Real Recursive

C-Interview Questions

SET 1 – Q1 – Q5

Q. Who developed C Programming Language ?

A. Dennis Ritchie at AT & T's Bell Laboratories in 1972.

Q. What is the meaning of **constant** in C ?

A. Constant is an entity whose value remains fixed.

Q. What is a variable in C ?

A. Variable is an entity that acts as storage place for a value.

For ex. int a = 20;

here, a is variable

Q. What are **Keywords** ?

A. Keywords are the special words that provide specific meaning to the language compiler. There are 32 keywords in C language.

Q. Can we use a **keyword** as a **variable-name** ?

A. No.

SET 2 – Q6 – Q10

Q. Explain **comments** in C ?

A. Comments are the user defined statements that ensures the better readability and makes code more understandable. Comments are ignored by compiler. Comments can be single line or even multi line.

Q. Can we do nesting of comments in C ?

A. No, not allowed.

Q. What are **escape sequence** ?

A. Escape sequences are basically control characters used for formatting the output. These are combinations of a backslash and a character in lowercase.

We use '**\n**' for **newline**.

Q. Can a program be **compiled** without **main()** ?

A. Yes, but it will not run.

Q. Is **main** a **keyword** ?

A. No.

SET 3 – Q11 – Q15

Q. Can you differentiate between **variable declaration & variable definition** ?

A. **Variable declaration :** assigning data type to a variable. For ex. : int x;

Variable definition : assigning a value to a variable. For ex. : int x = 10;

Q. What is **modular** programming ?

A. It is an approach in which we divide a program into subprograms where each subprogram has its own set of statements for a task.

Q. What are **Tokens** ?

A. Token refers to the smallest lexical unit of a program. It may be a identifier,keyword, constant etc.

Q. Explain **Identifiers** ?

A. Identifiers refers to the legal name given to a variable or a function. It cannot be a keyword and cannot start with a number.

Q. What could be **maximum length of an identifier** in C ?

A. 32 characters.

SET 4 – Q16 – Q20

Q. What do you know by **scope** of a variable ?

A. Scope defines the area over which the variable's declaration has an effect.

Q. Explain the use of **auto, extern, static, register** storage class ?

A.

Storage Specifier	Storage Place	Initial Value	Scope Area	Life
Auto	Stack	Garbage	Within block	End of block
Extern	Data Segment	Zero	Global	Program
Static	Data Segment	Zero	Within block	Program
Register	Register	Garbage	Within block	End of Block

Q. What are **Data Segments** ?

A. Data Segments contains initialized data.

Q. Why header files are enclosed inside **< >** while some are enclosed under **” ”** ?

A. If any file is enclosed under < >, then compiler searches it in **built-in path**.

If any file is enclosed under ” “, then given file is user defined and hence current working space is checked.

Q. How an **integer** with **negative sign** is **stored** in memory ?

A. Let us assume that we have to store x = -5 to the memory, so :

1. Find **1’s** complement of that number.

2. Add 1 to it.

SET 5 – Q21 – Q25

Q. What is a **Pointer** ?

A. Pointer is a special type of variable that holds the memory address of another variable.

It is denoted as *** (var_name)**.

Q. What is a **NULL pointer** ?

A. Null pointer is a type of pointer that points to nothing.

For ex : char *p = NULL;

Q. What are **Dangling pointers** ?

A. A pointer pointing to a memory address that has been deleted or not available. To solve this, allocate it to NULL.

Q. When we use **void** pointers ?

A. When we don't the know the type of value that has to be stored in a variable, in that case we go for void pointers.

Q. Explain **printf**, **sprintf**, **fprintf** ?

A. **printf :** standard output stream (stdout)

fprintf : prints content in file

sprintf : prints formatted output onto char array.

SET 6 – Q26 – Q30

Q. t++ or t=t+1, which one is recommended more ?

A. t++, as it is a single machine instruction (INS), hence it is faster.

Q. What is the purpose of using **typedef** ?

A. It is used to define alternative names to existing datatype.

Example : **typedef** long long int ll;

Q. What is a **function** ?

A. Function represents the block of statements that perform a specific task.

Syntax : return-type function-name(parameters)

Q. What are **actual & formal** parameters ?

A. Actual parameters : Parameters sent at calling end.

Formal Parameters : Parameters at the recieving end.

