C Programming Interview Questions

What do you mean by a sequential access file?

- While writing programs that store and retrieve data in a file, it is possible to designate that file into different forms.
- A sequential access file is such that data are saved in sequential order: one data is placed into the file after another.
- If you want to access a particular data within the sequential access file, data has to be read - one data at a time, until you reach the data you want.

Explain zero based addressing.

- The array subscripts always start at zero.
- These subscript values are used to identify the elements in the array.
- Since subscripts start at 0, arrays are said to use zero-based addressing.

Are bit fields portable?

- No, Bit fields are not portable.
- Since Bit fields cannot span machine words and the number of bits in a machine word is different on different machines, a particular program using bit fields might not compile on some machines.
- One should avoid using bit fields except when the machines can directly address bits in memory and the compiler can generate code.

Explain high-order and low-order bytes.

- The numbers are written from left to right in the decreasing order of significance. Similarly, the bits in a byte of computer memory can be considered as digits of a number written in base
- The byte holding the least significant 8 bits is called the low-order byte.
- The byte holding the most significant 8 bits is the most significant byte, or high- order byte.

What are the different types of endless loops?

- An endless loop can be of two types
    - A loop that is intentionally designed to go round and round until the condition within the loop is met. Once the condition is met, a break function gets the program out of the loop.
    - A loop condition written by mistake, causing the loop to run erroneously forever. Endless loops are also referred to as infinite loops.

Why are all header files not declared in every C program?

- Declaring all header files in every program would lead to increase in the overall file size and load of the program. It is not a good programming.
- The choice of header files that you want to declare in the program depends on the commands/functions you want to use in the program. Each header file contains different commands and functions. So we use only the files relevant to our program.

Which one would you prefer - a macro or a function?

- It actually depends on the purpose of the program

  - In case of macros, the corresponding code is inserted directly into your source code at the point where macro is called. There is no overhead involved in using a macro.This makes the Macros more efficient and faster than functions. However, macros are usually small and cannot handle large, complex coding constructs. So, if it is a complex situation that the program wants to handle, functions are more suitable.
  - Macros are expanded inline - this means that every time a macro occurs, the code is replicated. So, the code size in case of usage of macros will be larger in comparison to functions.

So, the choice of using macros or functions actually depends on your requirement - speed vs program size.
If your program contains small, repeated code sections, you should use Macros. If the program requires, large number of unique code lines - you should prefer functions.

What is break statement?

- A break statement causes the loop to terminate. The control then passes to the statement following the body of the loop.
- Explain spaghetti programming.
  - Spaghetti programming refers to codes that tend to get tangled and overlapped throughout the program.
  - It makes a program complex and analyzing the code becomes difficult. It usually happens due to the lack of work experience on developer's part.

Differentiate between the = symbol and == symbol?

- The = symbol is usually used in mathematical operations. It is used to assign a value to a given variable while the == symbol is a relational operator that is used to compare two values.

What are actual arguments?

- When some functions are created and used to perform an action on some given values, some values need to be passed to them. The values that are passed into the called function are referred to as actual arguments.

What is a newline escape sequence?

- A newline escape sequence is represented by the \n character.
- It is used to insert a new line while displaying the output data.
- To add more spaces you can use more \n characters.

How many levels deep can include files be nested?

- As such, there is no limit to the number of levels of nested include files you can have but your compiler might run out of stack space while trying to include an inordinately high number of files. This number depends on your hardware configuration and compiler.
- Although it is legal to nest include files yet you must avoid it. An include level should be created only where it is required and makes sense, like creating an include file that has an #include statement for each header required by the module you are working with.

What are pre-processor directives?

- Pre-processor directives are placed at the beginning of a C program. They begin with # symbol.
- This is the place, where library files are specified depending on the functions to be used in the program.

- Pre-processor directives are also used for declaration of constants.

What are compound statements?

- Compound statements are made up of two or more program statements which are executed together. They can be executed with a loop.
- Curly brackets { } are placed before and after compound statements.
- Compound statements are usually used while handling conditions in which a series of statements are executed when a TRUE or FALSE is evaluated.

How does placing some code lines between the comment symbol help in debugging the code?

- Placing comment symbols /* */ around a code isolates some code that the coder believes might be causing the errors in the program, without deleting it.
- If the code is correct, you can just remove the comment symbols, without needing to retype it.
- If it is wrong, you can just remove it.

Is it possible to pass an entire structure to functions?

- Yes, it is possible to pass an entire structure to a function in a call by method style. Some programmers prefer to declare the structure globally, then pass a variable of that structure type to a function. It helps in maintaining the consistency and uniformity in terms of argument type.

