PAGE
DATE / /

EY_Meghalchar_49398

Sunbeam Institute of Information Technology, Pune and Karad
Embedded C Programming Rapid Fire Sheet

Course : eDESD Batch : May 2021
Module Name : Embedded C Programming.

Rapid Fire Questions

Question 1

Explain compilation and execution flow of C program?

⇛

C Program :- A C program is a set of functions, data type definitions and variable declarations contained in a set of files.

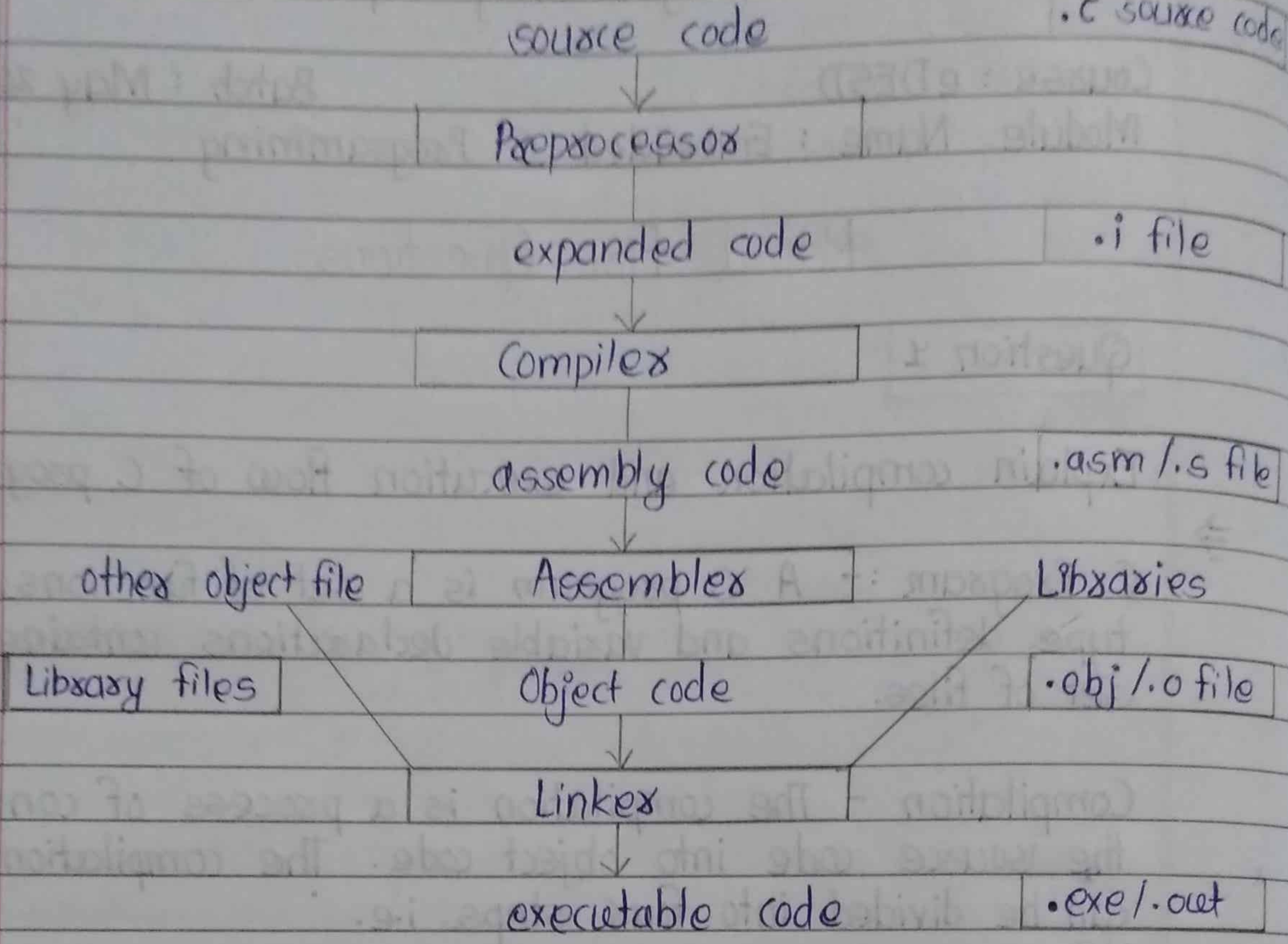
Compilation - The compilation is a process of converting the source code into object code. The compilation process can be divided into four steps, i.e.
1) Pre-processing
2) Compiling
3) Assembling
4) Linking.

Execution flow of C Program :
1st Create C Program :
1) Project creation in VS Code.
2) Project name & Program name.
3) Program contents
 - comments, header file, entry point function ((main())
4) Program execution steps.

