C & Embedded C

Difference between Declaration and Definition of a variable.

- Declaration of a variable is merely specifying the data type of a variable and the variable name. As a result of the declaration, we merely tell the compiler to reserve the space for a variable in the memory according to the data type specified.

- Whereas, a definition is an implementation/instantiation of the declared variable where we tie up appropriate value to the declared variable, so that linker will be able to link references to the appropriate entities.

C keywords:

- ***Break:*** *Used when it is otherwise to terminate the loop using the condition expression and conditional statements.*
- ***Continue:*** *Used when it is otherwise to ignore the remaining portion of the loop using conditional statements.*
- ***Goto label:*** *Transfers control unconditionally to the desired location.*
- ***Typedef:*** *The typedef declaration provides a way to declare an identifier as a type alias, to be used to replace a possibly complex type name.*
- ***Default:*** *Switch case statements are used to execute only specific case statements based on the switch expression. If switch expression does not match with any case, default statements are executed by the program.*

Sizeof type in C

32 Archi:

- *char ==> 1 byte*
- *signed char ==> 1 byte*
- *unsigned char ==> 1 byte*
- *int ==> 4 byte*
- *unsigned int ==> 4 byte*
- *short ==> 2 byte*
- *unsigned short ==> 2 byte*
- *long ==> 8 byte*
- *unsigned long ==> 8 byte*
- *float ==> 4 byte*
- *double ==> 8 byte*
- *long double ==> 10 byte*

Difference between including file with angular brace <> and double quote " ":

#include:

The #include preprocessor directive is used to paste code of given file into current file. It is used include system-defined and user-defined header files. If the included file is not found, the compiler renders error. It has three variants:

- #include <file>

This variant is used for system header files. It searches for a file named file in a list of directories specified by you, then in a standard list of system directories.

- #include "file"

This variant is used for header files of your own program. It searches for a file named file first in the current directory, then in the same directories used for system header files. The current directory is the directory of the current input file

- #include anything else

This variant is called a computed #include. Any #include directive whose argument does not fit the above two forms is a computed include

Rvalue & Lvalue:

The expression appearing on the right side of the assignment operator is called an rvalue. Rvalue is assigned to lvalue, which appears on the left side of the assignment operator. The lvalue should designate to a variable not a cons

Can a program be compiled without the main() function?

Yes, it can be but cannot be executed, as the execution requires main() function definition.

What is the advantage of declaring void pointers?

When we do not know what type of the memory address the pointer variable is going to hold, then we declare a void pointer for such.

Where an automatic variable is stored?

Every local variable by default being an auto variable is stored in stack memory.

Pointer of Pointer:

It's a pointer variable which can hold the address of another pointer variable. It de-refers twice to point to the data held by the designated pointer variable.

*Eg: int x = 5, *p=&x, **q=&p;*

*Therefore 'x' can be accessed by **q.*

What is a nested structure?

A structure containing an element of another structure as its member is referred to.

What is a self-referential structure?

A structure containing the same structure pointer variable as its element is called a self-referential structure.

Does a built-in header file contain built-in function definition?

No, the header file only declares function. The definition is in the library which is linked by the linker.

Explain modular programming.

Dividing the program into sub programs (modules/function) to achieve the given task is a modular approach. More generic functions definition gives the ability to re-use the functions, such as built-in library functions.

The macro

A macro is a segment of code which is replaced by the value of macro. Macro is defined by #define directive.

The C preprocessor

The C preprocessor is a macro processor that is used automatically by the C compiler to transform your program before actual compilation (Preprocessor directives are executed before compilation.). It is cal

What are bit fields?

We can create integer structure members of differing size apart from non-standard size using bit fields. Such structure size is automatically adjusted with the multiple of integer size of the machine.

What are command line arguments?

*The arguments which we pass to the main() function while executing the program are called as command line arguments. The parameters are always strings held in the second argument (below in args) of the function which is an array of character pointers. First argument represents the count of arguments (below in count) and updated automatically by the operating system: main(int count, char *args[]) {}*

Function Call:

- *Call by Value: the call by value method of passing argument to a function copies the actual value of argument into the formal parameter of the function. In this cas, changing mode to the parameter inside the function has no effect on the argument.*
- *Call by reference: The function copie the @ of an argument into the formal parameters. Inside the function, the @ is used to access the actual argument used in the call. It means the change mode to the parameter affects the passed argument.*

Where the address of operator (&) cannot be used?