What are header files? What are their uses?

- Header files are also called as library files.
- They carry two important things: the definitions and prototypes of functions being used in a program.
- For example: stdio.h is a header file that contains definition and prototypes of commands like printf and scanf.

Is it possible to create your own header files?

- Yes, it is possible to create a customized header file.
- To do this, you just need to include the function prototypes that you want to use in your program in it, and use the #include directive followed by the name of your header file.

Why is a semicolon (;) put at the end of every program statement?

- It is done for parsing process and compilation of the code.
- A semicolon acts as a delimiter. This tells the compiler where each statement ends, and can proceed to divide the statement into smaller elements for checking the syntax.

How do you access the values within an array?

- Arrays contain a number of elements, depending on the size you assigned it during variable declaration.
- Each element is assigned a number from 0 to number of elements minus 1.
- To assign or retrieve the value of a particular element, refer to the element number. For example: if you have a declaration that says “intmarks[6];”, then you have 6 accessible elements, namely: marks[0], marks[1], marks[2], marks[3], marks[4] and marks[5].

What is the role of && operator in a program code?

- The && is also referred to as AND operator.
- When this operator is used, all conditions specified must be TRUE before the next action can be carried out.

- If any of the conditions is false, the whole statement is false.

Differentiate between functions getch() and getche().
- Both functions accept a character input value from the user.
- When getch() is used, the key that was pressed will not appear on the screen. It is automatically captured and assigned to a variable.
- While when getche() is used, the key that was pressed by the user appears on the screen and is assigned to a variable.

What are the various types of control structures in programming?
- Mainly there are 3 types of control structures in programming: Sequence, Selection and Repetition.
- Sequential control follows a top- down flow in executing a program. This means that step 1 is first executed, followed by step 2 and so on.
- Selection means dealing with conditional statements. This means that the code is executed depending on the evaluation of conditions a TRUE or FALSE. It means that not all codes may be executed and there are alternative flows within.
- Repetitions are also called as loop structures. They will repeat one or two program statements as set by a counter.

Differentiate between the expression “++a” and “a++”?
- With ++a, the increment happens first on variable a, and the resulting value is used. This is called as prefix increment.
- With a++, the current value of the variable will be used in an operation. This is called as postfix increment.

What are control structures?
- Control structures decide which instructions in the program should be executed.
- This implies that the program flow may not necessarily move from one statement to the next one. Some alternative portions may be executed depending on the conditional statements.

Explain enumerated types.
- Enumerated types allow the programmers to use more meaningful words as values to a variable.
- Each item in the enumerated type variable is actually associated with a numeric code. For an enumerated type variable named Months can be created. Its values can be January, February,....December.

Are comments included during the compilation stage and placed in the EXE file as well?
- No, comments encountered by the compiler are disregarded.
- Their only purpose is guidance and ease of programming. They have no effect on the functionality of the program.

Tell us something about keyword "auto".
- "Automatic" is a local variable with a local lifetime.
- It is declared by auto storage-class specifier.
- This variable is visible only in the block in which it is declared.
- The value of uninitialized auto variables is undefined.
- The variables with auto storage class need to be initialised while storing them or initial values need to be assigned to them in statements within the block.

Explain the meaning of keyword 'extern' in a function declaration.

- 'extern' modifier in a method declaration implies that the method is implemented externally.
- The program doesn't reserve any memory for a variable declared as 'extern'.
- A variable that is required to be used in every file of the project is declared globally in one file - and not inside any function.
- 'extern' declaration of that variable is added to a header file not included in all others.

Differentiate between #include<...> and #include "..."

- #include<...> means that the directories other than the current one will be searched for the header file.
- #include "..." means that the current directory will be searched for the header file before any other directories.

Situation - The 'sizeof' operator reported a larger size than the calculated size for a structure type. What could be the reason?

- The 'sizeof' operator shows the amount of storage needed to store an object of the specified type of operand.
- The result of applying it to a structure type name is the number of bytes including internal and trailing padding.
- This may include internal leading & trailing padding used for the alignment of structure on memory boundaries. This may cause the error.

What does *p++ do? What does it point to?

- *p++ increments p.
- It returns the value pointed to by p before incrementation.

Explain "Bus Error".

- It is a fatal error in the execution of a machine language instruction.
- It occurs because the processor detects an abnormal condition on its bus.

The causes may be:

- Invalid address alignment.
- Accessing a physical address that does not correspond to any device.
- Other device specific hardware error.

