Programming in C

Interview Questions and Answers for C Programming

Compiled from various sources by

Ms Arshiya Khatoon

1. WHAT DOES STATIC VARIABLE MEAN?

Ans: Static variables are the variables that retain their values between the function calls. They are initialized only once their scope is within the function in which they are defined.

2. WHAT IS A POINTER?

Ans: Pointers are variables that store the address of another variable. That variable may be a scalar (including another pointer), or an aggregate (array or structure). The pointed-to object maybe part of a larger object, such as a field of a structure or an element in an array.

3. WHAT ARE THE USES OF A POINTER?

Ans: Pointer is used in the following cases

i) It is used to access array elements
ii) It is used for dynamic memory allocation.
iii) It is used in Call by reference
iv) It is used in data structures like trees, graphs, linked list, etc.

4. WHAT IS A STRUCTURE?

Ans: Structure constitutes a super data type that represents several different data types in asingle unit. A structure can be initialized if it is static or global.

5. WHAT IS A UNION?

Ans: Union is a collection of a heterogeneous data type but it uses an efficient memory utilizationtechnique by allocating enough memory to hold the largest member. Here a single area of memory contains values of different types at different time. A union can never be initialized.

6. WHAT ARE THE DIFFERENCES BETWEEN STRUCTURES AND UNION?

Ans: A structure variable contains each of the named members, and its size is large enough tohold all the members. Structure elements are of same size.

A union contains one of the named members at a given time and is large enough to hold thelargest member. Union element can be of different sizes.

7. WHAT ARE THE DIFFERENCES BETWEEN STRUCTURES AND ARRAYS?

Ans: Structure is a collection of heterogeneous data type but array is a collection ofhomogeneous data types.

1 ARRAY

1-It is a collection of data items of same data
type.2-It has declaration only

3-.There is no keyword.

4- array name represent the address of the starting element.

2 STRUCTURE

1-It is a collection of data items of different data
type.2- It has declaration and definition

3- keyword struct is used
4-Structure name is known as tag it is the short hand notation of the declaration.

8. IN HEADER FILES WHETHER FUNCTIONS ARE DECLARED OR DEFINED?

Ans: Functions are declared within header file. That is function prototypes exist in a headerfile,not function bodies. They are defined in library (lib).

9. WHAT ARE THE DIFFERENCES BETWEEN MALLOC () AND CALLOC ()?

Ans: Malloc Calloc 1-Malloc takes one argument Malloc(a);where a number of bytes 2-memory allocated contains garbage values

1-Calloc takes two arguments Calloc(b,c) where b no of object and c size of object
2-It initializes the contains of block of memory to zerosMalloc takes one argument, memoryallocated contains garbage values.

It allocates contiguous memory locations. Calloc takes two arguments, memory allocatedcontains all zeros, and the memory allocated is not contiguous.

10. WHAT ARE MACROS? WHAT ARE ITS ADVANTAGES AND DISADVANTAGES?

Ans: Macros are abbreviations for lengthy and frequently used statements. When a macro is called the entire code is substituted by a single line though the macro definition is of severallines.

The advantage of macro is that it reduces the time taken for control transfer as in case of function. The disadvantage of it is here the entire code is substituted so the program becomeslengthy if a macro is called several times.

Read more: c interview questions for freshers | FreeJobAlert.comhttp://www.freejobalert.com/clanguage-

interview-questions-part1/6125/#ixzz1O1xitnOp

11. DIFFERENCE BETWEEN PASS BY REFERENCE AND PASS BY VALUE?

Ans: Pass by reference passes a pointer to the value. This allows the callee to modify the variable directly.Pass by value gives a copy of the value to the callee. This allows the callee to modify the value without modifying the variable. (In other words, the callee simply cannot modify the variable, since it lacks a reference to it.)

12. WHAT IS STATIC IDENTIFIER?

Ans: A file-scope variable that is declared static is visible only to functions within that file. A function-scope or block-scope variable that is declared as static is visible only within that scope.Furthermore, static variables only have a single instance. In the case of function- or block-scopevariables, this means that the variable is not —automatic and thus retains its value across function invocations.

13. WHERE IS THE AUTO VARIABLES STORED?

Ans: Auto variables can be stored anywhere, so long as recursion works. Practically, they‘re stored on

the stack. It is not necessary that always a stack exist. You could theoretically allocate functioninvocation records from the heap.

14. WHERE DOES GLOBAL, STATIC, AND LOCAL, REGISTER VARIABLES, FREE MEMORY AND C PROGRAMINSTRUCTIONS GET STORED?

Ans: Global: Wherever the linker puts them. Typically the —BSS segment on many platforms. Static: Again, wherever the linker puts them. Often, they‘re intermixed with the globals. The only difference between globals and statics is whether the linker will resolve the symbols across compilation units.Local: Typically on the stack, unless the variable gets register allocated and never spills.Register: Nowadays, these are equivalent to —Local variables. They live on the stack unless they get register-allocated.

15. DIFFERENCE BETWEEN ARRAYS AND LINKED LIST?

Ans: An array is a repeated pattern of variables in contiguous storage. A linked list is a set of structures scattered through memory, held together by pointers in each element that point to the next element. With an array, we can (on most architectures) move from one element to the next by adding a fixed constant to the integer value of the pointer. With a linked list, there is a —next pointer in each structure which says what element comes next.

16. WHAT ARE ENUMERATIONS?

Ans: They are a list of named integer-valued constants. Example:enum color { black , orange=4,

yellow, green, blue, violet };This declaration defines the symbols —black , —orange , —yellow , etc. to have the values —1, —4, —5, … etc. The difference between an enumeration and a macro is that the enum actually declares a type, and therefore can be type checked.

17. DESCRIBE ABOUT STORAGE ALLOCATION AND SCOPE OF GLOBAL, EXTERN, STATIC, LOCAL AND REGISTERVARIABLES?

Ans:

Globals have application-scope. They're available in any compilation unit that includes an appropriate declaration (usually brought from a header file). They're stored wherever the linkerputs them, usually a place called the —BSS segment.

Extern? This is essentially —global.

Static: Stored the same place as globals, typically, but only available to the compilation unit that contains them. If they are block-scope global, only available within that block and its subblocks.**Local:** Stored on the stack, typically. Only available in that block and its subblocks.

(Although pointers to locals can be passed to functions invoked from within a scope where thatlocal is valid.)

Register: See tirade above on —local vs. —register. The only difference is that

the C compiler will not let you take the address of something you've declared as —register.

18. What are register variables? What are the advantages of using register variables? Ans:
If a variable is declared with a register storage class,it is known as register variable.The register variable is stored in the cpu register instead of main memory.Frequently used variablesare declared as register variable as it's access time is faster.

19. WHAT IS THE USE OF TYPEDEF?

Ans: The typedef help in easier modification when the programs are ported to another machine.A descriptive new name given to the existing data type may be easier to understand the code.

20. Can we specify variable field width in a scanf() format string? If possible how? Ans:
All field widths are variable with scanf(). You can specify a maximum field width for agiven

field by placing an integer value between the _%' and the field type specifier. (e.g. %64s). Such a specifier will still accept a narrower field width.

The one exception is %#c (where # is an integer). This reads EXACTLY # characters, and it is the

only way to specify a fixed field width with scanf().

21. OUT OF FGETS() AND GETS() WHICH FUNCTION IS SAFE TO USE AND WHY?

Ans: fgets() is safer than gets(), because we can specify a maximum input length. Neither one is completely safe, because the compiler can‘t prove that programmer won‘t overflow the buffer hepass to fgets ().

22. DIFFERENCE BETWEEN STRDUP AND STRCPY?

Ans: Both copy a string. strcpy wants a buffer to copy into. strdup allocates a buffer usingmalloc().

Unlike strcpy(), strdup() is not specified by ANSI .

23. WHAT IS RECURSION?

Ans: A recursion function is one which calls itself either directly or indirectly it must halt at a definite point to avoid infinite recursion.

24. DIFFERENTIATE BETWEEN FOR LOOP AND A WHILE LOOP? WHAT ARE IT USES?

Ans: For executing a set of statements fixed number of times we use for loop while when thenumber of iterations to be performed is not known in advance we use while loop.