- *It cannot be used on constants.*
- *It cannot be used on variables which are declared using register storage class.*

Is FILE a built-in data type?

No, it is a structure defined in stdio.h.

What is a reminder for 5.0 % 2?

Error, It is invalid that either of the operands for the modulus operator (%) is a real number.

How many operators are there under the category of ternary operators?

There is only one operator and is a conditional operator (? :).

Which compiler switch to be used for compiling the programs using the math library with gcc compiler?

Option –lm to be used as > gcc –lm <file.c>

Which operator is used to continue the definition of macro in the next line?

Backward slash (\) is used.

*E.g. #define MESSAGE "Hi, *

Welcome to C"

Which operator is used to receive the variable number of arguments for a function?

Ellipses (…) is used for the same. A general function definition looks as follows

```
void f(int k,…)  {
}
```

Which built-in function can be used to move the file pointer internally?

fseek()

What is an infinite loop?

A loop executing repeatedly as the loop-expression always evaluates to true such as

```
while(1) {
}
```

What is the default value of local and global variables?

Local variables get garbage value and global variables get a value 0 by default.

What is typecasting?

Typecasting is a way to convert a variable/constant from one type to another type.

Can a function return multiple values to the caller using return reserved word?

No, only one value can be returned to the caller.

Is there a way to compare two structure variables?

There is no such. We need to compare element by element of the structure variables.

When should we use pointers in a C program?

1. *To get address of a variable*
2. *For achieving pass by reference in C: Pointers allow different functions to share and modify their local variables.*
3. *To pass large structures so that complete copy of the structure can be avoided.*
4. *To implement "linked" data structures like linked lists and binary trees.*

What is a memory leak? Why it should be avoided

Memory leak occurs when programmers create a memory in heap and forget to delete it. Memory leaks are particularly serious issues for programs like daemons and servers which by definition never terminate.

What is the difference between the local variable and global variable in C?

	Local Variable	***Global Variable***
Declaration	*A variable which is declared inside a function or block is known as a local variable.*	*A variable which is declared outside a function or block is known as a global variable.*
Scope	*The scope of a variable is available within a function in which they are declared.*	*The scope of a variable is available throughout the program.*
Access	*Variables can be accessed only by those statements inside a function in which they are declared.*	*Any statement in the entire program can access variables.*
Life	*Life of a variable is created when the function block is entered and destroyed on its exit.*	*Life of a variable exists until the program is executing.*
Storage	*Variables are stored in a stack unless specified.*	*The compiler decides the storage location of a variable.*

The difference between call by value and call by reference in C

	Call by Value	***Call by reference***
Description	*When a copy of the value is passed to the function, then the original value is not modified.*	*When a copy of the value is passed to the function, then the original value is modified.*
Memory location	*Actual arguments and formal arguments are created in separate memory locations.*	*Actual arguments and formal arguments are created in the same memory location.*
Safety	*In this case, actual arguments remain safe as they cannot be modified.*	*In this case, actual arguments are not reliable, as they are modified.*
Argument	*The copies of the actual arguments are passed to the formal arguments.*	*The addresses of actual arguments are passed to their respective formal arguments.*

What is an array in C?

An Array is a group of similar types of elements. It has a contiguous memory location. It makes the code optimized, easy to traverse and easy to sort. The size and type of arrays cannot be changed after its declaration.

Arrays are of two types:

1. **One-dimensional array:** One-dimensional array is an array that stores the elements one after the other.
2. **Multidimensional array:** Multidimensional array is an array that contains more than one array.

What is the structure?

The structure is a user-defined data type that allows storing multiple types of data in a single unit. It occupies the sum of the memory of all members. The structure members can be accessed only through structure variables.

Structure variables accessing the same structure but the memory allocated for each variable will be different.

Syntax of structure:

```
struct student
{
   char name[10];
   int age;
}s1;
```

What is a union?