What are volatile variables?
Volatile variables are like other variables except that the values might be changed at any given point of time only by ‘some external resources’.
Ex:

```
volatile int number;
```

The value may be altered by some external factor, though if it does not appear to the left of the assignment statement. The compiler will keep track of these volatile variables.

What do you think about the following?

```
int i;
scanf("%d", i);
printf("%d", i);
```

- The program will compile without any errors.

- When you try to enter a value using "stdin", the system will try to store the value at location with address "i". "i" might be invalid leading the program to crash and core dump.
- It implies that this code has a bug.

What will be the output of the following?

```
int main()
main();
return 0;
```

- Runt time error. Stack overflow.

What are the advantages of using linked list for tree construction?

- Unless the memory is full, overflow will never occur.
- Insertions & deletions are easier in comparison to arrays.
- Moving pointer is faster & easier in comparison to moving items when records are large.

Which data structure is used to perform recursion?

- Stack data structure is used to perform recursion.
- Its LIFO property stores return addresses of the 'caller' and its successions in recursion to find

where the execution control should return.
Explain the following.
a.) Queue b.) Priority Queues -

- a.) Queue –
  - Queue is a FIFO or LIFO data structure.
  - It permits two operations - enqueue & dequeue.
  - The methods used to check the status of the queue are - empty() and full()

- b.) Priority Queues -
  - List of items with priority associated with it.
  - Priority queues effectively support finding the item with highest priority across a series of operations.
  - Basic priority queue operations are - insert, find maximum or minimum, delete maximum or minimum.

Explain the following.
a.) Binary height balanced tree b.) Ternary tree c.) Red-black trees

- a) Binary height balanced tree-
  - It is a binary tree in which for every node the heights of left and right subtree differ by not more than 1.
- b.) Ternary tree -
  - In this tree a node can have zero or more child nodes.

- c.) Red-black trees -
  - It is a binary search tree with following properties:
  - Root is black.
  - Every leaf is black.
  - Every node is either red or black.

- For a red node, both its children are black.
- All internal nodes have two children .

Every simple path from a node to a descendant leaf contains the same number of black nodes.

What is "Bus error"?

- A ‘bus error’ is certain undefined behavior result type. The cause for such error in a system could not be specified by the C language. The memory accessibility which CPU could not address physically, ‘bus error’ occurs. Also, any fault detected by a device by the computer system can also be a ‘bus error’. These errors caused by programs that generate undefined behavior which C language no longer specifies what can happen.

Throw some light on the following.
a.) Splay trees b.) B tree

- a.) Splay trees -
  - These are self adjusting binary search trees.
  - They can adjust optimally to any sequence of tree operations.
  - Everytime a node of the tree is accessed, a radical surgery is performed on it. This results into the newly accessed node turning into the root of the modified tree.
  - Splaying steps used are: Zig rotation, Zag rotation, Zig-zag, Zag-zig, Zig-zig, Zag-zag.
- b.) B tree -
  - The root is either a tree leaf or has at least two children.
  - Each node (except root & tree leaves) has between ceil(m/2) and m children. Ceil being the ceiling function.
  - Each path from root to tree leaf has same length.

Explain - a.) Threaded binary trees b.) B+ tree

- a) Threaded binary trees -
  - Every node without a right child has a link(thread) to its INORDER successor.
  - This threading avoids the recursive method of traversing a tree which makes stacks & consumes a lot of memory & time.

- b.) B+ tree -
  - Consists of root, internal nodes and leaves.
  - Root may be a leaf or internal node with two or more children.
  - For a B+ tree of order v, internal nodes contain between v and 2v keys. A node with 'k' keys has 'k+1' children.
  - Leaves exist on same level. They are the only nodes with data pointers.

Differentiate between full, complete & perfect binary trees.

- Full binary tree - Each node is either a leaf or an internal node with exactly two non-empty children.
- Complete binary tree - For a tree with height 'a', every level is completely filled, except possibly the deepest one. At depth 'a', all nodes must be as far left as possible.
- Perfect binary tree - For a tree with height 'd', every internal node has two children and all the leaf nodes exist at same level.

Explain the use of keyword 'register' with respect to variables.

- It specifies that a variable is to be stored in a CPU register.
- 'register' keyword for variables and parameters reduces code size.

- The no. of CPU registers is dependent on its architectural design. Mostly this number is 32.

Define recursion in C.

- A programming technique in which a function may call itself. Recursive programming is especially well-suited to parsing nested markup structures

What is the purpose of "register" keyword?

- The keyword ‘register’ instructs the compiler to persist the variable that is being declared , in a CPU register.

- Ex: register int number;

- The persistence of register variables in CPU register is to optimize the access. Even the optimization is turned off; the register variables are forced to store in CPU register.

