**SPANIDEA:**

**1. Explain memory management?**

(Have to explain the memory structure created for program. All section (code, data session, Heap, stack)

1.  Text segment - contains executable instructions. Read only area.  
2. Initialized data segment - initialized, global variables and static variables. Read\write area  
3. Uninitialized data segment- (bss – base source symbol) - Uninitialized, global variables and static variables. Initial value =0  
4. Stack - The stack area contains the program stack, - local variable initial value= garbage  
5. Heap- used for dynamic memory allocation. 

Linux command to print memory: $ gcc memory-layout.c -o memory-layout

$ size memory-layout

-----------------------------------------------------------------------------------------

**2. How you debug your code?**

We are using visual studio for compiling and debugging the code for windows platform, we place break point in the part of code where the application is crashing and we was check why it failing, for Linux we are using gcc gnu complier.

In our application we log the message from every function like the application is entered in this function and exit from function so using this logs we easily find out in which part of code the application is crashing so we will debug that part of code for resolve the issue. Or we enable that part of code in vs and debug this code.

**3. What you use in your development/what you did?** --- **toolchain, vs , makefile**

For development the code we use visual studio 2015 professional for code development and debug and tool chain environment for debug the code for Arm Linux environment(cross compilation) we use make file to build the application for Arm Linux

My role in this project is that I am working as scrum master, I am involve in the design team where we understand customer requirement and design our application flow also I am actively involved in code development, multithreading , IPC, memory warehousing, code review and commit in git. And managing agile scrum process within the tam.

**4. Why you want to change your job?**

I am working in this project from last 3 years and thedevelopment work is going to complete and only support is there so I want to change something to new.

**5. Your project is in which domain.**

This project domain is industrial internet of things (IIoT)

**6. What is function pointer? Write syntax?**

In [C programming language](https://beginnersbook.com/2014/01/c-tutorial-for-beginners-with-examples/), we can have a concept of Pointer to a function known as **function pointer in C**.

So instead of using function name we can call the function using its address.

**function\_return\_type(\*Pointer\_name)(function argument list)**

d**ouble (\*p2f)(double, char)**

Now the above pointer able to point any function which return type is double and having the same no. of and type of variable as pointer

Why do we need an extra bracket around function pointers like

void (\*fun\_ptr)(int) = &fun;

If we remove bracket, then the expression “void (\*fun\_ptr)(int)” becomes “

void \*fun\_ptr(int)” which is declaration of a function that returns void pointer.

**Following are some interesting facts about function pointers.**

**1)** Unlike normal pointers, a function **pointer points to code, not data.** Typically a function pointer stores the start of executable code.

**2)**Unlike normal pointers, we do not allocate de-allocate memory using function pointers.

**3)** A function’s name can also be used to get functions’ address.

**Use of Function pointer:**

**1.** Mainly use in call back function.

2. pointer flexibility- same pointer can point to multiple function which have same parameter and return type as pointer.

3. Function pointer can be used in place of switch case. For example, in below program, user is asked for a choice between 0 and 2 to do different tasks.

|  |
| --- |
| #include <stdio.h>  void add(int a, int b)  {      printf("Addition is %d\n", a+b);  }  void subtract(int a, int b)  {      printf("Subtraction is %d\n", a-b);  }  void multiply(int a, int b)  {      printf("Multiplication is %d\n", a\*b);  }    int main()  {      // fun\_ptr\_arr is an array of function pointers      void (\*fun\_ptr\_arr[])(int, int) = {add, subtract, multiply};      unsigned int ch, a = 15, b = 10;        printf("Enter Choice: 0 for add, 1 for subtract and 2 "              "for multiply\n");      scanf("%d", &ch);        if (ch > 2) return 0;        (\*fun\_ptr\_arr[ch])(a, b);        return 0;  } |

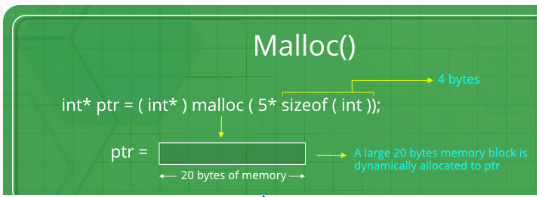
Enter Choice: 0 for add, 1 for subtract and 2 for multiply