The union is a user-defined data type that allows storing multiple types of data in a single unit. However, it doesn't occupy the sum of the memory of all members. It holds the memory of the largest member only.

In union, we can access only one variable at a time as it allocates one common space for all the members of a union.

What is the purpose of the sprintf() function?

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

Syntax

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

Let's see a simple example

```
#include<stdio.h>
int main()
{
 char a[20];
 int n=sprintf(a,"javaToint");
 printf("value of n is %d",n);
 return 0;
}
```

What is a token?

The Token is an identifier. It can be constant, keyword, string literal, etc. A token is the smallest individual unit in a program. C has the following tokens:

- Identifiers: Identifiers refer to the name of the variables.
- Keywords: Keywords are the predefined words that are explained by the compiler.
- Constants: Constants are the fixed values that cannot be changed during the execution of a program.
- Operators: An operator is a symbol that performs the particular operation.
- Special characters: All the characters except alphabets and digits are treated as special characters

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.

Can we access the array using a pointer in C language?

Yes, by holding the base address of the array into a pointer, we can access the array using a pointer.

What do you mean by the Scope of the variable? What is the scope of the variables in C?

Scope of the variable can be defined as the part of the code area where the variables declared in the program can be accessed directly. In C, all identifiers are lexically (or statically) scoped.

What is the main difference between the Compiler and the Interpreter?

Compiler is used in C Language and it translates the complete code into the Machine Code in one shot. On the other hand, Interpreter is used in Java Programming Language and other high-end programming languages. It is designed to compile code in line by line fashion.

Where can we not use &(address operator in C)?

We cannot use & on constants and on a variable which is declared using the register storage class.

Explain toupper() with an example.

Toupper() is a function designed to convert lowercase words/characters into upper case.

Explain Local Static Variables and what is their use?

A local static variable is a variable whose life doesn't end with a function call where it is declared. It extends for the lifetime of the complete program. All calls to the function share the same copy of local static variables.

Which statement is efficient and why? x=x+1; or x++;

x++; is the most efficient statement as it is just a single instruction to the compiler while the other is not.

Which variable can be used to access Union data members if the Union variable is declared as a pointer variable?

Arrow Operator(->) can be used to access the data members of a Union if the Union Variable is declared as a pointer variable.

How can you print a string with the symbol % in it?

There is no escape sequence provided for the symbol % in C. So, to print % we should use '%%' as shown below.

```
printf("there are 90%% chances of rain tonight");
```

Explain the #pragma directive.

The following points explain the Pragma Directive.

- *This is a preprocessor directive that can be used to turn on or off certain features.*
- *It is of two types #pragma startup, #pragma exit and pragma warn.*
- *#pragma startup allows us to specify functions called upon program startup.*
- *#pragma exit allows us to specify functions called upon program exit.*
- *#pragma warn tells the computer to suppress any warning or not.*

Which structure is used to link the program and the operating system?

- *The structure used to link the operating system to a program is file.*
- *The file is defined in the header file "stdio.h"(standard input/output header file).*
- *It contains the information about the file being used, its current size and its location in memory.*
- *It contains a character pointer that points to the character that is being opened.*
- *Opening a file establishes a link between the program and the operating system about which file is to be accessed.*

What are the limitations of scanf() and how can it be avoided?

The Limitations of scanf() are as follows:

- *scanf() cannot work with the string of characters.*
- *It is not possible to enter a multi word string into a single variable using scanf().*
- *To avoid this the gets( ) function is used.*
- *It gets a string from the keyboard and is terminated when the enter key is pressed.*
- *Here the spaces and tabs are acceptable as part of the input string.*

Suppose a global variable and local variable have the same name. Is it possible to access a global variable from a block where local variables are defined?

No. It is not possible in C. It is always the most local variable that gets preference.

What is the difference between ++a and a++?

'++a" is called a prefixed increment and the increment will happen first on a variable. 'a++' is called postfix increment and the increment happens after the value of a variable used for the operations.

Describe the difference between = and == symbols in C programming?

'==' is the comparison operator which is used to compare the value or expression on the left-hand side with the value or expression on the right-hand side.