Explain #pragma statements.

- Implementations of C & C++ supports features unique to the OS or host machine.
- #pragma directives offer a way for each compiler to offer machine and OS specific features while retaining overall compatibility.
- Pragmas are used in conditional statements, to provide new pre-processor functionality or implementation-defused information to the compiler.

Explain the properties of union. What is the size of a union variable

- With Union same storage can be referenced in different ways.
- When sizeof is applied to a union it shows the size of biggest member.
- Each initialization of union over-writes the previous one - there's no standard way.
- Syntax, format and use of tags and decorators are like struct, but members overlay each other, rather than following each other in memory.

What is the format specifier for printing a pointer value?

- %p format specifier displays the corresponding argument that is a pointer.
- %x can be used to print a value in hexadecimal format.

Why can arithmetic operations not be performed on void pointers?

- We don't know the size of what's being pointed to with a void *. So, it is difficult to determine how far to seek the pointer to get the next valid address.

What are bitwise shift operators?

<< - Bitwise Left-Shift

Bitwise Left-Shift is useful when to want to MULTIPLY an integer (not floating point numbers) by a power of 2.
Expression: a << b

>> - Bitwise Right-Shift

Bitwise Right-Shift does the opposite, and takes away bits on the right.
Expression: a >> b
This expression returns the value of a divided by 2 to the power of b.

What is the translation phases used in C language?

- There are various translation phases that can be used in C language. These are as follows:
    a) The first stage of the translation phase is to check the tri-graph and allow it to be used with the system.
    b) The second stage is to identify the type of program that has to be written for that, the identification of identifiers and others are figured out.
    c) The third stage is the important stage where the translation from comments to the space takes place. The space is not being seen in the case of strings or character constant. And multiple white spaces may be combined in one.
    d) The last stage involves the complete translation of the program.

What are the different types of objects used in C?

- There are two types of objects that are used in C and they are as follows:
    - Internal object: deals with the internal functionality of the program. Any function that is being defined inside is an internal function. Internal objects are used inside the program itself and don't include any external references.
    - External object: deals with the external functions used in the program. All the functions are used as external as they can't be defined inside one another. The C program is always a collection of external objects. The external objects are used for reducing the code and increasing the usability of it. These objects gets involved in the cross-file and library communication.

What is the use of linkage in C language?

- Linkage is the term that is used to describe the accessibility of the objects from one file to another file. This file can be either from the same file or different files. The linkage is really helpful in managing a large number of links when lots of files are linked together with one another. User can declare the same function more than once within the same scope or different scope. The packaging of the library function can be declared to the function in the header as its function is defined in a source file. The source file is included in the header file as:

```
// first.c #include "first.h" int first(int n)
{

...

}
```

- This way there is a surety that the function is defined in the source file and working with the same declarations as it is mentioned in the above files. As it can be seen in here the declaration is done twice. And the second declaration also consisted of the function as well as the definition. The linkage provides a way for the functions to be used in the same scope.

What are the different types of linkage exist in C?

- There are three types of linkages that are used to make an object accessible from one file to another. The linkages that are being provided are as follows:

a. No linkage: defines the linkage that is having internal functionality and internal functions with its arguments and variables internal to the application itself.

b. External linkage: external linkage defines the linkage that is located externally of the program. It is considered as the default linkage for the functions and other parameters that are defined outside the scope of the program. All the instances are referred as the same object if they are preceded with the external linkage. For example printf() is declared externally in as int printf(const char *, …) this is the function that returns an integer.

c. Internal linkage: deals with the names that are internal to the files or the same objects within the same file. This allows the user to define the linkage internally without shifting to many other files for the references. The internal linkage can be done by prefixing the keyword static in front of the object name.

   The following sample code will explain the linkages used:

// this is the second file that includes the first file externally

```
extern int i;
void f ()
{
// Write your own code here
i++;
}
```

Write a program to show the change in position of a cursor using C

- The program that can be made to show the changes in position of a cursor using C includes the libraries that are on the graphics level like <dos.h>. This way the cursor representation can be shown graphically to the user. The program of doing that is as follows:

```
int main()
{
union REGS, n, i;
n.h.b=2; //It is used to position the cursor
n.h.c=0;
n.h.c=40;
n.h.c=50;
int86(0x10,&n,&i);
printf("This will show the positioning of the cursor");
return 0;
}
```

What is the function of pragma directive in C?

- Pragma is a directive with different implementation rules and use. There are many directives that vary from one compiler to another compiler. The pragma directive is used to control the actions of the compiler that is ignored during the preprocessing. It passes the instructions to the compiler to perform various actions without affecting the program as whole.

