Interview Question On C

**1. What is Local variable and What is Global Variable?**

= Local var is a var which is declared with in the brackets or with in the function. This variable is accessed only with in the blocks or functions. When control exit from the block or function then this variable is destroyed.

Global var is a var which is declared outside of the brackets or outside of the function. This variable is accessed through out (any where) of the program . When the entire program is terminated then this variable is destroyed.

**2. What is the use of static variable in C ?**

**=**

* A static var is used because scope of static variable is global. So we can use static variable throughout the program and the life time of it Is until the program is terminated.
* Static variable is initialized by 0.
* Static var is used as a common value, Which is shared by all method.

**int fun()  
{  
  static int count = 0;  
  count++;  
  return count;  
}  
    
int main()  
{  
  printf("%d ", fun());  
  printf("%d ", fun());  
  return 0;  
}  
OUTPUT:  
1   2**

**3. What is format specifier?**

= Format specifier is used during takeing input and output. It says the compiler that what type of data is in a variable at the time of taking input and output.

Int -> %d %o %i %u(o means octal number of the int)

Float-> %f %e

**4. What is Static local and Static global Variable?**

**=** If the variable declared with a static keyword outside the function, then it is known as a **static global variable**. It is accessible throughout the program.

The variable with a static keyword is declared inside a function is known as a **static local variable**. The scope of the static local variable will be the same as the automatic local variables.

**5. What is the use of function in C?**

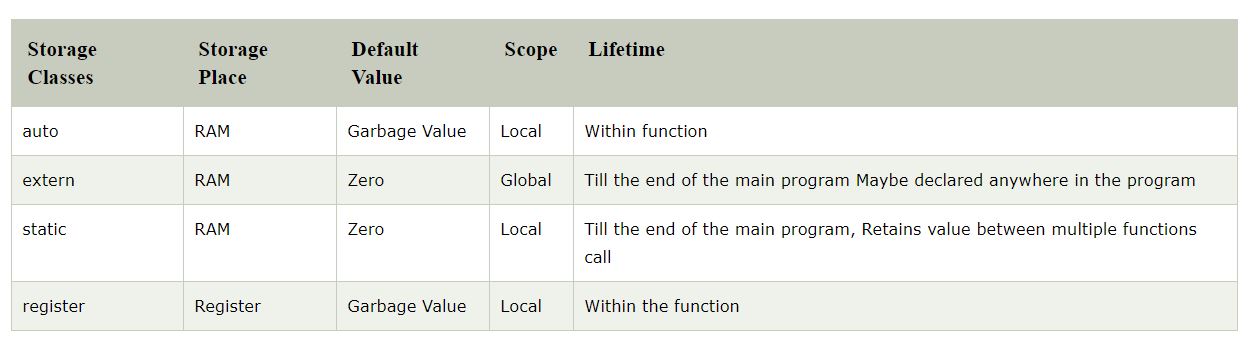
=

* If we use function then we need not to write same peace of code again and again
* We can call a function any where from the program.
* If we divided a code into multiple program then find the bugs or mistake will be become easy

**6. Storage Class is C:**

**=** Storage classes in C are used to determine **the lifetime, visibility, memory location, and initial value** of a variable. There are four types of storage classes in C

* Automatic
* External
* Static
* Register



**Auto=**

By default any variable is a auto storage class variable

**int main()**

**{**

**int a = 10,i;**

**printf("%d ",++a);**

**{**

**int a = 20;**

**for (i=0;i<3;i++)**

**{**

**printf("%d ",a); // 20 will be printed 3 times since it is the local value of a**

**}**

**}**

**printf("%d ",a); // 11 will be printed since the scope of a = 20 is ended.**

**}**

**Op=** 10 24 11 25 12 26

**Extern=**

We can only initialize the extern variable globally, i.e., we can not initialize the external variable within any block or method.