'=' is the assignment operator which is used to assign the value of the right-hand side to the variable on the left-hand side.

What is debugging?

Debugging is the process of identifying errors within a program. During program compilation, errors that are found will stop the program from executing completely. At this state, the programmer would look into the possible portions where the error occurred. Debugging ensures the removal of errors, and plays an important role in ensuring that the expected program output is met

What are run-time errors?

These are errors that occur while the program is being executed. One common instance wherein run-time errors can happen is when you are trying to divide a number by zero. When run-time errors occur, program execution will pause, showing which program line caused the error.

What are actual arguments?

When you create and use functions that need to perform an action on some given values, you need to pass these given values to that function. The values that are being passed into the called function are referred to as actual arguments.

What are formal parameters?

In using functions in a C program, formal parameters contain the values that were passed by the calling function. The values are substituted in these formal parameters and used in whatever operations as indicated within the main body of the called function

How do you search data in a data file using a random access method?

Use the fseek() function to perform random access input/output on a file. After the file was opened by the fopen() function, the fseek would require three parameters to work: a file pointer to the file, the number of bytes to search, and the point of origin in the file.

How are global variables different from static variables?

Global variables are variables with global scope, i.e., they are accessible throughout the program, unless shadowed. These variables are defined outside a function or code block. Static variables are variables allocated statically, i.e., their value can't be changed. It is fixed for the entire run of a program. They can be defined outside as well as inside functions. Moreover, they can be accessed from anywhere inside the program.

Compare static memory allocation with dynamic memory allocation?

Answer: Following are the important differences between static and dynamic modes of memory allocation:

Memory increase:

In dynamic memory allocation, memory can be increased while executing the program. This is not the case; however, with the static memory allocation where the option of increasing memory during program execution is not available.

Memory requirement

Static memory allocation needs more memory space compared to dynamic memory allocation.

Used in

Arrays use static memory allocation while linked lists use dynamic memory allocation.

When does it happen?

Static memory allocation takes place at compile-time, while dynamic memory allocation happens at runtime.

Explain what you understand by while(0) and while(1)?

Answer: while(0) means that the looping conditions will always be false, i.e., the code inside the while loop will not be executed. On the opposite, while(1) is an infinite loop. It runs continuously until coming across a break statement mentioned explicitly.

How will you resolve the scope of a global symbol?

We can use the extern storage specifier for resolving the scope of a global symbol. The extern keyword is used for extending the visibility or scope of variables and functions. As C functions are visible throughout the program by default, its use with function declaration or definition is not required.

List the difference between the source and object code.

Source code: *Source codes usually get a command from the programmer and these codes are responsible for instructing the computer regarding what to perform? With extension .C we can save this code in C programming.*

Object code: *With the extension. OBJ, we can save the object code in C programming. This is the major difference between the two codes.*

Compiler VS Interpreter VS Linker:

- *Compiler:*
  - *Compiler reads the entire source code and converts it to binary (machine language).*
  - *Program Speed is fast.*
  - *One time execution*
- *Interpreter:*
  - *Interpreter read the program one line at time and execute that line.*
  - *Program speed is slow*
  - *Interpretation occurs at every line of the program **
- *Linker:*
  - *It's a program which links a compiled program to the necessary library routines, to make it an executable program.*

Preprocessor Directives

The C preprocessor is a macro processor that is used automatically by the C compiler to transform your program before actual compilation (Preprocessor directives are executed before compilation.). It is called a macro processor because it allows you to define macros, which are brief abbreviations for longer constructs.

A macro is a segment of code which is replaced by the value of macro. Macro is defined by #define directive.

1. #include:

The #include preprocessor directive is used to paste code of given file into current file. It is used include system-defined and user-defined header files. If the included file is not found, the compiler renders error. It has three variants:

#include <file>

This variant is used for system header files. It searches for a file named file in a list of directories specified by you, then in a standard list of system directories.

#include "file"

This variant is used for header files of your own program. It searches for a file named file first in the current directory, then in the same directories used for system header files. The current directory is the directory of the current input file

#include anything else

This variant is called a computed #include. Any #include directive whose argument does not fit the above two forms is a computed include.