- pragma-string passes the instructions to the compiler with some of the required parameters. The parameters are as follows:
- Copyright- it specifies the copyright string.

- Copyright_date- it specifies the copyright date for the copyright string. Some version control can be given as parameter.

- The sample code is given below:

```
#pragma pragma-string.
#include
#pragma pragma-string
int main(){
printf("C is powerful language ");
return 0;
}
```

What is the data segment that is followed by C?

- The data segment is the segment that consists of the four parts:
  - Stack area: is the first segment part that consists of all the automatic variables and constants that are stored in the stack area. The automatic variables that are included in C are the local variables that are of default storage class, variable of auto class, integer, character, string constants, etc., function parameters and return values.
  - Data area: consists of all the extern and static variables that are stored in this area. It stores the data permanently the memory variables that are initialized and not initialized.
  - Heap area: consists of the memory that can be allocated dynamically to the processes. The memory can be dynamically allocated by the use of the function malloc() and calloc().
  - Code area: consists of a function pointer that is used to access only the code area. The size of the area remains fixed and it remains in the read only memory area.

What is the function of multilevel pointer in c?

- Multilevel pointer is a pointer that points to another pointer in a series. There can be many levels to represent the pointers. The multilevel pointers are given as:

```
#include<stdio.h>
int main(){
int s=2,*r=&s,**q=&r,***p=&q;
printf("%d",p[0][0][0]);
return 0;
}
```

- In the above code the ***p will be pointing to the pointer of pointer that is **q and **q in case will be pointing to another pointer that is s. This chain will continue if new addition will take place.

What is use of integral promotions in C?

- Integral promotions deals with the promotions that are internally being performed to convert the lower precision level to higher level like shorter to int. The conversations that is being defined includes: A short or char will be converted to the int automatically. If the assignment is not being done then the conversation will be assigned to the unsigned int. Through this the original value and sign can be preserved for further actions. The conversation of char can be treated as signed or unsigned but it will always been implementation dependent. These are helpful in applying a conversation on which sift, unary +, -, and ~ operators can be taken place.

What is the use of ?: operator?

- This ?: operator is used to show the conditional statement that allows the output of the statement to be either true or false. This operator is used as:

```
expression?expression1:expression2
```

- If the expression is true then the result will be expression1. If the expression is false then the result will be expression2. There is only one evaluation take place for the result. The expression that is used with arithmetic operators are easy to solve and it is easy to convert and apply to the expressions.

What is the condition that is applied with ?: operator?

- The condition that has to be applied with the ?: operator is as follows:
- The type of the result will be generated if there are two operands of compatible types. The pointers can be mixed together to give some result. The pointers that are involved for the result has two steps to follow:

a) If an operand is a pointer then the result will be of a pointer type.

b) If any of the operand is a null pointer constant then result will be considered of the operand. The example is shown below:

```
#include <stdio.h>
#include <stdlib.h>
main(){
int i;
for(i=0; i <= 10; i++){
printf((i&1) ? "odd\n" : "even\n");
}
exit(EXIT_SUCCESS);
```

What is the function of volatile in C language?

- Volatile is a keyword that is used with the variables and objects. It consists of the special properties that are related to the optimization and threading. It doesn't allow the compiler to apply the customization on the source code for the values that can't be changed automatically. It allows the access to the memory devices that are mapped to the particular function or the keyword. It also allows the use of variables that is between the function setjmp and longjmp. It handles the signals that are passed for the variables. The operations on the variables are not atomic and they are made before any relationship of the threading occurs.

What is the use of void pointer and null pointer in C language?

- Void pointer: is a type of pointer that points to a value that is having no type. It points to any of the datatype. It is used to take the type automatically on run time and it is used to change from integer or float to string of characters. It allows passing null pointer.

- Null pointer: null pointer is just used as a regular pointer that consists of any pointer type with some special value. It doesn't point to any valid reference or memory address. It is just used for easy to make a pointer free. The result is the result of the type-casting where the integer value can have any pointer type.

What is the process of writing the null pointer?

- The null pointer can be written either by having an integral constant with a value of 0 or this value can be converted to void* type. This can be done by using the cast function. For assigning any pointer type to any other pointer type then it first gets converted to other pointer type and then will appear in the program. The code that is being written for null pointer is:

```
int *ip;
ip = (int *)6;
*ip = 0xFF;
```

What are the different properties of variable number of arguments?