25. WHAT IS STORAGE CLASS? WHAT ARE THE DIFFERENT STORAGE CLASSES IN C?

Ans: Storage class is an attribute that changes the behavior of a variable. It controls the lifetime,scope and linkage. The storage classes in c are auto, register, and extern, static, typedef.

26. WHAT THE ADVANTAGES OF USING UNIONS?

Ans: When the C compiler is allocating memory for unions it will always reserve enough room for the

largest member.

27. WHAT IS THE DIFFERENCE BETWEEN STRINGS AND ARRAYS?

Ans: String is a sequence of characters ending with NULL .it can be treated as a one dimensional array

of characters terminated by a NULL character.

28. WHAT IS A FAR POINTER? WHERE WE USE IT?

Ans: In large data model (compact, large, huge) the address B0008000 is acceptable because inthese

model all pointers to data are 32bits long. If we use small data model(tiny, small, medium) the above address won‘t work since in these model each pointer is 16bits long. If we are working in a small data model and want to access the address B0008000 then we use far pointer. Far pointeris always treated as a 32bit pointer and contains a segment address and offset address

both of 16bits each. Thus the address is represented using segment : offset format B000h:8000h. For anygiven memory address there are many possible far address segment : offset pair. The segment register contains the address where the segment begins and offset register contains the offset of data/code from where segment begins.

29. WHAT IS A HUGE POINTER?

Ans: Huge pointer is 32bit long containing segment address and offset address. Huge pointersare

normalized pointers so for any given memory address there is only one possible huge addresssegment: offset pair. Huge pointer arithmetic is doe with calls to special subroutines so its arithmetic slower than any other pointers.

30. WHAT IS A NORMALIZED POINTER, HOW DO WE NORMALIZE A POINTER?

Ans: It is a 32bit pointer, which has as much of its value in the segment register as possible.

Since

a segment can start every 16bytes so the offset will have a value from 0 to F. for normalizationconvert the address into 20bit address then use the 16bit for segment address and 4bit for the offset address. Given a pointer 500D: 9407,we convert it to a 20bitabsolute address 549D7,Which then normalized to 549D:0007.

31. WHAT IS NEAR POINTER?

Ans: A near pointer is 16 bits long. It uses the current content of the CS (code segment) register(if

the pointer is pointing to code) or current contents of DS (data segment) register (if the pointer ispointing to data) for the segment part, the offset part is stored in a 16 bit near pointer. Using nearpointer limits the data/code to 64kb segment.

32. IN C, WHY IS THE VOID POINTER USEFUL? WHEN WOULD YOU USE IT?

Ans: The void pointer is useful because it is a generic pointer that any pointer can be cast intoand back again without loss of information.

33. WHAT IS A NULL POINTER? WHETHER IT IS SAME AS AN UNINITIALIZED POINTER?

Ans: Null pointer is a pointer which points to nothing but uninitialized pointer may point toanywhere.

34. ARE POINTERS INTEGER?

Ans: No, pointers are not integers. A pointer is an address. It is a positive number.

35. What does the error 'Null Pointer Assignment' means and what causes this error?
Ans: As null pointer points to nothing so accessing a uninitialized pointer or invalid location may cause an error.

36. WHAT IS GENERIC POINTER IN C?

Ans: In C void* acts as a generic pointer. When other pointer types are assigned to genericpointer,

conversions are applied automatically (implicit conversion).

37. Are the expressions arr and &arr same for an array of integers?
Ans: Yes for array of integers they are same.

38. IMP>HOW POINTER VARIABLES ARE INITIALIZED?

Ans: Pointer variables are initialized by one of the following ways.

I. Static memory allocation
II. Dynamic memory allocation

39. WHAT IS STATIC MEMORY ALLOCATION?

Ans: Compiler allocates memory space for a declared variable. By using the address of operator,the

reserved address is obtained and this address is assigned to a pointer variable. This way of assigning pointer value to a pointer variable at compilation time is known as static memoryallocation.

40. WHAT IS DYNAMIC MEMORY ALLOCATION?

Ans: A dynamic memory allocation uses functions such as malloc() or calloc() to get memory dynamically. If these functions are used to get memory dynamically and the values returned bythese function are assigned to pointer variables, such a way of allocating memory at run time isknown as dynamic memory allocation.

41. WHAT IS THE PURPOSE OF REALLOC?

Ans: It increases or decreases the size of dynamically allocated array. The function realloc (ptr,n) uses two arguments. The first argument ptr is a pointer to a block of memory for which the size is to be altered. The second argument specifies the new size. The size may be increasedor decreased. If sufficient space is not available to the old region the function may create a newregion.

42. WHAT IS POINTER TO A POINTER?

Ans: If a pointer variable points another pointer value. Such a situation is known as a pointer to apointer.

Example:

```
int *p1,**p2,v=10;
P1=&v; p2=&p1;
```

Here p2 is a pointer to a pointer.

43. WHAT IS AN ARRAY OF POINTERS?

Ans: if the elements of an array are addresses, such an array is called an array of pointers.

44. DIFFERENCE BETWEEN LINKER AND LINKAGE?

Ans: Linker converts an object code into an executable code by linking together the necessary

built in

functions. The form and place of declaration where the variable is declared in a programdetermine the linkage of variable.

45. IS IT POSSIBLE TO HAVE NEGATIVE INDEX IN AN ARRAY?

Ans: Yes it is possible to index with negative value provided there are data stored in this location. Even if it is illegal to refer to the elements that are out of array bounds, the compilerwill not produce error because C has no check on the bounds of an array.

46. WHY IS IT NECESSARY TO GIVE THE SIZE OF AN ARRAY IN AN ARRAY DECLARATION?

Ans: When an array is declared, the compiler allocates a base address and reserves enough spacein

memory for all the elements of the array. The size is required to allocate the required space andhence size must be mentioned.

47. WHAT MODULAR PROGRAMMING?

Ans: If a program is large, it is subdivided into a number of smaller programs that are called modules or subprograms. If a complex problem is solved using more modules, this approach isknown as modular programming.

48. WHAT IS A FUNCTION?

Ans: A large program is subdivided into a number of smaller programs or subprograms. Each subprogram

specifies one or more actions to be performed for the larger program. Such sub programs arecalled functions.

49. WHAT IS AN ARGUMENT?

Ans: An argument is an entity used to pass data from the calling to a called function.

50. WHAT ARE BUILT IN FUNCTIONS?

Ans: The functions that are predefined and supplied along with the compiler are known asbuiltin

functions. They are also known as library functions.

51. DIFFERENCE BETWEEN FORMAL ARGUMENT AND ACTUAL ARGUMENT?

Ans: Formal arguments are the arguments available in the function definition. They are precededby

their own data type. Actual arguments are available in the function call. These arguments aregiven

as constants or variables or expressions to pass the values to the function.

52. IS IT POSSIBLE TO HAVE MORE THAN ONE MAIN() FUNCTION IN A C PROGRAM ?

Ans: The function main() can appear only once. The program execution starts from main.

53. What is the difference between an enumeration and a set of pre-processor # defines?
Ans: There is hardly any difference between the two, except that #defines has a global effect (throughout the file) whereas an enumeration can have an effect local to the block if desired. Some advantages of enumeration are that the numeric values are automatically assigned whereasin #define we have to explicitly define them. A disadvantage is that we have no control over the size of enumeration variables.

54. HOW ARE STRUCTURE PASSING AND RETURNING IMPLEMENTED BY THE COMPLIER?

Ans: When structures are passed as argument to functions, the entire structure is typicallypushed on

the stack. To avoid this overhead many programmer often prefer to pass pointers to structure instead of actual structures. Structures are often returned from functions in a location pointed to by an extra, compiler-supported _hidden' argument to the function.

55. IMP>what is the similarity between a Structure, Union and enumeration?
Ans: All of them let the programmer to define new data type.

56. CAN A STRUCTURE CONTAIN A POINTER TO ITSELF?

Ans: Yes such structures are called self-referential structures.