2. #Define:

Define used to define a macro.There are two types of macros:

1. Object-like Macros

The object-like macro is an identifier that is replaced by value. It is widely used to represent numeric constants.#define PI 3.1415

2. Function-like Macros

The function-like macro looks like a function call. For example:

#define MIN(a,b) ((a)<(b)?(a):(b))

3. #undef:

To undefine a macro means to cancel its definition. This is done with the #undef directive.

4. #ifdef:

The #ifdef preprocessor directive checks if macro is defined by #define. If yes, it executes the code.

5. #ifndef:

The #ifndef preprocessor directive checks if macro is not defined by #define. If yes, it executes the code.

6. #if:

The #if the preprocessor directive evaluates the expression or condition. If the condition is true, it executes the code.

7. #else:

The #else preprocessor directive evaluates the expression or condition if condition of #if is false. It can be used with #if, #elif, #ifdef and #ifndef directives.

8. #error:

The #error preprocessor directive indicates error. The compiler gives fatal error if #error directive is found and skips further compilation process.

9. #pragma:

The #pragma preprocessor directive is used to provide additional information to the compiler. The #pragma directive is used by the compiler to offer machine or operating-system feature. Different compilers can provide different usage of #pragma directive.

Example:

#pragma Startup: Needed to run before program Startup.

#pragma Exit: Used just before program Exit

10. #Typedef:

Used to give defined data type a new name. for example: typedef unsigned long ulong.

11. ## : pasting operator:

Used to concat two tokens.

12. #line #file:

Used for debugging: “line” for number of line, “file” for name of file.

⇒ Advantage of C preprocessor:

- *Easier to read and modify.*
- *C-code is more transportable between machine architecture.*

Compilation steps:

1. Preprocessing:

First phase through which Source code is passed.This phase include:● Removal Comment.

- Expansion of Macros.
- Expansion of include file.

Input: file.c [gcc -E file.c] Output: file.i

2. Compilation:

In this Step the preprocessed Code is translated to Assembly instruction specific to the target processor architecture:

- gcc -S file.c ⇒ Output: file.s

3. Assembly:

An assembler is used to translate the assembly instruction to object code.

- gcc -c file.c

⇒ Output: file.o the contain of the object file can be inspected using hexdump.

4. Linking:

Linked adds some extra code which is required when the program start and end, like code for setting up the environment such as passing command line arguments.

This task can be verified using “size file.o” ⇒ the size will be increased .

- gcc -o file file.c ⇒ output: executable file.

Storage Class:

Auto storage:

All variables declared inside a function without any storage classes keyword is considered as auto storage class by default. In auto storage class a memory is allocated automatically to the variables when the function is called and deallocates automatically when the function exits.Variables under auto storage classes are local to the block or function. So, it is also called as local variables.The keyword auto is rarely used because all uninitialized variables are said to have auto storage classes.

They can be accessed only within the function they have been declared, but can be accessed also outside their scope using the concept of pointer, by default they are assigned a garbage value.

The visibility of a variable is the function or block in which variable is declared.

Static storage:

A static variable is a variable that tells the compiler to retain the value until the program terminates.They are created once when the function is called, even though the function gets repeated it retains the same value and exists until the program terminates.The default value of static variable is zero(0).The keyword for a variable to declared under static the storage class is static. The visibility of a variable is the function or block in which variable is declared.

The memory for auto variables is allocated in heap and memory for static variables is allocated in the data segment. So, The static variable remains in memory until the program is terminated. Therefore, static variable is initialized only once and value of the static variable is maintained during the function call.

Static global variable: Make the variable visible within only the c file and should not define this variable in the header file.

Static function:

By default, function are extern so they will be visible from other file, if functions are declared as Static , they will be invisible from the outer file.It’s used when we want to restrict access to function or reuse the same function name in the file.

Register storage:

Register variables are similar to auto variables. The only difference is register variables are stored in CPU register instead of memory. Register variables are faster than normal variables. Mostly programmers uses register to store frequently used variables. Programmers can store only few variables in the CPU register because size of register is less. The keyword for a variable to be declared under register storage class is register.