The external storage class is used to tell the compiler that the variable defined as extern is declared with an external linkage elsewhere in the program

The extern variable must be defined anywhere. Otherwise if we try to change the value the it gives us a error

Ex=

**extern int var;**

**int main(void)**

**{**

**var = 10;**

**return 0;**

**}**

**Op= error**

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

**extern int var = 0;**

**int main(void)**

**{**

**var = 10;**

**return 0;**

**}**

**Op= compiled successfully**

* **IF initially we want to not define a variable or not allocate memory then we can use extern storage class. We can initialize this variable with in another file and we have to include the file with in the program where we use it. If we not include that file where it is defined then it gives us a error. We need not to define the variable with the program where we use it. But we have to define it atleast one. Act as a global var.**

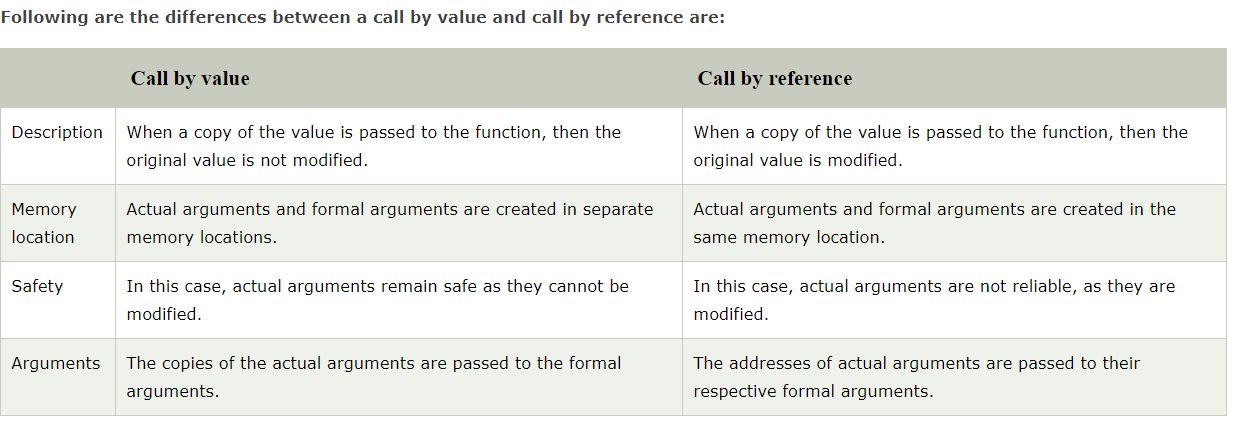
**7. Difference between Declaration and Defination**

= **Declaration** of a variable is for informing to the compiler about the name of the variable, type of value it holds. So declaration gives details about the properties of a variable. Whereas, **Definition** of a variable says where the variable gets stored. i.e., memory for the variable is allocated during the definition of the variable.

**8. What is Call by value and call by references in C?**

**=** when we pass the variable as an argument with in the function then its known as Call by Value

when we pass the address of variable as an argument with in the function then its known as Call by references



**9. What is Static memory allocation and dynamic memory allocation. State the difference among them.**

= **Static memory allocation** is a type of memory allocation where memory allocation done at the time of compiletion. In this type of memory allocation if we need more memory at the time of execution then we can’t increase the memory. (memory can’t be increased at the time of execution)

**Dynamic memory allocation** is a type of memory allocation where memory allocation done at run time. In this type of memory allocation if we need more memory at the time of execution then memory can be increased. (memory can be increased at the time of execution)

|  |  |
| --- | --- |
| **STATIC MEMORY ALLOCATION** | **DYNAMIC MEMORY ALLOCATION** |
| 1. Memory is allocated at compile time | 1. Memory is allocated at run time |
| 2. memory can’t be increased at the time of execution | 2. memory can be increased at the time of execution |
| 3. It is used in array | 3. It is used in link list |

\* **Dynamic memory allocation in c language is possible by 4 functions of stdlib.h header file.**

**10. Explain the function malloc() , calloc():**

= **malloc()**

* With the help of malloc() we can allocates a single block of memory
* Initially it consist garbage value
* It returns null at the time of failure

ptr=(cast-type\*)malloc(byte-size)

**calloc()**

* With the help of calloc() we can allocates a multiple block of memory
* Initially it consist 0
* It returns null at the time of failure

ptr=(cast-type\*)calloc(n,byte-size)

**11. What is recursion in C?**

**=** When a function calls itself, and this process is known as recursion. The function that calls itself is known as a recursive function.

