explain the following with syntax:

(i) fread (ii) fscanf (iii) fprintf
(iv) fopen (v) fwrite

Ans. (i) fread () function : This function is used to read the records of the file.

The syntax is

fread (<& rec >,< bytes >,< n >,< fp >:)

Where rec is the name of the record, bytes is the number of bytes required to store the record, n is the number of records to be read & fp is the file pointer. This statement assumes that the file is already open in the binary mode for reading by the following statement:


```
fp=fopen("xyz","rb");
```

Write a program which reads the employee name, employee number & salary from the file named "xyz".

```
#include <stdio.h>
struct employee
{
    int eno;
    char name [20];
    int salary;
}
main ()
{
    struct employee e;
    char ans;
    FILE *fp;
    fp = fopen("xyz","rb");
    if(fp = NULL)
    {
        printf ("error in file opening");
        exit(0);
    }
    while (!feof (fp))
    {
        /* reads information from file */
        fread (&e, sizeof (e), 1, fp);
        /* display information */
        printf ("/n%d% 20s% 10d /n", e.eno,e.name, e.salary);
    }
    fclose (fp);
}
```

(ii) fscanf () function : This function reads characters from a file & converts the string into values of C variables according to the format specified. It requires three arguments, file pointer, format string & the list of variables. For example, the statement

```
fscanf (fp, "%4 d% 3d%c", x,y, z);
```

reads the record from file pointed to fp & puts it in the variable x, y & z.

The general format of fscanf (1) is

```
fscanf (<file-pointer>, "<input-format>", <variable-list>);
```

Write a program to display the file

```
#include <stdio.h>
struct employee
{
    int eno;
```

```

        char name [20];
        int salary;
    }

    main ()
    {
        struct employee e;
        int i, n;
        FILE *fp;
        fp = fopen ("xyz" "r");
        if(fp ==NULL)

        {
            printf ("Error in file opening");
            exit(0);
        }
        {
            /* reads information form file */
            fscanf (fp, "%d% 20s% 1 Od \n" &e eno, e.name, & e-salary);
            /* display information */
            printf ("%d %20s% 10d \n", e.eno, e.name, e.salary);
        }
        fclose (fp);
    }
}

```

(iii) fprintf () function: This function writes characters strings & values of c variable to file. It takes three arguments, file pointer of the file to which information is written, format-string which is to be written & the list of variables. For example, the statement

```
fprintf (fp, "% 4d % 3d% c\n", x, y, z);
```

The general syntax of the fprintf () is

```
fprintf (<file-pointer>, "<output-format>", <variable-list>);
```

Write a program to read the information from the keyboard & write it to the file.

```

#include <stdio.h>
struct employee
{
    int eno;
    char name [20];
    int salary;
}

main ()
{
    struct employee e [10];
    int i, n;
    FILE *fp;
    fp = fopen ("xyz", "w");
    if(fp=NULL)

    {

```

```

        printf ("Error in file opening");
        exit(O);
    }

    printf ("No. of employee?");
    scanf ("%d",&n);
    for(i=1;i<= n;i++)

    {
        printf ("\n Employee No.:");
        scanf ("%d, De[i].eno);
        printf ("\n Name:");
        printf ("n salary :");
        scanf ("%d",&e[i].salary);
        fprintf (fp, "%6d% 20s % 10\n,
        e [i].eno, e [i]. name, e[i].salary);
    }

    fclose (fp);
}•

```

(iv) fopen () function : A file has to be opened before any input/output operation can be performed on it. The file must have a name. The name should be, a maximum of 8 character long. There should be no characters other than the alphanumeric characters & the unders core (_) in the file name. The file name can have an extension of 3 characters.

Whenever we want to read from or write to one of files we are working with, we identify that file by using its file pointer. The file is also a type of data & it should be declared as usual. The syntax of the statement is
FILE*fp;

Where fp is a point to the file. The file is opened for input/output by function fopen (). The syntax for opening a file is,

```
* fp = fopen ("filename", "mode");
```

Where file name is the name of the file, mode determines the purpose for which file will be used.

The valid modes are given below :

Mode	Purpose
r	Open for reading. The file must already exist.
r+	Open for reading & writing. The file must already exist.
w	Open for writing. If the file already exists, its contents will be overwritten. If it does not exist, it will be created.
w+	Open for reading & writing.
a	Open for append. Data will be added to the end of the existing file, if file already exists. If it does not exists, it will be created.
a+	Open for reading & appending.

(v) fwrite () function : This function allows a blocks record to be written onto a file with a single statement. The syntax of fwrite () function is,

```
fwrite (< & rec >, <bytes>, <n>, <fp>);
```

Where rec is the name of the record to be written, bytes is the number of bytes required to store the record,

n is the number of records to be written & fp is the file pointer.

The general syntax of fwrite () is,

fwrite (< & rec>, <size of (rec)>, <n>, <fp>);

Program:

```
# include <stdio.h>
struct employee
{
    int eno;
    char name [20];
    int salary;
}

main ()
{
    struct employee e;
    char ans;
    FILE*fp;
    fp = fopen ("xyz", "wb");
    if(fp == null)
    {
        printf ("Error in file opening");
        exit(0);
    }

    do
    {
        printf ("\n Employee no.");
        scanf ("%d", & e.eno);
        printf ("\n Name");
        scanf ("s", & e.name);
        printf ("\n salary");
        scanf ("%d", & e.salary);
        /* writes information to file */
        fwrite (& e, size of (e), 1, fp);
        printf ("\n do you want to add more records /n");
        ans = getch ();
    }
    while (ans != 'n');
    fclose (fp);
}
```