- The properties of variable number of arguments are as follows:
- The first parameter of the argument should be of any data type. The invalid declaration of a function will be done like

- int(a);

- The valid declaration of the function will be like:

- int(char c);
- void(int,float);

- There should not be any passing from one data type to another data type. There is also invalid declaration on the basis of:

- int(int a,int b);

- There should not be any blank spaces in between the two periods. The placing of any other data type can be done in place of ellipsis.

How does normalization of huge pointer works?

- Huge pointer is the pointer that can point to the whole memory of the RAM and that can access all the segments that are present in a program. The normalization can be done depending on the microprocessor of the system. The physical memory of the system is represented in 20 bit and then there is a conversion of 4 byte or 32 bit address. The increment of the huge pointer will also affect the offset and segment address. Through the huge pointer the access and modification of device driver memory, video memory, etc. Huge pointer manages the overall system of the memory model and also normalizes the overall use of the pointers.

What are the different types of pointers used in C language?

- There are three types of pointers used in C language. These pointers are based on the old system architecture. The types are as follows:
    - Near pointer: this is the pointer that points only to the data segment. There is a restriction of using it beyond the data segment. The near pointer can be made by using the keyword as “near”.
    - Far pointer: it will access the total memory of the system and can be use to point to every object used in the memory.
    - Wild pointer: it is a pointer that is not being initialized.

What is the difference between null pointer and wild pointer?

- Wild pointer is used to point to the object but it is not in initialization phase. Null pointer points to the base address of a particular segment. Wild pointer consists of the addresses that are out of scope whereas null pointer consists of the addresses that are accessible and in the scope. Wild pointer is declared and used in local scope of the segment whereas null is used in the full scope.

What is a dangling pointer?

- A dangling pointer is one that has a value (not NULL) which refers to some memory which is not valid for the type of object you expect. For example if you set a pointer to an object then overwrote that memory with something else unrelated or freed the memory if it was dynamically allocated.

Differentiate between: a.) Declaring a variable b.) Defining a variable

- Declaring a variable - Declaration informs the compiler about size and type of variable at the time of compilation. No space is reserved in the memory as a result of declaring the variables.

- Defining a variable - means declaring it and reserving a place for it in the memory.
- Defining a variable = Declaration + Space reservation.

Where are auto variables stored? What are the characteristics of an auto variable?

- Auto variables are defined under automatic storage class. They are stored in main memory.
    - Main memory
    - CPU registers

- Memory is allocated to an automatic variable when the block containing it is called. When the block execution is completed, it is de-allocated.

- Characteristic of auto variables:
    - Stored in: main memory
    - Default value: garbage
    - Scope: Local to the block where it is defined
    - Life: Till the control stays in the block where it is defined

Why do we use namespace feature?

- Multiple library providers might use common global identifiers. This can cause name collision when an application tries to link with two or more such libraries.

- The namespace feature surrounds a library's external declaration with a unique namespace that eliminates the potential for those collisions.

namespace [identifier] { namespace-body }

- A namespace declaration identifies and assigns a name to a declarative region.
- The identifier in a namespace declaration must be unique in the declarative region in which it is used.
- The identifier is the name of the namespace and is used to reference its members.

How are structure passing and returning implemented?

- When structures are passed as arguments to functions - the whole structure is usually pushed on the stack, through as several words as are needed.

- In order to avoid this overhead, usually pointers to structures are used.
- Some compilers simply pass a pointer to the structure, even though they have to produce a local copy to save pass-by-value semantics.
- Structures are usually returned from functions in a position pointed to by an additional, compiler-supplied hidden argument to the function.
- Some old compilers used to use unique, static locations for structure returns.

Why can't we compare structures?

- A way for a compiler to apply structure evaluation that is constant with lower level flavor of C does not exist.
- A plain byte-by-byte comparison could be found on random bits available in unused 'holes' in the structure.
- This filling maintains the arrangement of following fields accurate.
- A field-by-field assessment may require improper amounts of recurring code for larger structures.

What do you mean by invalid pointer arithmetic?

- Invalid pointer arithmetic include:
  - Adding, dividing and multiplying two pointers
  - Adding double or float to pointer
  - Masking or shifting pointer
  - Assigning a pointer of one type to another type of pointer.

What is the purpose of ftell?

- ftell() function is used to get the current file referred by the file pointer.
- ftell(fp); returns a long integer value referring the current location of the file pointed by the file pointer fp.
- If there's an error, it will return -1.

Explain a pre-processor and its advantages.

- Pre-processor practices the source code program before it is sent through the compiler.

  - It increases the readability of a program and enables implementing comprehensive program.
  - It helps in easier modification
  - It facilitates writing convenient programs
  - It helps in easier debugging and testing a portion of the program

What do the 'c' and 'v' in argc and argv stand for?