Recursive function comes in two phases:

1. Winding phase
2. Unwinding phase

**Winding phase**: When the recursive function calls itself, and this phase ends when the condition is reached.

**Unwinding phase**: Unwinding phase starts when the condition is reached, and the control returns to the original call.

**12. What is pointer in C?**

**=** A pointer is a variable that refers to the address of a value. It makes the code optimized and makes the performance fast. When we declare a variable in program then a memory is allocated for the variable. This memory have an address. Pointer variable holds this address. The data type of the pointer is the same as the variable which address is stored by the specific pointer.

**13. What is Uses of pointer?**

**=**

* With the help of pointer we can traverse a array or a string
* It is used in Dynamic memory allocation
* It is used in structure, link list, graph, tree

**14. What is Null pointer?**

**=** NULL is used to indicate that the pointer doesn’t point to a valid location. Ideally, we should initialize pointers as NULL if we don’t know their value at the time of declaration.

**15. What is Dangling pointer? How to overcome this problem?**

**=** Dangling Pointer is a pointer that doesn’t point to a valid memory location. Dangling pointers arise when an object is deleted or deallocated, without modifying the value of the pointer, so that the pointer still points to the memory location of the deallocated memory.

**void main()**

**{**

**int \*ptr = malloc(constant value); //allocating a memory space.**

**free(ptr); //ptr becomes a dangling pointer.**

**}**

In the above example, initially memory is allocated to the pointer variable ptr, and then the memory is deallocated from the pointer variable with the help of free func. Now, pointer variable, i.e., ptr becomes a dangling pointer.

The problem of a dangling pointer can be overcome by assigning a NULL value to the dangling pointer. Let's understand this through an example:

**void main()**

**{**

**int \*ptr = malloc(constant value); //allocating a memory space.**

**free(ptr); //ptr becomes a dangling pointer.**

**ptr=NULL; //Now, ptr is no longer a dangling pointer.**

**}**

**16. What is far pointer in C?**

**=** A pointer which can access all the 16 segments (whole residence memory) of RAM is known as far pointer. A far pointer is a 32-bit pointer that obtains information outside the memory in a given section.

**17. What is pointer to pointer in C?**

= In case one pointer refers to the address of another pointer. The pointer to pointer is a chain of pointers. Generally, the pointer contains the address of a variable. The pointer to pointer contains the address of a first pointer.

**Int \*ptr,\*\*pter;**

**Ptr=&a;**

**Pter= &ptr;**

**18. Difference between calloc and malloc function?**

=

|  |  |
| --- | --- |
| **Calloc()** | **Malloc()** |
| 1. calloc func allocates multiple blocks of memory | 1. malloc func allocates a single blocks of memory |
| 2. It is initialized by 0 | 2. It is initialized by garbage value |
| 3. It Consist of two argument | 3. It consist of one argument |

**19. What is Structure?**

= The structure is a user define data type which allow us to make a user define data type with multiple data type.

The data types by which a structure variable is made that is known as **member of the structure.**

It occupies the **sum of the memory** of all members.

The member can be accessed through structure variable

**20. What is Union?**

**=** The Union is a user define data type which allow us to make a user define data type with multiple data type.

The data types by which a Union variable is made that is known as **member of the union.**

it doesn't occupy the sum of the memory of all members. It holds the memory of the **largest member** only.

**21. What is Auto keyword in C?**

**=** Auto is a Storage class in C. The Scope of a Auto variable is Local. That’s mean we can access a auto variable with in the block or with in the function itself. This variable destroyed when the control get outside from the block or function.

**22. What is the purpose of sprintf() function?**

= The sprintf() stands for "string print." The sprintf() function does not print the output on the console screen. It returns the total number of characters present in the string.

int sprintf ( char \* str, const char \* format, ... );

**int main()**

**{**

**char a[20];**

**int n=sprintf(a,"javaToint");**

**printf("value of n is %d",n);**

**return 0;}**

**Output:value of n is 9**

**23. Can we compile a program without main() function?**

= Yes, we can compile, but it can't be executed.