57. HOW CAN WE READ/WRITE STRUCTURES FROM/TO DATA FILES?

Ans: To write out a structure we can use fwrite() as Fwrite(&e, sizeof(e),1,fp);Where e is a structure

variable. A corresponding fread() invocation can read the structure back from file. calling fwrite() it writes out sizeof(e) bytes from the address &e. Data files written as memory images with fwrite(),however ,will not be portable, particularly if they contain floating point fields or Pointers. This is because memory layout of structures is machine and compiler

dependent. Therefore, structures written as memory images cannot necessarily be read back byprograms running on other machine, and this is the important concern if the data files you‘re writing will ever be interchanged between machines.

58. Write a program which employs Recursion? Ans:
int fact(int n) { return n > 1 ? n * fact(n – 1) : 1; }

59. WRITE A PROGRAM WHICH USES COMMAND LINE ARGUMENTS?ANS:

```
#include

void main(int argc,char *argv[])

{

int i;
clrscr();
for(i=0;i

printf(—\n%d  ,argv[i]);

}
```

60. DIFFERENCE BETWEEN ARRAY AND POINTER?ANS:

Array

1- Array allocates space automatically
2- It cannot be resized

3- It cannot be reassigned
4- sizeof (arrayname) gives the number of bytes occupied by the array.

3 POINTER

1-Explicitly assigned to point to an allocated space.
2-It can be sized using realloc()

3-pointer can be reassigned.
4-sizeof (p) returns the number of bytes used to store the pointer variable p.

61. WHAT DO THE 'C' AND 'V' IN ARGC AND ARGV STAND FOR?

Ans: The c in argc(argument count) stands for the number of command line argument theprogram is

invoked with and v in argv(argument vector) is a pointer to an array of character string thatcontain the arguments.

62. IMP>WHAT ARE C TOKENS?

Ans: There are six classes of tokens: identifier, keywords, constants, string literals, operators and other separators.

63. WHAT ARE C IDENTIFIERS?

Ans: These are names given to various programming element such as variables, function, arrays.It is a combination of letter, digit and underscore.It should begin with letter. Backspace isnot allowed.

64. DIFFERENCE BETWEEN SYNTAX VS LOGICAL ERROR?ANS:

Syntax Error

1-These involves validation of syntax of language.
2-compiler prints diagnostic message.

4 LOGICAL ERROR

1-logical error are caused by an incorrect algorithm or by a statement mistyped in such a waythat it doesn't violet syntax of language.
2-difficult to find.

65. WHAT IS PREINCREMENT AND POST INCREMENT?

Ans: ++n (pre increment) increments n before its value is used in an assignment operation or anyexpression containing it. n++ (post increment) does increment after the value of n is used.

66. WRITE A PROGRAM TO INTERCHANGE 2 VARIABLES WITHOUT USING THE THIRD ONE.ANS:

a ^= b; ie a=a^b
b ^= a; ie b=b^a;
a ^= b ie a=a^b;

here the numbers are converted into binary and then xor operation is performed.

You know, you're just asking —have you seen this overly clever trick that's not worth applying on

modern architectures and only really applies to integer variables?

67. WHAT IS THE MAXIMUM COMBINED LENGTH OF COMMAND LINE ARGUMENTS INCLUDING THE SPACEBETWEEN ADJACENT ARGUMENTS?

Ans: It depends on the operating system.

68. WHAT ARE BIT FIELDS? WHAT IS THE USE OF BIT FIELDS IN A STRUCTURE DECLARATION?

Ans: A bit field is a set of adjacent bits within a single implementation based storage unit that wewill call a —word .

The syntax of field definition and access is based on
structure.Struct {

unsigned int k :1;
unsigned int l :1;
unsigned int m :1;

}flags;

the number following the colon represents the field width in bits.Flag is a variable that containsthree bit fields.

69. WHAT IS A PREPROCESSOR, WHAT ARE THE ADVANTAGES OF PREPROCESSOR?

Ans: A preprocessor processes the source code program before it passes through the compiler.1- a preprocessor involves the readability of program

2- It facilitates easier modification
3- It helps in writing portable
programs
4- It enables easier debugging
5- It enables testing a part of program
6- It helps in developing generalized program

70. WHAT ARE THE FACILITIES PROVIDED BY PREPROCESSOR?ANS:

1-file inclusion
2-substitution facility
3-conditional compilation

71. WHAT ARE THE TWO FORMS OF #INCLUDE DIRECTIVE?ANS:

1.#include filename
2.#include

the first form is used to search the directory that contains the source file.If the search fails in thehome directory it searches the implementation defined locations.In the second form ,the preprocessor searches the file only in the implementation defined locations.

72. HOW WOULD YOU USE THE FUNCTIONS RANDOMIZE() AND RANDOM()?ANS:

Randomize() initiates random number generation with a random value.Random() generates random number between 0 and n-1;

73. WHAT DO THE FUNCTIONS ATOI(), ITOA() AND GCVT() DO?ANS:

atoi() is a macro that converts integer to character.itoa() It converts an integer to string

gcvt() It converts a floating point number to string

74. HOW WOULD YOU USE THE FUNCTIONS FSEEK(), FREED(), FWRITE() AND FTELL()?ANS:

fseek(f,1,i) Move the pointer for file f a distance 1 byte from location i.
fread(s,i1,i2,f) Enter i2 dataitems,each of size i1 bytes,from file f to string s.
fwrite(s,i1,i2,f) send i2 data items,each of size i1 bytes from string s to file f.
ftell(f) Return the current pointer position within file f.

The data type returned for functions fread,fseek and fwrite is int and ftell is long int.

75. WHAT IS THE DIFFERENCE BETWEEN THE FUNCTIONS MEMMOVE() AND MEMCPY()?

Ans: The arguments of memmove() can overlap in memory. The arguments of memcpy()cannot.

76. WHAT IS A FILE?

Ans: A file is a region of storage in hard disks or in auxiliary storage devices.It contains bytes of information .It is not a data type.

77. IMP>what are the types of file?
Ans: Files are of two types

1-high level files (stream oriented files) :These files are accessed using library functions
2-low level files(system oriented files) :These files are accessed using system calls

78. IMP>WHAT IS A STREAM?

Ans: A stream is a source of data or destination of data that may be associated with a disk orother

I/O device. The source stream provides data to a program and it is known as input stream. Thedestination stream eceives the output from the program and is known as output stream.

79. WHAT IS MEANT BY FILE OPENING?

Ans: The action of connecting a program to a file is called opening of a file. This requires creating

an I/O stream before reading or writing the data.

80. WHAT IS FILE?

Ans: FILE is a predefined data type. It is defined in stdio.h file.

81. WHAT IS A FILE POINTER?

Ans: The pointer to a FILE data type is called as a stream pointer or a file pointer. A file pointer points to the block of information of the stream that had just been opened.

82. HOW IS FOPEN()USED ?

Ans: The function fopen() returns a file pointer. Hence a file pointer is declared and it isassigned

as

FILE *fp;

fp= fopen(filename,mode);

filename is a string representing the name of the file and the mode represents:

—r for read operation

—w for write operation

—a for append operation

—r+ , w+ , a+ for update operation

5 83HOW IS A FILE CLOSED ?

Ans: A file is closed using fclose() function
Eg. fclose(fp);

Where fp is a file pointer.

84. WHAT IS A RANDOM ACCESS FILE?ANS:

A file can be accessed at random using fseek() function
fseek(fp,position,origin);

fp file pointer

position number of bytes offset from origin

origin 0,1 or 2 denote the beginning ,current position or end of file respectively.

85. WHAT IS THE PURPOSE OF FTELL ?

Ans: The function ftell() is used to get the current file represented by the file pointer.ftell(fp);

returns a long integer value representing the current file position of the file pointed by thefile pointer fp.If an error occurs ,-1 is returned.

86. WHAT IS THE PURPOSE OF REWIND() ?

Ans: The function rewind is used to bring the file pointer to the beginning of the file.Rewind(fp);

Where fp is a file pointer.Also we can get the same effect byfeek(fp,0,0);

87. DIFFERENCE BETWEEN A ARRAY NAME AND A POINTER VARIABLE?

Ans: A pointer variable is a variable where as an array name is a fixed address and is not avariable. A

pointer variable must be initialized but an array name cannot be initialized. An array name beinga constant value , ++ and — operators cannot be applied to it.