External storage:

Extern variable is a programmer's shorthand to represent external variable.Extern variables are also known as global variables because extern variables are declared above the main function.So, the variables can be accessed by any function. We can also access extern variables of one file to another file. But make sure both files are in same folder. They are used to extend the visibility of variable, default value of Extern is zero, and lifetime till the end of the program execution.

Local/Global Scope :

Local: A variable declared inside a function or a block has its scope only in that function/block, and their lifetime equal to the lifetime of the function/Block.

Global: *a variable declared outside function has global scope which can be accessed by any function or block.They are unaffected by scope and always available until the end of program.*

Note:

If a function has local/static variable with same name as global variable, local/static value is used.

If a function does not find a required variable in local scope, it searches the variable in global scope. If it does not find even in global scope, it will throw an error.

NB:

If a variable is declared with register storage class and the register in CPU is not available, then register storage class will be convert to auto storage class if register is not available and it will give warning not error

Typedef:

- Defines a new type based on an existing type.

Const: It's a qualified, used in the declaration of variable to specify that's value will not be changed. We may change it's value using pointer.

Volatile:

Volatile tell the compiler that the value of the variable may changed at any time without any action being taken by the code. In practice only 3 type of variable could change:

- *Memory mapped peripheral register.*
- *Global variable modified by an interrupt service routine.*
- *Global variable accessed by multiple task.*

Volatile function:

Means that does not apply volatile to the function but to the return of the function may change at any time .

Storage Class	**Storage**	**Initial Value**	**Scope**	**Life**
Auto	Stack	Garbage	With in block	End of Block
Extern	Data Segment	Zero	Multiple File	Till end program
Static	Data Segment	Zero	With in block	Till end program
Register	CPU Register	Garbage	With in block	End of Block

Error Type while using Storage Classes:

```
extern int i;
printf("%d\n", i);
```

⇒ Run time error: Memory will not be allocated to variable i unless we initialized.

```
static int i = 25;
printf("%d\n", i);
static int i = 20
```

⇒ Compilation error: Redeclaration of variable.

```
int a;
printf("%d \n", a);
```

⇒ Garbage value: as a is a local variable so will be initialized with garbage value.

```
static int i;
printf("%d\n", i);
```

⇒ 0: Static variable initial value is 0 either it's local or global.

Memory Segmentation:

1. Text Segment (Code Segment):

- Is one of the sections of a program in an object file which contains executable instruction. Text segment may be placed below the heaperthe stack in order to prevent heap and stack overflow from overwriting.

- Text segment is often read only to prevent the program from modifying its instruction.

2. Initialized data segment (data segment):

- It's a portion of virtual @ space of program, which contains the global variable and static variable that are initialized by the programmer.Data segment is not read only since the variable can be altered at run time

3. Uninitialized data segment:

- Called also "bss: block started by symbol", data in this segment is initialised by the kernel to arithmetic 0, before the program starts execution.

- Uninitialized data start at the end of the data segment and contain all global variable and static variable that are initialised to zero or do not have explicit initialization in source code.

4. Stack:

- Stack areas contain the program stack , a LIFO Structure, located in the higher part of memory. A stack pointer register tracks the top of the stack, it's adjusted each time a value is pushed on the stack.

5. Heap:

- It's a segment where dynamic memory allocation usually takes place. The heap is managed by malloc, calloc, realloc and free.The heap area is shared by all shared libraries and dynamically loaded modules in the process.

Memory management:

1. Malloc:

Memory allocation used to dynamically allocate a single large block of memory with the specified size. It return a pointer of type void which can be cast into a pointer of any form.

**ptr = (int*) malloc(100 * sizeof(int))*

2. Calloc:

Dynamic allocation with the specified number of block of memory of the specified type.It initialize each block with zero

**ptr = (float*) calloc (25, Sizeof(float))*

⇒ For Malloc & Calloc: if the space is insufficient, allocation fails and return a NULL pointer;

3. Free:

Used to dynamically de-allocated the memory.It helps to reduce wastage of memory by freeing it.

4. Re-alloc:

Used to change the memory allocation of previously allocated memory.

5. Memory leak:

It occurs when programmers create a memory in heap and forget to delete it.