Q. When we mention the prototype of a function, are we defining it or declaring it ?

A. We are declaring it.

SET 7 – Q31 – Q35

Q. What is **size_t** ?

A. It is the result of **sizeof() operator**. sizeof() operator returns the size of a variable that belongs to a particular type.

Q. What will happen if we include same header file **twice** ?

A. Error.

Q. Explain **Recursion** ?

A. **Recursion** is the technique in which a function is called by itself again and again.

Q. Which data structure is used during the process of recursion ?

A. Stack for proper function scheduling.

Q. What is **Preprocessor** ?

A. It processes the source code, before compilation, by adding external libraries, unzipping macros etc.

SET 8 – Q36 – Q40

Q. What are **macros** ?

A. Macro is a name given to a block of C statements as a pre-processor directive. Being a pre-processor, the block of code is communicated to the compiler before entering into the actual coding (main () function). A macro is defined with the preprocessor directive, #define.

Q. What is the basic difference between a **function** and a **macro** ?

A. The basic difference is that function is compiled and macro is preprocessed.

Q. How to define **multiline macros** ?

A. Using ‘ \ ‘ (backslash).

Q. **Function vs Macros** – which is better ?

A. It **depends**. Usually macros make the program run faster but increases the program size, whereas function makes program compact and smaller.

Q. What is an **Array** ?

A. Array is a linear data structure having collection of homogenous data items.

SET 9 – Q41 – Q45

Q. What is the **base address of an array** ?

A. Base address represents the address of first element of an array.

Q. What will be the equivalent pointer expression for following : a[i][j] ?

A. *(*(a+i)+j)

Q. Highlight the difference between **Structures** and **Union** ?

A.

- Structure is a user defined data type containing heterogenous type of data. Union is also a user defined data type containing heterogenous type of data.
- Structure is achieved using **struct** keyword.Union is achieved using **union** keyword.
- Memory size of structure is equal to the sum of memory taken by its data members individually. Memory size of union is equal to the size of largest data member.

Q. What is **Self Referential structure** ?

A. A Structure is recognized as self-referential structure, if it has one or more pointers pointing to itself.

Q. What are **Command Line Arguments** ?

A. The arguments which we pass to the main() function during the execution of a program are command line arguments.

Syntax : int main(int argc, char *argv[])

argc = argument count

argv = argument value

SET 10 – Q46 – Q50

Q. What is a String ?

A. String represents a sequence of characters, ending with '\0' (null).

Q. Explain the utility of **stricmp()** ?

A. It compares two strings by ignoring their case.

Q. Explain **goto statements** ?

A. It is used for unconditional branching within a program.

Q. What is **FILE** in FILE *fp; ?

A. FILE is a structure in stdio.h, where as fp is a file pointer.

Q. What is a **NULL** statement ?

A. It is a **non-executable statement**.

SET 11 – Q51 – Q55

Q. What are static function ?

A. Functions having static as **keyword** are static function. We make static function, when we want to restrict access to function.

By default, all functions are static.

Q. Explain the utility of **ellipses operator** ?

A. It is used to recieve the variable number of arguments for a function.

It is denoted as ‘ **…** ‘ .

Syntax : void func(int k, …)

Q. What is Enumeration ?

A. It is the user defined data type. It assigns names to integer constants. It is achieved through keyword **enum**. Enum names can have same values. But no two enums can have same values. Enums are automatically assigned to 0.

Q. Explain the use of **fseek()** in File Handling ?

A. It is used for moving the file pointer internally.

Q. What is **dynamic memory allocation** ?

A. It is the process in which memory is allocated during runtime or execution of a program.

SET 12 – Q56 – Q60

Q. How we achieve **dynamic memory allocation** in C ?

A. We achieve **dynamic memory allocation** using functions like **malloc(), calloc(), free(), realloc()**.

Q. Explain **malloc()** ?

A. **malloc()** is one of the functions used for **dynamic memory allocation**. It takes up single arguments, which denotes the number of bytes to be allocated.

malloc() is **faster** than **calloc()**.