88. REPRESENT A TWO-DIMENSIONAL ARRAY USING POINTER?

Ans:

Address of a[I][j] Value of a[I][j]
&a[I][j]

or

a[I] + j
or

*(a+I) + j

*&a[I][j] or a[I][j]
or

*(a[I] + j)
or

*(* (a+I) +j)

89. DIFFERENCE BETWEEN AN ARRAY OF POINTERS AND A POINTER TO AN ARRAY?ANS:

Array of pointers

1- Declaration is: data_type
*array_name[size];2-Size represents the row
size.

3- The space for columns may be dynamically

6 POINTERS TO AN ARRAY

1-Declaration is data_type (
*array_name)[size];2-Size represents the
column size.

90. Can we use any name in place of argv and argc as command line arguments ?
Ans: yes we can use any user defined name in place of argc and argv;

91. WHAT ARE THE POINTER DECLARATIONS USED IN C?ANS:

1- Array of pointers, e.g , int *a[10]; Array of pointers to integer
2-Pointers to an array,e.g , int (*a)[10]; Pointer to an array of
into

3-Function returning a pointer,e.g, float *f( ) ; Function returning a pointer to
float4-Pointer to a pointer ,e.g, int **x; Pointer to apointer to int

5-pointer to a data type ,e.g, char *p; pointer to char

92. DIFFERENTIATE BETWEEN A CONSTANT POINTER AND POINTER TO A CONSTANT?ANS:

const char *p; //pointer to a const character.
char const *p; //pointer to a const character.
char * const p; //const pointer to a char
variable.

const char * const p; // const pointer to a const character.

93. IS THE ALLOCATED SPACE WITHIN A FUNCTION AUTOMATICALLY DEALLOCATED WHEN THE FUNCTIONRETURNS?

Ans: No pointer is different from what it points to .Local variables including local pointers variables in a function are deallocated automatically when function returns.,But in case of a local pointer variable ,deallocation means that the pointer is deallocated and not the block ofmemory allocated to it. Memory dynamically allocated always persists until the allocation isfreed

or the program terminates.

94. DISCUSS ON POINTER ARITHMETIC?ANS:

1- Assignment of pointers to the same type of pointers.2- Adding or subtracting a pointer and an integer.

3-subtracting or comparing two pointer.
4-incrementing or decrementing the pointers pointing to the elements of an array. When a pointerto an integer is incremented by one , the address is incremented by two. It is done automatically by the compiler.
5-Assigning the value 0 to the pointer variable and comparing 0 with the pointer. The pointerhaving address 0 points to nowhere at all.

95. WHAT IS THE INVALID POINTER ARITHMETIC?ANS:

i) adding ,multiplying and dividing two pointers.
ii) Shifting or masking pointer.
iii) Addition of float or double to pointer.
iv) Assignment of a pointer of one type to a pointer of another type ?

96. WHAT ARE THE ADVANTAGES OF USING ARRAY OF POINTERS TO STRING INSTEAD OF AN ARRAY OFSTRINGS?

Ans:

i) Efficient use of memory.
ii) Easier to exchange the strings by moving their pointers while sorting.

97. ARE THE EXPRESSIONS *PTR ++ AND ++ *PTR SAME?

Ans: No,*ptr ++ increments pointer and not the value pointed by it. Whereas ++ *ptrincrements the value being pointed to by ptr.

98. WHAT WOULD BE THE EQUIVALENT POINTER EXPRESSION FOE REFERRING THE SAME ELEMENT ASA[P][Q][R][S] ?

Ans : *(* (* (* (a+p) + q) + r) + s)

99. Are the variables argc and argv are always local to main?
Ans: Yes they are local to main.

100. CAN MAIN () BE CALLED RECURSIVELY?

Ans: Yes any function including main () can be called recursively.

101. IMP>CAN WE INITIALIZE UNIONS?

Ans: ANSI Standard C allows an initializer for the first member of a union. There is no standard way

of initializing any other member (nor, under a pre-ANSI compiler, is there generally any way of initializing a union at all).

102. What's the difference between these two declarations?
Ans: struct x1 { … };

typedef struct { … } x2;

The first form declares a structure tag; the second declares a typedef. The main difference is thatthe second declaration is of a slightly more abstract type.its users don't necessarily know that it is a structure, and the keyword struct is not used when declaring instances of it.

103. WHY DOESN'T THIS CODE: A[I] = I++; WORK?

Ans: The sub expression i++ causes a side effect. it modifies i's value. which leads to undefined behavior since i is also referenced elsewhere in the same expression.

104. WHY DOESN'T STRUCT X { … };X THESTRUCT;

work?

7 ANS:

C is not C++. Typedef names are not automatically generated for structure tags.

105. WHY CAN'T WE COMPARE STRUCTURES?ANS:

There is no single, good way for a compiler to implement structure comparison which is consistent with C's low-level flavor. A simple byte-by-byte comparison could founder on random bits present in unused —holes in the structure (such padding is used to keep the alignment of later fields correct). A field-by-field comparison might require unacceptableamounts of repetitive code for large structures.

106. HOW ARE STRUCTURE PASSING AND RETURNING IMPLEMENTED?

Ans: When structures are passed as arguments to functions, the entire structure is typicallypushed on

the stack, using as many words as are required. Some compilers merely pass a pointer to the structure, though they may have to make a local copy to preserve pass-by-value semantics. Structures are often returned from functions in a location pointed to by an extra,compilersupplied

—hidden argument to the function. Some older compilers used a special,static location

for structure returns, although this made structure-valued functions non-reentrant, which ANSI Cdisallows.

BELOW ARE OTHER SET OF QUESTIONS

Q1. What is C programming?

Ans. C is one of the oldest programming languages and was developed in 1972 by Dennis Ritchie. It is a high-level, general-purpose, and structured programming language that is widely used in various tasks, such as developing system applications, desktop applications, operating systems as well as IoT applications. It is the simple and flexible language used for a variety of scripting system applications, which form a significant part of Windows, UNIX, and Linux operating systems.

Q2. Why is C dubbed as a mother language?

Ans. It is known as a mother language because most of the JVMs, compilers, and Kernels are written in C language. If you know C, then you can easily grasp other programming languages.

Q3. Who is the founder of the C language? When was the C language developed?

Ans. C is developed by Dennis M. Ritchie in 1972 at the Bell laboratories of AT & T.

Q4. What are the key features of the C Programming language?

Ans. The following are the major features of C:

- Machine Independent
- Mid-Level programming language
- Memory Management
- Structured programming language
- Simple and efficient
- Function-richch libraries
- Case Sensitive

Q5. Name some different storage class specifiers in C.

Ans. Storage classes represent the storage of any variable. There are four storage classes in C:

- Auto
- Register
- Extern
- Static

Q6. What are the data types supported in the C Programming language?

Ans. This is one of the commonly asked C programming interview questions. The data type specifies the type of data used in programming. C language has some predefined data types with different storage capacities:

- **Built-in data types:** These includes int, char, double, float and void
- **Derived data types:** It includes array, pointers, and references
- **User-defined data types:** Union, structure, and Enumeration

Q7. What do you mean by the scope and lifetime of a variable in C?

Ans. The scope and lifetime of any variable are defined as the section of the code in which the variable is executable or visible. Variables that are described in the block are only accessible in the block. The lifetime of the variable defines the existence of the variable before it is destroyed.

Q8. What do you mean by the pointer in C?

Ans. Pointers are the variables in C that store the address of another variable. It allocates the memory for the variable at the run time. The pointer might be of different data types such as int, char, float, double, etc.

Example:

```
#include <stdio.h>

int main()

{

  int *ptr, q;

  q = 50;
```

```
  /* address of q is assigned to ptr */
  ptr = &q;

  /* display q's value using ptr variable */
  printf("%d", *ptr);
  return 0;
}
```

Q9. Define Null pointer?

Ans. A null pointer represents the empty location in the computer's memory. It is used in C for various purposes:

- It will assign the pointer variable to the variable with no assigned memory
- In pointer related code we can do error handling by checking null pointer before assigning any pointer variable
- Pass a null pointer to the variable if we don't want to pass any valid memory address

Example:

```
int fun(int *ptr)
{
 /*Fun specific stuff is done with ptr here*/
 return 10;
}

fun(NULL);
```

Q10. What is Dangling Pointer?