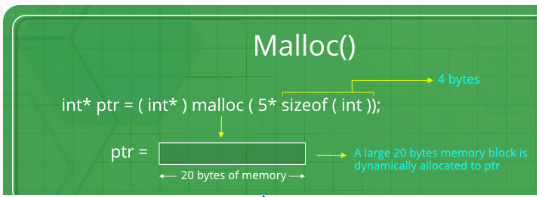
2

Multiplication is 150

4. Like normal data pointers, a function pointer can be passed as an argument and can also be returned from a function.

**6.**  Many object oriented features in C++ are implemented using function pointers in C. For example [virtual functions](https://www.geeksforgeeks.org/virtual-functions-and-runtime-polymorphism-in-c-set-1-introduction/). Class methods are another example implemented using function pointers.





**7. malloc() and calloc() difference and syntax.(syntax, return type, initial**

**value, no of memory block assigned)**

We will use malloc() and calloc() for assigning dynamic memory (heap).

Both are returning void pointer. Malloc () assign memory block of given size in heap and return the pointer for that memory where calloc() take two argument that is size of block and number of block and assign that much of block of given size and return the pointer .

| **BASIS OF COMPARISON** | **MALLOC()** | **CALLOC()** |
| --- | --- | --- |
| No of blocks | Allocates an only single block of requested memory. | Allocates multiple blocks of the requested memory. |
| Syntax | void \*malloc(size\_t size); | void \*calloc(size\_t num, size\_t size); |
| Initialization | malloc() doesn't clear and initialize the allocated memory. | calloc() initializes the allocated memory to zero. |
| Manner of Allocation | malloc() function allocates memory of size 'size' from the heap. | calloc() function allocates memory the size of which is equal to num \*size. |
| Speed | Fast | Comparatively slow. |

Dynamic Memory Allocation (Malloc, calloc, realloc, free)

| Function | Use of Function |
| --- | --- |
| [malloc()](http://www.programiz.com/c-programming/c-dynamic-memory-allocation#malloc) | * memory allocation --ptr=(int\*)malloc(100\*sizeof(int)); * Allocates single block of memory * Return a pointer of type void, so type casting is done * Does not initialize the memory allocated. * Takes one argument that is, number of bytes/ (size of ().) * malloc is faster than calloc |
| [calloc()](http://www.programiz.com/c-programming/c-dynamic-memory-allocation#calloc) | * contiguous allocation -- ptr=(int\*)calloc(n,element-size); * Calloc() allocates multiple blocks of memory each of same size * Returns a pointer to memory * Initializes the allocated memory to ZERO. * Take two arguments : number of blocks and size of each block (Depends on data type) * calloc takes little longer than malloc because of the extra step of initializing the allocated memory by zero |
| [free()](http://www.programiz.com/c-programming/c-dynamic-memory-allocation#free) | dellocate the previously allocated space |
| [realloc()](http://www.programiz.com/c-programming/c-dynamic-memory-allocation#realloc) | Reallocation--ptr=realloc(ptr,newsize)  Change the size of previously allocated space  If the previously allocated memory is insufficient or more than sufficient. Then, you can change memory size previously allocated using realloc(). |

**8. Volatile.**

# Volatile is a keyword in c. which tell the complier to avoided optimization on that variable

# When we are accessing any variable that time control go to address of that variable in memory and fetch the value in CPU and process on it, this process take some time to process.

# When we are accessing any variable again and again (like share variable and data coming from port) , and if the optimization is not disable then to speed-up the code execution complier make the copy of that variable in CPU register and process on that, instead of going to original place of the variable.

# So this is problem when we accessing data which is coming from port or when multiple thread accessing same global variable, and if complier make that variable as register then we may not get latest value of the variable.

# When we declare any variable as volatile then complier always read its value from its original place.

# So volatile keyword is used to tell complier that the value of the variable change any time so read its value from its original location instead of chase this variable value. Eg. volatile int a;

**Proper Use of C volatile Keyword**

1. Memory-mapped peripheral registers

2. Global variables modified by an interrupt service routine

3. Global variables accessed by multiple tasks within a multi-threaded application

4. data coming from port.