Syntax : int *p = (int *)malloc(sizeof(int))

Q. Explain **calloc()** ?

A. **calloc()** is one of the functions used for **contiguous dynamic memory allocation**. It takes up two arguments, in which first argument denotes the number of bytes to be allocated, and second argument denotes size of each block. It initializes allocated memory by 0.

Syntax : int *p = (int *)calloc(10,sizeof(int))

Q. **malloc() vs calloc() – which is faster ?**

A. **malloc()** is **faster** than **calloc()**, because of one extra step of initializing the allocated memory by 0, in case of **calloc()**.

Q. Explain the functionality of **realloc()** ?

A. It resizes the memory block that was previously allocated with a call to malloc or calloc. It takes two arguments : first parameter is a pointer to a memory block that was previously allocated with malloc, or calloc, second parameter represents the size of new memory block.

SET 13 – Q61 – Q65

Q. When we use **free()** ?

A. It is use to free the memory block that had been allocated dynamically.

Q. What are **wild pointers** ?

A. Wild pointers are uninitialized pointers.

Q. Are **null pointers** and **wild pointers** same ?

A. No.

Q. What are **Bit Fields** ?

A. Bit Fields are used to save space in structures having several binary flags or other small fields.

Q. Can we define an array of **bit field** ?

A. No.

SET 14 – Q66 – Q70

Q. What are the **valid operations** that can be performed on pointers ?

A. 1. Comparison

2. Addition

3. Subtraction

Q. Explain the concept of **#undef** ?

A. #undef is used to undefine the existing macro.

Q. What you can tell me about **far pointers** and **near pointers** ?

A. Near pointers can access upto 2^{16} memory space.

Far pointers can access upto 2^{32} memory space.

Q. Can we have nested comments in C ?

A. No.

Q. Can we left **main() function empty** ?

A. Yes.

SET 15 – Q71 – Q75

Q. Name some **entry** and **exit control** loops ?

A. **Entry control loops :** for, while

Exit control loops : do

Q. What if we insert ' **;** ' **(semi colon)** after a header file ?

A. Program will compile and run with a **warning**.

Q. Why **preprocessors** does not have ' **;** ' at the last ?

A. Semicolon is needed by compiler, for compiling a program. As the name itself suggests that preprocessor directives are processed before compilation of a program, hence it is not needed.

Q. Give the example of **type promotion & type demotion** ?

A.

Q. Explain the various phases in the process of **recursion** ?

A. Recursion has two phases :

1. **Winding Phase :** This phase runs until desired condition is fulfilled.

2. **Unwinding Phase :** Once winding phase gets over, control is transferred back to original call, this is unwinding phase.

SET 16 – Q76 – Q80

Q. What is the basic difference between **getch() & getche()** ?

A. **getch() :** reads single character from keyboard.

getche() : reads single character but output is displayed on screen.

Q. Can we call **main() function recursively** ?

A. Yes.

Q. Is **main()** function a user defined function ?

A. Yes.

Q. When we use ‘ **%p** ‘ inside printf() ?

A. For printing **address location** of a variable.

Q. What is __**STDC**__ ?

A. It is a predefined macro which is used to determine whether your compiler is **ANSI STANDARD** or not.

SET 17 – Q81 – Q85

Q. What are the different types of streams in C File Handling ?

A. Basically, there are two types of file handling streams : Text and Binary.

Q. Can a function in C return more than 1 values ?

A. No, not possible.

Q. Name some predefined macros in C ?

A. __LINE__ , __STDC__ , __FILE__ , __DATE__ , __TIME__ .

Q. What is **#include<pragma.h>** ?

A. The **#pragma** directive is the method specified by the C standard for providing additional information to the compiler, beyond what is conveyed in the language itself.

Q. How to compile & run your C code on **Linux** ?

A. **$ gcc filename.c**

$ ./a.out