Ans. Dangling pointers are the pointers pointing to the memory location that has been deleted or released. There are three different types of dangling pointers:

Return local variable in a function call

```
#include<stdio.h>

#include<string.h>


char *getHello()

 {

char str[10]; strcpy(str,"Hello!");

 return(str);

 }


 int main()

{

//str falls out of scope

//function call char *getHello() is now a dangling pointer

printf("%s", getHello());

 }
```

Variable goes out of scope

```
void main()

{

  int *p1;

  …..

  {

     int ch;

     p1 = &ch;

  }
```

```
  .....
  // Here ptr is a dangling pointer
}
```

De allocation or free variable memory

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

int main()
{
char **strPtr;
char *str = “Hello!”;
 strPtr = &str; free(str);
 //strPtr is now a dangling pointer
 printf(“%s”, *strPtr);
 }
```

Q11. What is the use of function in C?

Ans. Functions are the basic building blocks of any programming language. All C programs are written using the function to maintain reusability and understandability.

Uses of functions in C:

- Functions can be used multiple times in a program by calling them whenever required
- In C, functions are used to avoid rewriting of code
- It is easy to track any C program when it is divided into functions

Syntax of a function

```
return_type function_name( parameter list )
{
//body of the function
```

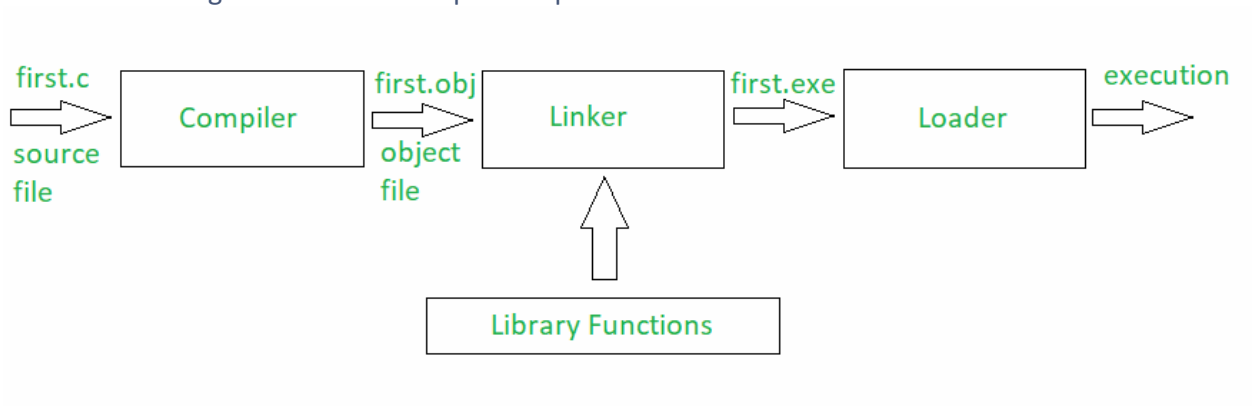
```
}
```

Let's take a look at some more **C programming interview questions**.

Q12. What happens when you compile a program in C?

Ans. At the time of compilation, the compiler generates a file with the same name as the C program file with different extensions.

Below is the image to show the compilation process:



Q13. What are header files in C?

Ans. Header files are those which contain C function declaration and macro definitions that are to be shared between sourced files. Header files are generated with the extension **.h**.

There are two types of header files:

- Programmer generated a header file
- Files that come with your compiler

Syntax: #include <file>

Q14. How many types of operators are there in C Programming?

Ans. An operator is a symbol used to operate the values of variables. There is a wide range of operators used in C programming, including –

Arithmetic Operators:

These are used to perform mathematical calculations such as addition, subtraction, multiplication, division, and modulus.

Example:

```
#include <stdio.h>


int main()
{
  int a = 9,b = 4, c;
  c = a+b;
  printf("a+b = %d \n",c);
  c = a-b;
  printf("a-b = %d \n",c);
  c = a*b;
  printf("a*b = %d \n",c);
  c = a/b;
  printf("a/b = %d \n",c);
  c = a%b;
  printf("Remainder when a divided by b = %d \n",c);
  return 0;
}
```

Relational Operators:

These are used to check the relation between two operands. If the relation is TRUE, it returns 1; If the relation is FALSE, It returns 0.

Example:

```
#include <stdio.h>


int main()
```

```
{
  int a = 2, b = 2, c = 6;
  printf("%d == %d is %d \n", a, b, a == b);
  printf("%d == %d is %d \n", a, c, a == c);
  printf("%d > %d is %d \n", a, b, a > b);
  printf("%d > %d is %d \n", a, c, a > c);
  printf("%d < %d is %d \n", a, b, a < b);
  printf("%d < %d is %d \n", a, c, a < c);
  printf("%d != %d is %d \n", a, b, a != b);
  printf("%d != %d is %d \n", a, c, a != c);
  printf("%d >= %d is %d \n", a, b, a >= b);
  printf("%d >= %d is %d \n", a, c, a >= c);
  printf("%d <= %d is %d \n", a, b, a <= b);
  printf("%d <= %d is %d \n", a, c, a <= c);
  return 0;
}
```

Logical Operators:

These are used in decision-making operations. If the expression is TRUE, it returns 1; if the expression is FALSE, it returns 0.

Example:

```
#include <stdio.h>

int main()
{
  int a = 2, b = 2, c = 6, result;
  result = (a == b) && (c > b);
  printf("(a == b) && (c > b) is %d \n", result);
```

```
  result = (a == b) && (c < b);
  printf("(a == b) && (c < b) is %d \n", result);
  result = (a == b) || (c < b);
  printf("(a == b) || (c < b) is %d \n", result);
  result = (a != b) || (c < b);
  printf("(a != b) || (c < b) is %d \n", result);
  result = !(a != b);
  printf("!(a == b) is %d \n", result);
  result = !(a == b);
  printf("!(a == b) is %d \n", result);
  return 0;
}
```

Assignment Operators:

These are used to assign value to a variable. The most used assignment operator is '='.

Example:

```
#include <stdio.h>
int main()
{
  int a = 5, c;
  c = a;     // c is 5
  printf("c = %d\n", c);
  c += a;    // c is 10
  printf("c = %d\n", c);
  c -= a;    // c is 5
  printf("c = %d\n", c);
  c *= a;    // c is 25
  printf("c = %d\n", c);
```

```
  c /= a;    // c is 5
  printf("c = %d\n", c);
  c %= a;    // c = 0
  printf("c = %d\n", c);
  return 0;
}
```

Increment and Decrement Operators:

These are used to change the value of an operand (constant or variable) by 1.

Example:

```
#include <stdio.h>
int main()
{
  int a = 10, b = 100;
  float c = 10.5, d = 100.5;
  printf("++a = %d \n", ++a);
  printf("–b = %d \n", –b);
  printf("++c = %f \n", ++c);
  printf("–d = %f \n", –d);
  return 0;
}
```

Bitwise Operators:

These are used to perform bit-level operations between two variables.

Example: Bitwise OR, Bitwise AND

Conditional Operator:

These are used in conditional expressions.

Example: '?:' Conditional operator

Special Operators:

There are some special operators in C used for:

sizeof(): Returns the size of the memory location

&: Returns the address of a memory location

***:** Pointer of a variable

Q15. Explain the process of creating increment and decrement operators in C.

Ans. This is one of the most important **C programming interview questions.** If you want to perform an increment operation, then use 'i++,' which will increase the value by 1. If you want to perform a decrement operation, then use 'i–,' it will decrease the value by 1.

Example:

```
#include<stdio.h>


int main()
{
   int x = 10, y = 1;
   printf("Initial value of x = %d\n", x);
   printf("Initial value of y = %d\n\n", y);
   y = ++x;
   printf("After incrementing by 1: x = %d\n", x);
   printf("y = %d\n\n", y);
   y = –x;
   printf("After decrementing by 1: x = %d\n", x);
   printf("y = %d\n\n", y);
   return 0;
}
```

Q16. Mention the difference between local variables and global variables in C?

Ans. Global variables are declared outside the function. That variable can be used anywhere in the program, whereas local variables are declared inside the function, and their scope is only inside that function.