But, if we use #define, we can compile and run a C program without using the main() function.

**#include<stdio.h>**

**#define start main**

**void start() {**

**printf("Hello");**

**}**

**24. What is pointer to an array and what is array of pointer in c?**

= **Pointer to an array** means a pointer which points to an array. It is used for access an array.

int a[3] = {3, 4, 5 };

int \*ptr = a;

Here we declare a pointer which can store only base address of the array not the entire element of the array.

Now if we want to store the entire array element with in the pointer then the Syntax will be little bit changed.

// pointer to an array of five numbers

int (\* ptr)[5] = NULL;

The above declaration is the pointer to an array of five integers.

**int(\*a)[5];**

**int b[5] = { 1, 2, 3, 4, 5 };**

**// Points to the whole array b**

**a = &b;**

**Array of pointer** means an array which store the memory references, not the value.

int \*ptr[3];

int arr[] = { 1, 2, 3 };

    int i, \*ptr[SIZE];

    for (i = 0; i < SIZE; i++) {

        // assigning the address of integer.

        ptr[i] = &arr[i];

    }

**25. What is library function in C?**

= C Standard library functions are inbuilt functions in C programming.

The function definitions of these functions are present in their respective header files. To use these functions we need to include the header file in our program.

For Example if we want to use printf() function then We have to include Stdio.h header file

**26. What is Preprocessor?**

= The C preprocessor is a micro processor that is used by compiler to transform your code before **compilation**. It is called micro preprocessor because it allows us to add macros.

All preprocessor directives starts with hash # symbol.

Let's see a list of preprocessor directives #include,#define,#undef,#ifdef,#ifndef,#if,#else,#elif,#endif,#error,#pragma

**27. What is Macros in C?**

= A macro is a segment of code which is replaced by the value of macro. Macro is defined by #define directive. There are two types of macros:

1. Object-like Macros
2. Function-like Macros

**Object-like Macros**

= The object-like macro is an identifier that is replaced by value. It is widely used to represent numeric constants. For example:

**#define PI 3.14**

Its means that if we use “PI” in our code instead of 3.14 then the “pi” is replaced to 3.14 by compiler.