- The c in argument count, argc stands for the number of the command line argument that the program is invoked with.
- And v in argument vector, argv is a pointer to an array of the character string that contains the argument.

How can we read/write Structures from/to data files?

- In order to compose a structure fwrite() can be used as Fwrite(&e, sizeof(e),1,fp). e refers to a structure variable.
- A consequent fread() invocation will help in reading the structure back from file.
- Calling function fwrite() will write out sizeof(e) byte from the address & e.
- Data files that are written as memory images with function fwrite() will not be portable, especially if they include floating point fields or pointers.
- This happens because structures' memory layout is compiler and machine dependent.

Differentiate between ordinary variable and pointer in C.

- Ordinary variable - It is like a container which can hold any value. Its value can be changed anytime during the program.
- Pointer - Stores the address of the variable.

Explain "far" and "near" pointers in C.

- "Far" and "Near" - non-standard qualifiers available in x86 systems only.
- Near pointer - Refers to an address in known segment only.
- Far pointer - Compound value containing a segment number and offset into that segment

When should you use a type cast?

- Type cast should be used in two cases:
  - To change the type of an operand to an arithmetic operation so that the operation can be performed properly.
  - To cast pointer types to and from void *, to interface with functions that return void pointers.

What are storage classes in C?

- Automatic storage classes : Variable with block scope but without static specifier.
- Static storage classes: Variables with block scope and with static specifier. Global variables with or without static specifier.
- Allocated storage classes: Memory obtained from calls to malloc(), alloc() or realloc().

Explain null pointer.

- Null pointer - A pointer that doesn't point to anything.
- It is used in three ways:
  - To stop indirection in a recursive data structure.
  - As an error value
  - As a sentinel value

How do you initialize pointer variables?

- Pointer variables are initialized in two ways:
  - By static memory allocation
  - By dynamic memory allocation

What is an lvalue?

- lvalue - an expression to which a value can be assigned.
- It is located on the left side of an assignment statement
- lvalue expression must reference a storable variable in memory.
- It is not a constant.

What are the advantages of external class?

- Advantages of external storage class:
  - Retains the latest value with persistent storage.
  - The value is available globally.

Disadvantages of external storage class:

- Storage for external variable exists even when it is not needed.
- Modification of the program becomes difficult.
- It affects the generality of the program.

When should you not use a type cast?

- You should not use type cast to override a constant or volatile declaration as it can cause the failure of effective running of program.
- It should not be used to turn a pointer to one type of structure or data type into another.
- Explain modulus operator. What are the restrictions of a modulus operator?
- Modulus operator - provides the remainder value.
- It is applied to integral operands.
- It can not be applied to float or double.

Explain: a.) Function b.) Built-in function

a.) Function

- When a large program is divided into smaller subprograms - each subprogram specifying the actions to be performed - these subprograms are called functions.
- Function supports only static and extern storage classes.

b.) Built-in function

- Predefined functions supplied along with the compiler.
- They are also called library functions.

Explain: a.) goto b.) setjmp()

- goto - statement implements a local jump of program execution. This statement bypasses the code in your program and jumps to a predefined position. You need to provide the program a labelled position to jump to. This position has to be within the same function. It is not possible to implement goto between different functions.
- setjmp() - implement a non local, far jump of program execution. This function can be implemented between different functions. When setjmp() is used, the current state of the program is saved in a structure of type jmp_buf.

Using goto and setjmp() is not a good programming practice. They cause a wastage of memory leading to a largely reduction in the efficiency of the program.

What do you mean by Enumeration Constant?

- Enumeration is a data type. Using enumeration constant, it is possible to create your own data type and define the values it can take.
- This helps in increasing the readability of the program.

Enum is declared in two parts:

- Part one declares the data types and specifies the possible values called enumerators.
- Part two declares the variables of this data type.

Explain Union. What are its advantages?

- Union - collection of different types of data items.
- Though it has members of different data types, at a time it can hold data of only one member.
- Same memory is allocated to two members of different data types in a Union. The memory allocated is equal to the maximum size of members.
- The data is interpreted in bytes depending on the member accessed.
- Union is efficient when its members are not required to be accessed at the same time.

What is the purpose of main() function?

- Execution starts from main() in C
- It can contain any number of statements

- Statements are executed in the sequence in which they are written
- It can call other functions. The control is passed to that function.

```
E.g. int main();
OR
int main(int argc, char *argv[]);
```

Explain argument and its types.

- Argument : An expression which is passed to a function by its caller for the function to perform its task.

Arguments are of two types: actual and formal.