```
// Global variable
float x = 1;
void my_test() {
  // Local variable called y.
  // This variable can't be accessed in other functions
  float y = 77;
  println(x);
  println(y);
}
void setup() {
  // Local variable called y.
  // This variable can't be accessed in other functions.
  float y = 2;
  println(x);
  println(y);
  my_test();
  println(y);
}
```

Q17. What is static memory allocation in C?

Ans. Static memory allocation is defined as the allocation of a fixed amount of memory at the compile time, and the operating system uses the data structure called stacks to manage memory allocation.

Example:

```
void demo
{
  int x;
}

int main()
{
  int y;
  int c[10];
  return 1;
}
```

Q18. What is dynamic memory allocation in C?

Ans. Dynamic memory allocation is the process of memory allocation at the run time. There are a group of functions in C used to dynamic memory management i.e. calloc(), malloc(), realloc() and free().

Q19. Explain the difference between calloc() and malloc().

Ans. The main difference between calloc() and malloc() is that calloc() takes two arguments while malloc() takes one argument. Secondly, calloc() initializes allocated memory to ZERO while malloc() does not initialize allocated memory.

Syntax of calloc()

```
void *calloc(size_t n, size_t size);
```

Syntax of malloc()

```
void *malloc(size_t n);
```

Q20. What is the output of the following C code?

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

enum {false, true};

int main()
{
  int i = 1;
  do
  {
    printf("%d\n", i);
    i++;
    if (i < 15)
      continue;
  }
while (false);
  getchar();
  return 0;
}
```

Ans. Output: 1

The do-while loop executes at every iteration. After the continue statement, it will come to the while (false) statement, and the condition shows false, and 'i' is printed only once.

Q21. How can a negative integer be stored in C?

Ans. If the number is with a negative sign, then at the time of memory allocation, the number (ignoring the minus sign) is converted into the binary equivalent. Then the two's complement of the number is calculated.

Example:

```
#include <stdio.h>
 int main()
 {
 int a = -4;
 int b = -3;
 unsigned int c = -4;
unsigned int d = -3;
 printf("%f\n%f\n%f\n%f\n", 1.0 * a/b, 1.0 * c/d, 1.0*a/d, 1.*c/b);
 }
```

Output:

1.333333

1.000000

-0.000000

-1431655764.000000

Q22. What is the use of nested structure in C?

Ans. A nested structure is used to make the complicated code easy. If we want to add the address of employees with other more details, then we have to create a nested structure for it.

Example:

```
#include<stdio.h>


struct address
{
```

```
    char city[20];
    int pin;
    char phone[14];
};

struct employee
{
    char name[20];
    struct address add;
};

void main ()
{
    struct employee emp;
    printf("Enter employee information?\n");
    scanf("%s %s %d %s",emp.name,emp.add.city, &emp.add.pin, emp.add.phone);
    printf("Printing employee information...\n");
printf("name:%s\nCity:%s\nPincode:%d\nPhone:
%s",emp.name,emp.add.city,emp.add.pin,emp.add.phone);
}
```

Output:

```
Enter employee information?
Joe
Delhi
110001
1234567890

Printing employee information...
```

Name: Joe

City: Delhi

Pincode: 110001

Phone: 123456789

Q23. How to write a program in C for swapping two numbers without the use of the third variable?

Ans.

```
#include <stdio.h>


int main()
{
  int x, y;
  printf("Input two integers (x & y) to swap\n");
  scanf("%d%d", &x, &y);
  x = x + y;
  y = x – y;
  x = x – y;
  printf("x = %d\ny = %d\n",x,y);
  return 0;
}
```

Q24. Write a C program for Fibonacci series.

Ans.

```
#include<stdio.h>


int main()
```

```
{
 int n1=0,n2=1,n3,i,number;
 printf("Enter the number of elements:");
 scanf("%d",&number);
 printf("\n%d %d",n1,n2);//printing 0 and 1
 for(i=2;i<number;++i)//loop starts from 2 because 0 and 1 are already printed
 {
  n3=n1+n2;
  printf(" %d",n3);
  n1=n2;
  n2=n3;
 }
  return 0;
 }
```

Q25. Explain the difference between = and == symbols in C programming?

Ans. The assignment operator (=): It is a binary operator used to operate two operands. It is used to assign the value to the variable.

Example: x=(a+b);

y=x;

Equal to operator (==): It is also a binary operator used to compare the left-hand side and right-hand side value, if it is the same, it returns 1 else 0.

```
int x,y;
x=10;
y=10;
if(x==y)
printf("True");
```

```
else

printf("False");
```

When the expression x==y evaluates, it will return 1

Let's move forward and take a look at some more frequently asked **C programming interview questions**.

Q26. What is the use of an extern storage specifier?

Ans. It enables you to declare a variable without bringing it into existence. The value is assigned to it in a different block, and it can be changed in the various blocks as well. So extern storage specifier is a global variable that can be used anywhere in the code.

Q27. What is the difference between rvalue and Ivalue in C?

Ans. The term rvalue refers to objects that appear on the right side, while an Ivalue is an expression that appears on the left side.

Q28. Can a program be compiled with the main function?

Ans. Yes.

Q29. Define stack.

Ans. It is a data structure that is used to store data in a particular order in which operations are performed. There are two types of storing orders, i.e. LIFO (last in first out) and FIFO (first in last out).

Basic operations performed in the stack:

- Push
- Pop
- Peek or Top
- isEmpty

Q30. When is the arrow operator used?

Ans. The Arrow operator is used to access elements in structure and union. It is used with a pointer variable. Arrow operator is formed by using a minus sign followed by a greater than a symbol.

Syntax:

(pointer_name)->(variable_name)

Q31. Can two operators be combined in a single line of program code?

Ans. Yes.

Q32. Write a program to find the factorial of a number using functions?

Ans: Program to find the factorial of a number

```
#include <stdio.h>


int factorial_of_a_number(int n)
{
int fact = 1, i;
if(n == 0)
return 1;
else
for(i = 1; i <= n; i++)
{
fact = fact * i;
}
return fact;
}
int main()
{
```

```
int n;
printf(“Enter the number : “);
scanf(“%d”,&n);

if(n < 0)
printf(“Invalid output”);

else
printf(“Factorial of the number %d is %d” ,n, factorial_of_a_number(n));
return 0;
}
```

Q33. What is a token in C?

Ans. The smallest individual unit in a C program is known as a token. Tokens can be classified as:

- Keywords
- Constants
- Identifiers
- Strings
- Operators
- Special symbols

Q34. Name the keyword used to perform unconditional branching.

Ans. A go-to statement is used to perform unconditional branching.

Q35. What is the use of the comma operator?

Ans. It is used to separate two or more expressions.

E.g. printf (“hello”);

Q36. Write a program to find a sum of first N natural numbers.

Ans.

```
#include <stdio.h>

void main()

{

  int i, num, sum = 0;

  printf("Enter an integer number \n");

  scanf ("%d", &num);

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

  {

     sum = sum + i;

  }

  printf ("Sum of first %d natural numbers = %d\n", num, sum);

}
```

Q37. What is the length of an identifier?

Ans. Its length is 32 characters in C.

Q38. What is typecasting in C?

Ans. It is a way to convert constant from one type to another type. If there is a value of float data type then you can typecast it into other data types.

There are two types of typecasting in C:

- Implicit conversion
- Explicit conversion

Example:

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

```
main() {
  int sum = 17, count = 5;
  double mean;
  mean = (double) sum / count;
  printf("Value of mean : %f\n", mean );
}
```

Q39. How are random numbers generated?

Ans. Random numbers are generated by using the rand () command.

Example:

```
#include <time.h>
#include <stdlib.h>
Srand (time (NULL));
Int r = rand ();
```

Q40. What are the disadvantages of void pointer?

Ans. Disadvantages of a void pointer:

- Pointer arithmetic is not defined for void pointer
- Void pointers can't be dereferenced

Q41. Define compound statements.

Ans. These are made up of two or more program statements that are executed together.

Q42. Write a program to print numbers from 1 to 100 without using a loop.

Ans. Program to print numbers from 1 to 100

```
/* Prints numbers from 1 to n */
void printNos(unsigned int n)
{
  if(n > 0)
  {
    printNos(n-1);
    printf("%d ",  n);
  }
}
```

Q43. What is FIFO?