**Function-like Macros**

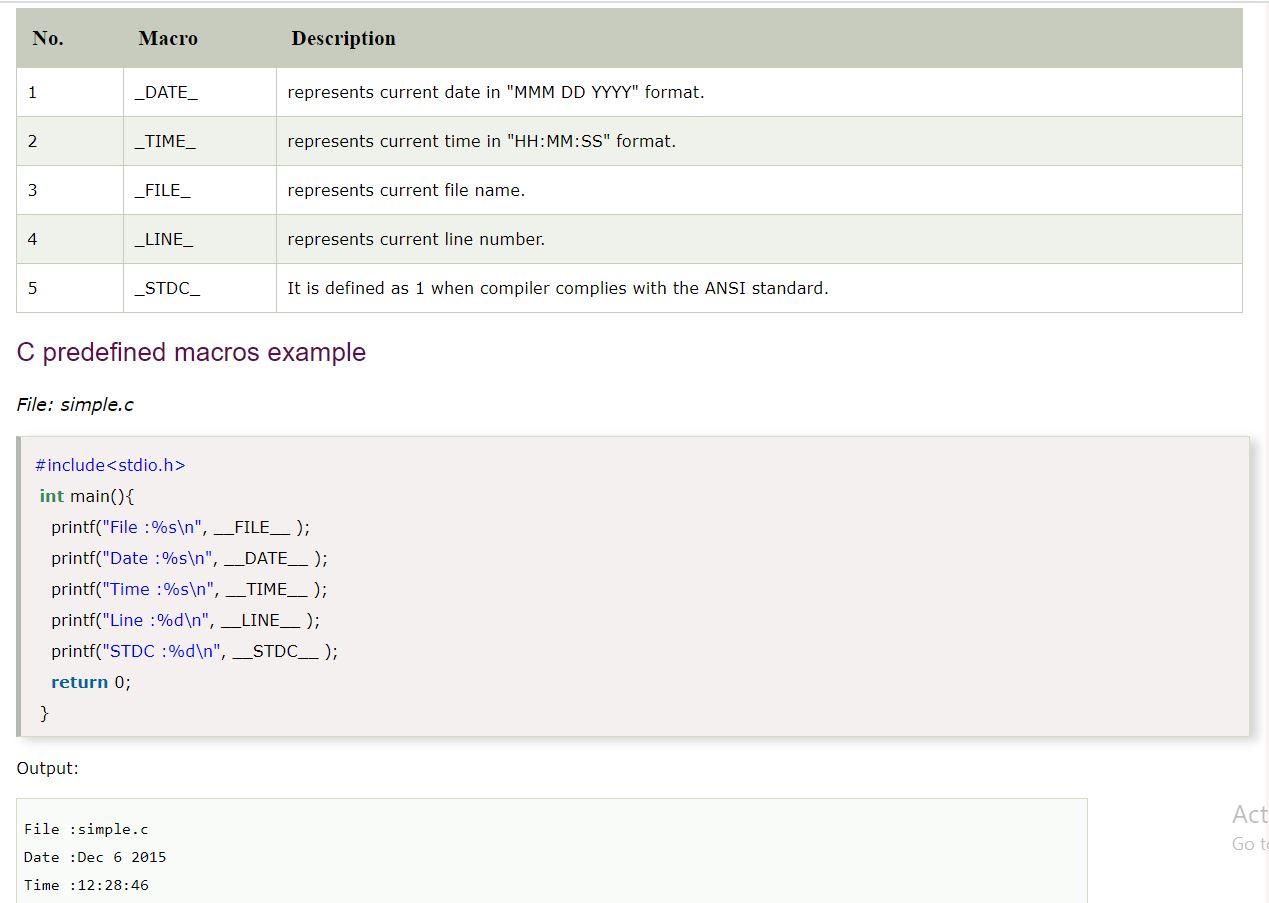
**=** The function-like macro looks like function call

**#define MIN(a,b) ((a)<(b)?(a):(b)**

Here min is macro name.

**28. What is Predefined macros in C?**

=



**29. What is the meaning of #include?**

= #include is a preprocesser directive. It says to compiler that in the program there are some predefine functions and the function definition of that predefine functions is declared with in some files and with the help of this preprocessor directive we are adding or including that header file in our program.

**30. What is command line argument?**

= The argument is passed through the main at the time of execute the program is known as command line argument. The syntax is given bellow.

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

Within the main we have to pass two argument. One is “argc” which defines the argument count. That’s mean how many argument is passed at run time. The second argument is argv[], a char array pointer. Its holds the actual value which is passed at run time. All the value which is stored in the argv[] that is in the form of string. We can convert this string into integer with the help of **atoi()** function.

Let consider a program where I give a input that is= messi 10

Then the memory representation will be like that=

0 1 2

|  |  |  |
| --- | --- | --- |
| .a/.out | Messi | 10 |

**Argv[]**

The input Will start to store from argv[1]. All are in string format.

**31. What is Structure Padding?**

**=** Structure padding is a concept in C that adds the one or more empty bytes between the memory addresses to align the data in memory and reduce the CPU cycle to access a single variable. It is done by compiler itself.

**32. Why Structure padding is needed?**

= Let assume a code

struct student

{

char a; // 1 byte

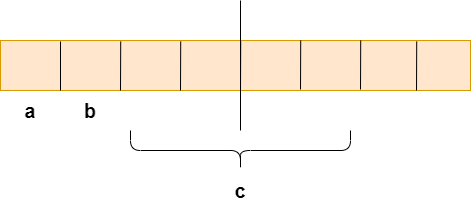
char b; // 1 byte

int c; // 4 bytes

}

We know that in a 32 bit machine processor can read 4 byte at a time and in a 64 bit machine processor can read 8 byte at a time.