6. Static memory allocation:

Memory is allocated at compile time and can't be increased while executing the program.It's used basically in the array. Lifetime of the variable in static memory is the lifetime of a programme.Static memory is implemented using stack or heap.It's faster than dynamic memory.

7. Dynamic memory allocation:

Memory is allocated at runtime and can be increased while executing the program. Malloc & Calloc is required to allocate the memory at the runtime.The dynamic memory is implemented using the data segment.

Pointer:

A pointer is a variable that refers to the @ of value, when a variable is declared , the systems allocate some memory to The variable pointer makes the code optimized and the performance fast.

- *Usage of pointer:*
  - *- Accessing arrays elements.*
  - *- Dynamic memory Allocation.*
  - *- Call by reference.*
  - *- Construction of different data structures.*

What are the differences between references and pointers?

Both references and pointers can be used to change local variables of one function inside another function. Both of them can also be used to save copying of big objects when passed as arguments to functions or returned from functions, to get efficiency gain.

Despite above similarities, there are following differences between references and pointers:

References are less powerful than pointers:

- *Once a reference is created, it cannot be later made to reference another object; it cannot be reseated. This is often done with pointers.*
- *References cannot be NULL. Pointers are often made NULL to indicate that they are not pointing to any valid thing.*
- *A reference must be initialized when declared. There is no such restriction with pointers*

References are safer and easier to use:

- *Safer: Since references must be initialized, wild references like wild pointers are unlikely to exist. It is still possible to have references that don't refer to a valid location*

● *Easier to use: References don't need a dereferencing operator to access the value. They can be used like normal variables. '&' operator is needed only at the time of declaration. Also, members of an object reference can be accessed with dot operator ('.'), unlike pointers where the arrow operator (->) is needed to access members.*

NULL Pointer:

A pointer that doesn't refer to any @ of value, but NULL is known as Null Pointer. When we assign "0" to a pointer of any type then it becomes a NULL Pointer.

Eg:

```
#define NULL 0
int *ptr=NULL;
```

If you try to dereference a NULL pointer it will result in runtime error such as segmentation fault.

Used of NULL pointer:

Pointer does not contain valid @ and should not be dereferenced .

Pointer is available to be assigned a new/valid @

Far Pointer:

A pointer which can access all the 16 segments of RAM. A far pointer is 32 bit pointer that obtain information outside the memory in a given section.

Near pointer:

Near pointer is used to store 16 bit addresses within current segments on a 16 bit machine. The limitation is that we can only access 64kb of data at a time.

Huge pointer

a huge pointer is also typically 32 bit and can access outside segments. In case of far pointers, a segment is fixed. In far pointer, the segment part cannot be modified, but in Huge it can be

Dangling pointer:

If a pointer is pointing any memory location, in the meanwhile another pointer delete this memory while the first is still point to that memory location then the first pointer will be known as a dangling pointer.

Pointer to pointer:

One pointer refers to the @ of another pointer. Generally, the pointer contains the @ of value, so the pointer to pointer contains the @ of the first pointer.

Void pointer:

When we don't know what type of memory @ the pointer variable is going to hold, then we declare it as a void pointer.
⇒ Dereference a void pointer without cast ⇒ Compilation error .

Constante Pointer:

A pointer which is not allowed to be altered to hold another @ after it is holding one. So it can't change the @ of the variable that's pointing to.

Aliasing pointer:

Aliasing refers to the situation where the same memory location can be accessed using different names. So pointer alias when they point to the same @.

Wild pointer:

Uninitialized pointer are know as wild pointer.these are point to some arbitrary memory location and can cause bad behaviours or program crashes.

Struct / Union / Enum:

1. Struct:

It is a user defined data type, that can be used to group items of different types into a single type.

2. Enum:

User defined data type used to assign names to integrals constants, the name make a program easy to read and maintain;

3. Union:

Union is a user defined data type in which all members share the same memory location.

⇒ Difference between Structure and Union:

- In Struct each member get separate space in memory, in total memory required to store a structure variable is equal to the sum of all members.

- In Union total memory space allocated is equal to the member with the largest size.