Ans. FIFO means first in first out. It is a cost flow assumption, which is used to remove costs from the inventory account.

Q44. What is the use of a built-in stricmp() function?

Ans. It takes two strings and returns an integer.

Q45. What is the name of the function used to close the file stream?

Ans. Fclose().

Q46. What is the structure?

Ans. A user-defined data type that enables the combination of different data types to store a particular type of record, is known as a structure.

Example:

```
struct Point
{
  int x, y;
} p1;  // The variable p1 is declared with 'Point'


struct Point
{
  int x, y;
};
int main()
{
  struct Point p1;  // The variable p1 is declared like a normal variable
}
```

Q47. Write a program to reverse a number.

Ans. Below is the program to reverse a number in C:

```
#include <stdio.h>
int main()
{
int n, rev = 0, rem;
printf("\nEnter a number : ");
scanf("%d", &n);
printf("\nReversed Number : ");
while(n != 0)
{
rem = n%10;
```

```
rev = rev*10 + rem;
n /= 10;
}
printf("%d\n", rev);
return 0;
}
```

Q48. Write a program to check the prime number in C.

Ans. Below is the program to check the prime number in C:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,m=0,flag=0;    //declaration of variables.
clrscr();    //It clears the screen.
printf("Enter the number to check prime:");
scanf("%d",&n);
m=n/2;
for(i=2;i<=m;i++)
{
if(n%i==0)
{
printf("Number is not prime");
flag=1;
break;    //break keyword used to terminate from the loop.
}
}
```

```
if(flag==0)
printf("Number is prime");
getch();   //It reads a character from the keyword.
}
```

Q49. Write a program to check Armstrong number in C.

Ans. The program to check Armstrong number in C is as follows:

```
#include<stdio.h>
#include<conio.h>
main()
{
int n,r,sum=0,temp;   //declaration of variables.
clrscr(); //It clears the screen.
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=sum+(r*r*r);
n=n/10;
}
if(temp==sum)
printf("armstrong  number ");
else
printf("not armstrong number");
getch();  //It reads a character from the keyword.
```

```
}
```

Q50. Write a program to check the palindrome number in C.

Ans. The program to check palindrome number in C is as follows:

```
#include<stdio.h>


#include<conio.h>
main()
{
int n,r,sum=0,temp;
clrscr();
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=(sum*10)+r;
n=n/10;
}
if(temp==sum)
printf("palindrome number ");
else
printf("not palindrome");
getch();
}
```

Q51. What are reserved keywords? How many reserved keywords are there in C?

Ans. Reserved keywords are those keywords that have predefined meanings and cannot be used as a variable name. Such keywords are restricted for general use while writing a program. There are 32 reserved keywords in C programming language:

Reserved Keywords			
auto break case char const continue default do	double else enum extern float for goto if	int long register return short signed sizeof static	 struct switch typedef union unsigned voidvolatilewhile

Q52. What is a union?

Ans. A union is a user-defined data type that enables you to store multiple types of data in a single unit or the same memory location. While you can define a union with different members, only one member can hold a value at any given time. Also, it does not hold the sum of the memory of all members and holds the memory of the largest member only.

Q53. What happens if a header file is included twice?

Ans. If a header file is included twice, the compiler will process its contents twice, resulting in an error. You can use a guard(#) to prevent a header file from being included multiple times during the compilation process. Thus, even if a header file with proper syntax is included twice, the second one will get ignored.

Q54. Can a C program compile without the main() function?

Ans. Yes, a C program can be compiled without the main() function. However, it will not execute without the main() function.

Q55. What is the difference between static memory allocation and dynamic memory allocation?

Ans. The differences between static memory allocation and dynamic memory allocation are:

Static Memory Allocation	Dynamic Memory Allocation
It is done before the program execution.	It takes place during program execution.
Variables get allocated permanently.	Variables get allocated when the program unit gets active.
Less efficient.	More efficient.
It uses a stack for managing the static allocation of memory.	It uses heap for managing the dynamic allocation of memory.
No memory re-usability.	It allows memory re-usability.
Execution is faster.	Execution is slower.
Memory is allocated at compile time.	The allocation of memory is done at run time.
Memory size can not be modified while execution. Example: Array	Memory size can be modified while execution. Example: Linked List

Q56. What do you mean by a memory leak in C?

Ans. A memory leak is a kind of resource leak that happens when programmers create a memory in heap and forget to delete it. Thus, the memory which is no longer needed remains undeleted. It may also occur when an object is inaccessible by running code but it is still stored in memory. A memory leak can result in additional memory usage and can affect the performance of a program.

Q57. What do you understand by while(0) and while(1)?

Ans. In while(1) and while(0), 1 means TRUE or ON and 0 means FALSE or OFF.

while(0) means that the looping conditions will always be false and the code inside the while loop will not be executed. On the other hand, while(1) is an infinite loop. It runs continuously until it comes across a break statement mentioned explicitly.

Q58. Explain the difference between prefix increment and postfix increment.

Ans. Both prefix increment and postfix increment do what their syntax implies, i.e. increment the variable by 1.

The prefix increment increments the value of the variable before the program execution and returns the value of a variable after it has been incremented. The postfix increment, on the other hand, increments the value of the variable after the program execution and returns the value of a variable before it has been incremented.

- ++a <- Prefix increment
- a++ <- Postfix increment

Q59. What is the difference between a null pointer and a void pointer?

Ans. A null pointer is one that does not point to any valid location and its value isn't known at the time of declaration.

Syntax: <data type> *<variable name> = NULL;

On the other hand, void pointers are generic pointers that do not have any data type associated with them. They can contain the address of any type of variable. Void pointers can point to any data type.

Syntax: void *<data type>;

Q60. Explain the difference between Call by Value and Call by Reference?

Ans. Call by Value sends the value of a variable as a parameter to a function. Moreover, the value in the parameter is not affected by the operation that takes place.

Call by Reference, on the other hand, passes the address of the variable, instead of passing the values of variables. In this, values can be affected by the process within the function.

Q61. How can the scope of a global symbol be resolved in C?

Ans. The scope of a global symbol can be resolved by using the extern storage, which extends the visibility or scope of variables and functions. Since C functions are visible throughout the program by default, you don't need to use it with function declaration or definition.

Q62. What is the difference between actual parameters and formal parameters?

Ans. The differences between actual parameters and formal parameters are:

Actual Parameters	Formal Parameters
They are included at the time of function call.	They are included at the time of the definition of the function.
Actual Parameters do not require data type but the data type should match with the corresponding data type of formal parameters.	Data types need to be mentioned.
These can be variables, expressions, and constant without data types.	These are variables with their data type.

Q63. Explain modular programming.

Ans. Modular programming is an approach that focuses on dividing an entire program into independent and interchangeable modules or sub-programs, such as functions and modules for achieving the desired functionality. It separates the functionality in such a manner that each sub-program contains everything necessary to execute just one aspect of the desired functionality.

Q64. What is a sequential access file?

Ans. As the name suggests, a sequential access file is used to store files sequentially, i.e. one data is placed into one file after another. It means that if you have to access files you will have to check each file sequentially (reading one data at a time) until your desired file is reached. It is more limitations as access time will be very high and storage cost is high.

Q65. What is the difference between const char* p and char const* p?

Ans. const char* p is a pointer to a constant character whereas char const* p is a constant pointer to a non-constant character. In const char* p, we cannot change the value pointed by ptr. However, we can the pointer. In char *const ptr, we cannot change the pointer p. However, the value pointed by ptr can be changed.

Q66. Can we use the ‘if’ function to compare strings?

Ans. No, we cannot use the ‘if’ function to compare two strings. The ‘if’ function compares numerical and single character values. We can use the ‘strcmp’ function to compare two strings character by character.

Q67. What is the use of toupper() function in C?

Ans. The toupper() function in C converts the lowercase alphabet to uppercase. We define the toupper() function using the ctype.h header file.

Syntax:

int toupper(int ch);

Q68. Explain the newline escape sequence.

Ans. the newline escape sequence denoted by \n is used to insert a newline character into a string.

Q69. Explain the # pragma directive in C.

Ans. # pragma is a special purpose directive that is used to turn on or off certain features. Some of the #pragma directives are:

- #pragma startup: It is used to specify the functions that are needed to run before the program starts.
- #pragma exit: It is used to specify the functions that are needed to run just before program exit.
- #pragma warn: It is used to hide the warning messages which are displayed during compilation.
- #pragma GCC poison: This directive is used to remove an identifier completely from the program.
- #pragma GCC dependency: This directive allows you to check the relative dates of the current file and another file.

- #pragma once: It allows the current source file to be included only once in a single compilation.

Q70. What is variable initialization in C Programming?

Ans. Variable initialization refers to the process of assigning a value to the variable before it is used in the program. Using variables without initialization can result in unexpected outputs. There are two types of variable initialization:

- Static Initialization: variable is assigned a value in advance and it acts as a constant.
- Dynamic Initialization: variable is assigned a value at run time and can be altered every time the program is being run.

Syntax of the variable initialization:

data_type variable_name=constant/literal/expression;

or

variable_name=constant/literal/expression;

Q71. What is the difference between Source Code and Object Code?

Ans. The differences between source code and object code are:

Source Code	Object Code
It is created by a programmer.	Object code refers to the output, which is produced when the Source Code is compiled with a C compiler.
Source code is high-level code.	It is a low-level code.
Source code is written in plain text by using some high-level programming language like C, C++, Java, or Python.	It is written in machine language.
It is understandable by humans but not directly understandable by machines.	Object code is machine-understandable but not human-understandable.
Source code is the input to the compiler or	It is the output of the compiler or any other

any other translator.	translator.
It can be easily modified.	Object code can not be modified.

Q72. In C programming, how can you determine if a number is odd or even?

Ans. In C programming, we can determine if a number is even or odd by dividing it by 2. If the number is exactly divisible by 2 (the remainder is 0) then it is an even number else it is odd.

Program to check if a number is even or odd using Bitwise Operator:

```
#include <stdio.h>

int main() {
   int num;
   printf("Enter an integer: ");
   scanf("%d", &num);
   // true if num is perfectly divisible by 2
   if(num % 2 == 0)
      printf("%d is even.", num);
   else
      printf("%d is odd.", num);
   return 0;
}
```

Output:

Enter an integer: 77

77 is odd.

Q73. Write a program to swap two numbers without the use of the third variable.

Ans. We can swap two numbers without the use of the third variable by two methods:

Method 1: Using sum and subtraction

```
#include<stdio.h>

 int main()

{

int a=10, b=20;

printf(“Before swap a=%d b=%d”,a,b);

a=a+b;//a=30 (10+20)

b=a-b;//b=10 (30-20)

a=a-b;//a=20 (30-10)

printf(“\nAfter swap a=%d b=%d”,a,b);

return 0;

}
```

Method 2: Using multiplication and division

```
#include<stdio.h>

#include<stdlib.h>

 int main()

{

int a=10, b=20;

printf(“Before swap a=%d b=%d”,a,b);

a=a*b;//a=200 (10*20)

b=a/b;//b=10 (200/20)

a=a/b;//a=20 (200/10)

 system(“cls”);

printf(“\nAfter swap a=%d b=%d”,a,b);

return 0;

}
```

Output:

For both methods, the output will be:

Before swap a=10 b=20

After swap a=20 b=10

Q74. What is recursion in C? Write a program to find the factorial of a number using recursion.

Ans. Recursion refers to the process in which a function calls itself directly or indirectly. The function that calls itself is known as a recursive function.

Program to find the factorial of a number using recursion:

```
#include <stdio.h>
int fact (int);
int main()
{
   int n,f;
   printf("Enter a positive integer: ");
   scanf("%d",&n);
   f = fact(n);
   printf("Factorial = %d",f);
}
int fact(int n)
{
   if (n==0)
   {
      return 0;
   }
   else if ( n == 1)
```

```
    {
        return 1;
    }
    else
    {
        return n*fact(n-1);
    }
}
```

Output

Enter a positive integer: 5

Factorial = 120

Q75. What is the difference between macros and functions?

Ans: The differences between macros and functions are:

Macros	Functions
Macros are pre-processed. It means that all the macros will be processed before the program compiles.	Functions are not preprocessed but compiled.
Macros do not check for compilation errors which leads to unexpected output.	Function checks for compilation errors. Chances of unpredictable output are less.
It increases the code length.	Code length remains the same.
Macros are useful when small code is repeated many times.	Functions are useful in large codes.
Before Compilation, the macro name is replaced by macro value.	During function call, transfer of control takes place.

Faster in execution.	A bit slower in execution than Macros.
They do not check any Compile-Time errors.	Functions check Compile-Time errors.

Q76. Explain pointer to pointer in C.

Ans. Pointer to pointer is a concept in which one pointer refers to the address of another pointer. It is a form of multiple indirections or a chain of pointers. The first pointer stores the address of a variable whereas the second pointer stores the address of the first pointer.

Example:

```
#include <stdio.h>

int main () {

  int  var;

  int  *ptr;

  int  **pptr;

  var = 2000;


  /* take the address of var */

  ptr = &var;


  /* take the address of ptr using address of operator & */

  pptr = &ptr;


  /* take the value using pptr */

  printf("Value of var = %d\n", var );

  printf("Value available at *ptr = %d\n", *ptr );

  printf("Value available at **pptr = %d\n", **pptr);

  return 0;
```

```
}
```

Output:

Value of var = 2000

Value available at *ptr = 2000

Value available at **pptr = 2000

Q77. Write a program to print the following:

1

12

123

1234

12345

Ans: Program to print the above pattern is:

```
#include<stdio.h>
int main()
{
    for(i=1;i&lt;=5;1++)
    {
        for(j=1;j&lt;=5;j++)
        {
            print("%d",j);
        }
        printf("n");
    }
    return 0;
}
```

Q78. Explain Bubble Sort Algorithm with the help of a program.

And. Bubble Sort Algorithm is a simple sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. Swapping is repeated until the list is sorted.

Program to execute Bubble Sort:

```
#include <stdio.h>

int main()

{

   int a[100], number, i, j, temp;

   printf("\n Enter the total Number of Elements:  ");

   scanf("%d", &number);

   printf("\n Enter the Array Elements:  ");

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

      scanf("%d", &a[i]);

   for(i = 0; i < number – 1; i++)

   {

      for(j = 0; j < number – i – 1; j++)

      {

         if(a[j] > a[j + 1])

         {

            temp = a[j];

            a[j] = a[j + 1];

            a[j + 1] = temp;

         }

      }

   }

   printf("\n List Sorted in Ascending Order: ");

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

```
    {
        printf(" %d \t", a[i]);
    }
    printf("\n");
    return 0;
}
```

Output:

Enter the total Number of Elements: 6

Enter the Array Elements: 2 9 7 7 1 -3

List Sorted in Ascending Order: -3 1 2 7 7 9

Q79. Explain Insertion Sort Algorithm with the help of a program.

Ans: In insertion sort, the array is virtually split into a sorted and an unsorted part. Values from the unsorted part are taken out and placed at the correct position in the sorted part of the array.

Program to execute Insertion Sort in C:

```
#include <stdio.h>
// Function to print an array
void printArray(int array[], int size) {
for (int i = 0; i < size; i++) {
printf("%d ", array[i]);
}
printf("\n");
}
void insertionSort(int array[], int size) {
for (int step = 1; step < size; step++) {
```

```
int key = array[step];
int j = step – 1;

// Compare key with each element on the left of it until an element smaller than it is found.
while (key < array[j] && j >= 0) {
array[j + 1] = array[j];
–j;
}
array[j + 1] = key;
}
}
// Main function
int main() {
int data[] = {9, 5, 1, 4, 3};
int size = sizeof(data) / sizeof(data[0]);
insertionSort(data, size);
printf(“Sorted array in ascending order:\n”);
printArray(data, size);
}
```

Output:

Sorted array in ascending order:

1

3

4

5

9

Q80. What is /0 character?

Ans: /0 is the Symbol mentioned is called a **Null Character**. It's considered the terminating character. It's used in strings to alert the end of the string to the compiler.

Q 81. What is Preprocessor?

Ans: A preprocessor is a software program that processes a source file before sending it for compilation. Inclusion of header files, macro expansions, conditional compilation, and line control are all possible with the preprocessor.