Now if we are try to run this code in a 32 bit machine then we will face a problem. The Problem is discussed below.

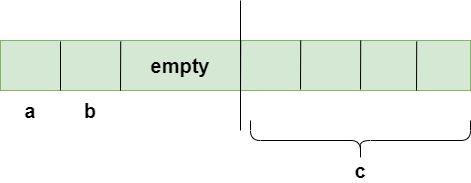


As we know that structure occupies the contiguous block of memory as shown in the above diagram, i.e., 1 byte for char a, 1 byte for char b, and 4 bytes for int c.

We know that 4 byte can be accessed at a time. Now compiler first read the **“a variable”** which is of 1 byte, then **“b”** which is also of 1 byte then it comes to **“c”.** Already 2 byte is accessed by processor, It can accessed two bytes more in this CPU cycle. But Size of the c is 4 byte. So to read the **“c variable”** it require one more CPU cycle. To read **“C variable”** we need total 2 CPU cycle, which is a wastage of CPU cycle. **That’s why we need structure padding.**

**33. How the structure padding is done?**

=



To achieve the structure padding, an empty row is created on the left, as shown in the above diagram, and the two bytes which are occupied by the 'c' variable on the left that are shifted to the right from left. So, all the four bytes of 'c' variable are on the right. Now, the 'c' variable can be accessed in a single CPU cycle. After structure padding, the total memory occupied by the structure is 8 bytes (1 byte+1 byte+2 bytes+4 bytes), which is greater than the previous one.

**34. State the difference between Structure and Union?**

**=**

|  |  |
| --- | --- |
| **STRUCTURE** | **UNION** |
| 1. The keyword Struct is used to define a Structure. | 1. The keyword union is used to define a Union. |
| 2. The required memory is greater than or equals to the sum of size of its members. | 2. The required memory is equals to the largest size member |
| 3. Individuals members can be accessed at a time | 3. At a time only one member can be accessed. |
| 4. Alternating the value of members will not effect other member of structure | 4. Alternating the value of members will effect other member of union |

**35. What is nested Structure in C?**

= **Nested structure in C** is nothing but **structure** within **structure**. One **structure** can be declared inside other **structure** as we declare **structure** members inside a **structure.**

**Separate Structure:**

Here, we create two structures, but the dependent structure should be used inside the main structure as a member.

struct Date

{

int dd;

int mm;

int yyyy;

};

struct Employee

{

int id;

char name[20];

struct Date doj;

}emp1;

**Embedded structure:**

Here we have to declare structure with in structure

struct Employee

{

int id;

char name[20];

struct Date

{

int dd;

int mm;

int yyyy;

}doj;

}emp1;

**Emp1.doj.dd**

**36. What is ASCII code in C?**

= The full form of ASCII is the **American Standard Code for information interchange**. It is a character encoding scheme used for electronics communication. Each character or a special character is represented by some ASCII code, and each ascii code occupies 7 bits in memory. A character variable does not contain a character value itself rather the ascii value of the character variable.

**37. What is generic pointer?**

= A void or generic pointer is a pointer that has no data type with it. A void pointer can hold address of any type and can be typcasted to any type.

**#include<stdio.h>**

**int main()**

**{**

**int a = 10;**

**void \*ptr = &a;**

**printf("%d", \*(int \*)ptr);**

**return 0;**

**}**

**38. What is type-casting?**

= 1st year note

**39. Difference between while and Do-while loop?**

= 1st year note

**40. Difference between break and continue keyword?**

= 1st year note

**41. What is fall through in C?**

= In Switch case we directly hit the desire case. After return control will blindly go through the all the rest case operation. This is known as fall through.

Using break keyword is the remedy of this problem

**42. What is the Difference between getch() and getche() ?**

= The **getch()** function reads a single character from the keyboard. It doesn't use any buffer, so entered data will not be displayed on the output screen.

The **getche()** function reads a single character from the keyword, but data is displayed on the output screen. Press Alt+f5 to see the entered character.

\*\* String ar short code like palindrome, upper case to lower case ………