**Const** **Volatile Qualifier:-**

Const Volatile means that the program cannot modify the variable's value, but the value can be modified from the outside, thus no optimizations will be performed on the variable

**Not volatile means --value of variable cannot be changed by any external device or hardware**

**interrupt.**

What is meaning of the declaration:

const volatile int a=6;

Answer:

Value of variable cannot be changed by program (due to const) but its value can be changed by external device or hardware interrupt (due to volatile)

**9. Difference between structure and union ?**

Frist Both are user defined data types. types which contains variables of different data types. Both of them have same syntax for definition, declaration of variables and for accessing members. Still there are many difference between structure and union.

i) A structure is a user-defined data type available in C that allows to **combining**

**data items of different kinds**. Structures are used to represent a record.

i). A union is a special data type available in C **that allows storing different data**

**types in the same memory location**

iii). In structure each member get separate space in memory.

**e.g. truct student { int rollno; char gender; float marks; }s1;**

**The total memory required to store a structure variable is equal to the sum of size of all**

**the members. In above case 7 bytes (2+1+4) will be required to store structure variable s1.**

iv). In union, the total memory space allocated is equal to the member with

largest

Size. All other members share the same memory space. This is the biggest

difference between structure and union.

**E.g. union student { int rollno; char gender; float marks; }s1;**

**In above example variable marks is of float type and have largest size (4 bytes). So the**

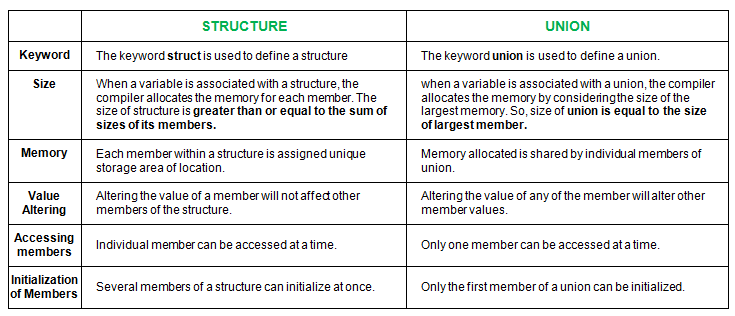
**total memory required to store union variable s1 is 4 bytes.**

Vi). In Structure We can access any member in any sequence without any error.

Vii). In Union code will work fine but will show erroneous output in the case of union. We can access only that variable whose value is recently stored.

**Similarities between Structure and Union**

1. Both are user-defined data types used to store data of different types as a single unit.
2. Their members can be objects of any type, including other structures and unions or arrays. A member can also consist of a bit field.
3. Both structures and unions support only assignment = and sizeof operators. The two structures or unions in the assignment must have the same members and member types.
4. A structure or a union can be passed by value to functions and returned by value by functions. The argument must have the same type as the function parameter. A structure or union is passed by value just like a scalar variable as a corresponding parameter.
5. **‘.’** operator is used for accessing members.



**10. What is Structure padding?**

Structure padding is the process where complier insert extra meaning less bytes in structure memory This extra bytes is called padding.

The computer architecture is such that it access one word at a time to speed up execution so while assigning the memory to structure member, if there is data type change in structure member then complier assign memory from new word.

e.g. Suppose we have a structure struct n{ char a;

char b;

int c; }

So theoretically the size of structure is 6 byte but if we print size then it is not it is more than that because of padding. For structure n

**Padding** [aligns](http://en.wikipedia.org/wiki/Data_structure_alignment) structure members to "natural" address boundaries - say, int members would have offsets, which are mod(4) == 0 on 32-bit platform. Padding is on by default. It inserts the following "gaps" into your first structure:

struct mystruct\_A {

char a;

char gap\_0[3]; /\* inserted by compiler: for alignment of b \*/

int b;

char c;

char gap\_1[3]; /\* -"-: for alignment of the whole struct in an array \*/

} x;