Function:

1. Pointer function:

A pointer that hold the reference of the function.Function pointer can be useful when you want to create a callback mechanism, and need to pass @ of a function to another function.

2. Callback function:

The callback is any executable code that is passed as an argument to other code, that is expected to call back or to execute the argument at a given time. If the reference of function is passed to another function argument for calling, then it is called the callback function.

3. Static function:

A Static function is a function that has a scope that is limited to its object file.this mean the function static is visible in its object file.

4. Recursive function:

It's a function which calls itself and includes an exit condition in order to finish the recursive call. Recursive works by "stacking" calls until the exit condition is true.

5. Inline function:

A function call is directly replaced by an actual program code.It does not jump to any block because all the operation are performed inside the inline function.

Macros VS Inline:

Macros are more flexible, they can expand other macros where inline functions don't necessarily do this.
The return keyword can not be used in macros to return value as in the case of function.
Macros are generally used for code reuse.
Inline functions can provide scope for variables, Preprocessor macros can only do this in code block {...}.
Inline function is expanded by the compiler whereas the macros are expanded by the preprocessor .
Debugging code is easy in case of inline function.

Extern and Static Function:

By default any function is extern, these functions can be used in any other source file of the same function .
Static function, these functions can't be used in other files of the same project.

File Handling:

File pointer is a pointer which is used to handle and keep track of the file being accessed. A new data type called "File" is used to declare a file pointer.

Embedded C

Embedded C is extension of standard C, is mainly used for desktop application where it has access to RAM, ROM, OS, etc. It's used for microcontroller based application which has limited amount of RAM, ROM, I/O.

- *Program Optimization technique in Embedded Programming:*
- *Speed optimization.*
- *Memory Optimization.*
- *Power Optimization.*

NB:

- Static variable declared in file1.c, how to access its value in file2.c ??

- *Using Pointer or Function.*

- Can a variable be declared as extern static ?

No, variables have only one storage class Else ⇒ Compilation error.

- Auto variable, used for local variables and stored in Sack.

Volatile:

Tell the compiler that the variable of the value may change to any time without any action being taken by the code .
Used:
Situation where a variable or pointer can be modified by hardware peripheral event or ISR. If the volatile keyword is not used then the com

Volatile function:

*⇒ void (*volatile fptr)(void*)*
- We can use both volatile and const: volatile int const HWStatusReg
Hardware Status Register are best example as these register are updated based on HW event but are read only for program.
Here const indicates that HWstatusReg will not be modified by program.
Volatile indicates it can be modified by external HW events.

Size of pointer:

Size of pointer depends on several factors such as CPU architecture, OS, and compiler configuration. Generally size of pointer is equal to data bit it can process in one shot eg 4 bit for 32 bit Microprocessor.

Which is the faster Macro OR function and Why ??

Macro are faster, when function is called , before executing of first instruction of the function, there are several step as: storing current register context in stack, storing return @, jump to destination @ and allocate /initialize new stack frame for the function to be executed. All these step take time but when use macros, no need for these ste

For decremented loop VS For Increment loop:

Decremented loop is best because processor has to compare value of index with zero and that it is an optimization opération.

Cause of Segmentation fault:

It's a runtime error which may occurs when:

- Usage of the dereference pointer (point which may not have a valid @/memory location)

- try to access memory areas which read only.

- free a memory which is already freed.

- Segmentation fault is the reason to generate stack overflow

Stack Overflow error:

- Program tries to access the memory beyond its available max limit.

- If pointer exceeds the stack limitation

⇒ When this error occurs the program terminates and does not execute further instruction.

Can pointers be Volatile?

*Yes, if we see the declaration : volatile int *p, it means that itself is not volatile and points to an integer that is volatile. this tells the compiler that p is pointing to an integer and the value of that integer may change unexpectedly.*

How to protect char pointers from modification?

*Constante char pointer (const char *) prevent the unnecessary modification of the pointer @ in the string.*

Macros that return largest argument

```
#define Max(A,B) (A>B?A:B)
```

How can you avoid including a header file multiple times?

```
#ifndef _MY_HDR_H
#define _MY_HDR_H
// code of the file
// code of the file
#end if
```