- Actual argument: These are the arguments passed in a function call. They are defined in the calling function.
- Formal argument: Formal arguments are the parameters in a function declaration. Their scope is local to the function definition. They belong to the function being called and are a copy of actual arguments. Change in formal argument doesn't get reflected in actual argument.

Which of these functions is safer to use : fgets(), gets()? Why?

- Out of the two fgets() is safer.
- gets() receives the string from the keyboard and stops when the "enter" key is hit
- There is no limit to the size of the string which can cause memory over flow.

What are pointers? Why are they used?

Pointers are variables which store the address of other variables.

Advantages of using pointers:

- They allow to pass values to functions using call by reference - especially useful while large sized arrays are passed as arguments to functions.
- They allow dynamic allocation of memory.
- They help in resizing the data structures.

What are “near” and “far” pointers?

- “Near” and “Far” pointers are system specific and the use of it, has been decreased over the years, it has now become obsolete. These are supported by 16-bit programming languages. To know the machine that doesn’t require these kinds of pointers, use a preprocessor or remove the unnecessary “near”, “far” pointer keywords.

How can I change the size of the dynamically allocated array?

- You can change the size of the array if it is defined dynamically, by using the realloc() function. realloc() function’s syntax is written as:
- dynarray = realloc(dynarray, 20 * sizeof(int));
- But, realloc() function can’t just enlarge the memory, and this memory allocation depends on the space available in the system. If realloc() can’t find the space available, then it returns null pointer.

How to write a multi-statement macro?

- Macro are simple statements which are written to, dynamically execute the method or function. The function call syntax is:
  - MACRO(arg1, arg2);

- Macro can be included with the conditional statements using the external else clause. You can include simple expressions in macro with no declarations or loops like:
- #define FUNC(arg1, arg2) (expr1, expr2, expr3)

What's the right way to use errno?

- The errors should be checked by return values. Errno defines among various causes of an error, such as “File not found” or “Permission denied”. Errno detects only when a function doesn’t have a unique or unambiguous errors. Errno can be set manually to 0 to find the errors. Error messages can be made useful if the program name with the error can be printed so that the error can be noticed and solved.

Why can't I perform arithmetic on a void* pointer?

- There are no arithmetic operations that can be performed on void* pointer is because the compiler doesn’t know the size of the pointed objects. Arithmetic operations can only be done on the pointed objects. Some compilers make it an exception to perform the arithmetic functions on the pointer.

How can I return multiple values from a function?

- A function can return multiple values in several ways by using the pointers. These include the examples like hypothetical calculation of polar-to-rectangular coordinate functions. The pointer allows you to have a function in which you can pass multiple values and use them.

Example code:

```
#include
polar_to_rectangular(double a, double b, double *xp, double *yp)
{ a=10;
b=20;
*xp = a;
*yp = b;
}
```

What's the total generic pointer type?

- There is nothing like total generic pointer type, only void* holds the object (i.e. data) pointers. Converting function pointer to type void* is not a portable and it is not appropriate to convert it also. You can use any function type like int*() or void*(), these pointers can be treated as a generic function pointer. The best way to have the portability, is to use void* and a generic pointer combination.

Which one to choose from "initialization lists" or "assignment", for the use in the constructor?

- Initialization list consists of all the member objects, which a constructor initializes. By doing this, performance of the execution increases and compiler does faster processing in the case of initialization list. This is better than the assignment as it is an inefficient way to represent the objects and the performance also degrades, when it gets used. The code runs faster by using the initialization list rather than the assignment.

Can “this” pointer by used in the constructor?

- It is not advised to use “this” pointer inside the constructor, because of the object initialization, as it takes more time for the object to be ready for the execution. You can use “this” pointer in the constructor body part and also in the initialization list. The rules of the programming language have to be known before using “this” operator in the constructor.

What is the "Named Constructor Idiom"?

- Named constructor idiom is a technique that gives better intuitive and safer construction operations for user to perform in the class. The way by which the constructor name and the class name can be differentiated is by storing the name in the parameter list. Using this idiom declaration of all the class's constructor method can be used in private and protected sections. These are the static methods which are used to return the objects of that class in which the constructor is defined.

For example:

```
class t {
public:
t(float x);
t(float r); // ERROR: Overload is Ambiguous: t::t(float)
};
int main()
{
t p = t(5.7); // Ambiguous: Which coordinate system?
// Your statements
}
```

To solve this ambiguity we use the Named Constructor Idiom:

```
#include // To get std::sin() and std::cos()
class t {
public:
static t rectangular(float x); // Rectangular coord's
static Point polar(float radius); // Polar coordinates
// These static methods are the so-called "named constructors"
}
```

-