**Packing**, on the other hand prevents compiler from doing padding - this has to be explicitly requested - under GCC it's \_\_attribute\_\_((\_\_packed\_\_)), so the following:

struct \_\_attribute\_\_((\_\_packed\_\_)) mystruct\_A {

char a;

int b;

char c;

};

**11. Multi-threading? Syntax. For creating thread?**

**12. Storage class**

**13. Which IPC you use.**

**14. Socket? How the communication happen with server and client.**

**15. Find output**

Char \*str=”Spanidea”

\*st+1;

Printf(“%c\n”,\*str+1);

ANS: T (2 nd line don’t have any meaning as value not assign to anything, print =value+1 that is T as it comes after s )

16. int i=6,j=7;

J=i--;

Printf(“%d%d”,I,j)

ANS : 56 (post increment)

**17. How to find linklist looping. In singly linklist.**

**18. How to find out merge point in singly linklist?**

---------------------------------------------------------------------------------------------------------

**Mind Tree:**

**Mainly ask bit field and bit operation in MNC and most important code PDF**

1. Print no of set bit and reset bit in 32 bit register

2. I2C details, data format of I2C

3. Set bit and reset bit. Code.

**APTIV:**

1.1 Projects details.

1.2 Tell me about yourself.

1. Storage class?

2. Details on static, if declare global variable as static and register variable as static where it is store initial value.

3. What happen if I declare function as static?

4. Extern variable in details, can we use global variable in other file how? Globle and static in deep

5. Extern int a, is it declaration on definition, what is different between “int a and extern int a”

6. Constant? Where the constant variable stored? What is the initial value? What is diff. const int a & const int a=10;

7. Register variable in details, initial value

8. Memory structure,

9. Difference between structure and union. memory assignment.

10. What is the real use of the union?

11. Structure padding. Size of structure how will calculate

12. Structure bit field.

13 bit field.

14. What is void?

15. Difference between malloc() and calloc(), initial value.

16. Which controller you are used.

17. Write a code for find the given no is in the power of 2. Write code using bit operator and explain.

18. What is the endianness? How to find out the system is little endian or big endian. Write a code.

19. Write a code to reverse link-list

20. Write a code for finding even or odd number in single line using the relational operator? Bitwise operator.

21. Write a code for set 4th no of bit, how to diff in bit and byte, the no is in byte but have to set bit.

22. What is the task?

23. What is the difference between share memory and message queue, which is the faster.

24. How to handle critical section.

25. Difference between semaphore and mutex.

26 if we have a mutex for a critical section can we use another mutex inside this of critical section?

27. Scheduling algorithms.

28. Pointers what is difference

i. int const \*p

ii. int \*const p

iii. const int \*P

iV const \*int p (need to see how to write and diff)

29. Signals details.

30. tcp-ip format

31. Dangling pointer in details, when we get the error and how to fix this.

32. Memory leak.

33. Compilation process? Types of compilers. Complier scripts?

34. Can we write printf() globaly? Why?

35 Write a code for swao two nibble

36 can we define structure variable as static

Extra:

1. [Write your own printf() function in c](http://www.firmcodes.com/write-printf-function-c/)

2. Call Back function

3. can we use pointer with register variable>

4. how will you free the memory without using free()?

5. difference between string and char array.

1. string is the 1 dimensional array ending with\0

2. char array may store in data section or stack but not able to store in code

segment(read only area) BUT string may also store in code section(we can

not able to change this string)

3.  **size of a character can be evaluated using sizeof operator while of strings, strlen function does so. strlen function returns actual no. of characters in the string excluding NULL byte. Therefore, while declaring a character array to be a string, keep one byte extra for NULL terminator plus size, in bytes, for maximum no. of characters in the string.** For example, for a string containing maximum 10 characters, we need to declare character array of size 11 bytes, as:

char msg[11] = "hello dear"; */\* 'msg' is 10 char string \*/*

4. The statements ‘**char s[] = “geeksquiz”**‘ creates a character array which is like any other array and we can do all array operations. The only special thing about this array is, although we have initialized it with 9 elements, its size is 10 (Compiler automatically adds ‘\0’)