PAGE:
DATE: / /

E1_Megha Lohar_49398

Program Execution Phases :-



1) C program (source code) is sent to preprocessor first.
2) Expanded source code is sent to compiler which compiles the code and converts it into assembly code.
3) The assembly code is sent to assembles the code and converts it into object code.
Now a simple .obj file is generated.

E4_Megha Lohar_49398

1) C program (source code) is sent to preprocessor first. The preprocessor is responsible to convert preprocessor directives into their respective values. The preprocessor generates an expanded source code.

2) Expanded source code is sent to compiler which compiles the code and converts it into assembly code.

3) The assembly code is sent to assembler which assembles the code and converts it into object code. Now a simple .obj file is generated.

4) The object code is sent to linker which links it to the library such as header files. Then it is converted into executable code. A simple .exe file is generated.

5) The executable code is sent to loader which loads it into memory and then it is executed. After execution, output is sent to console.

Question 2

What is the difference between declaration and definition? Explain in context of functions and variables.

The difference between declaration and definition of functions & variables is as follows:

PAGE:
DATE: / /

E1 Megha Lohar - 49398

Declaration	Definition.
A variable or a function can be declared any number of times	A variable or a function can be defined only once.
Memory will not be allocated during declaration.	Memory will be allocated
int f(int); The above is a function declaration. This declaration is just for informing the compiler that a function named f with return type and argument as int will be used in the function.	int f (int a) { return a; } The system allocates memory by seeing the above function definition.

Question 3

Write a function to print fibonacci series using recursion?

E1_Megha Lohar_ 49398

```
#include <stdio.h>
# include <stdlib.h>

int fib (int number)
  {
    if (number ==0) return 0;
    if (number == 1) return 1;
    return fib (number -1) + fib (number - 2);
  }
 int main (void)
  {
   int count;
   for (int count = 0; count <= 13; ++count)
      {
        int value = fib (count);
        printf ("%5d", value);
      }
      printf ("\n");
      return 0;
    }
```

o/p

```
0  1  1  2  3  5  8  13  21  34  55  89  144
233
```

PAGE:
DATE: / /

E1_Megha lohar _ 49398

Question 4.

What is significance of storage class ? Which storage classes are in C ? Explain static in context of functions and variables.

- Storage Classes are used to describe the features of a variable / function. These features basically include the scope. visibility and life-time which help us to trace the existence of a particular variable during the runtime of a program.

- There are four different storage classes in a C program -
  auto
  register
  static
  extern

- The static variables and functions
  Basically, when static variables are declared. they create only a single copy of them and are also termed as Class Variables.

  Static functions in C basically restrict the scope of the method to the corresponding file. These functions can also be called without ~~t~~ having the object initialized.

EI_Megha Lohar_49398

Question 5

What is pointer ? How and when const keyword is used with pointer ?

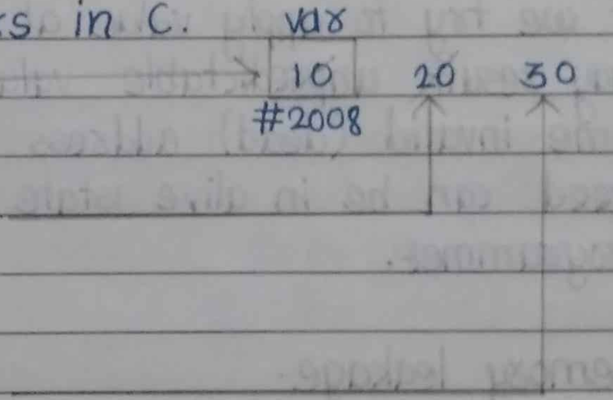
Pointer
- Pointer is known as derived data type.
- A pointer is a variable that stores address of a memory location.
- Using pointer we can create a variable to store address.
- We should always store a valid / in alive state address in pointer.

e.g. How pointer works in C.

```
    int var = 10;

    int *ptr = & var;
        *ptr = 20;

    int **ptr = & ptr;
        **ptr = 30;
```



- The qualifier const~~an~~ can be applied to the declaration of any variable to specify that its value will not be changed (Which depends upon where const variables are stored, we may change the value of const variable by using pointer).

- The result is implementation - defined if an attempt is made to change a const.

FY_Megha Ichar - 49398

PAGE
DATE / /

Question 7

What is the difference between passing argument by value and by address? Which type of arguments cannot be passed by value?

Pass by value
Pass by value means you are making a copy in memory of the actual parameter's value that is passed in. a copy of the contents of the actual parameter. Use pass by value when you are only "using" the parameter for some computation, not changing it for the client program.

Pass by address
Pass by address also called as pass by reference. A copy of the address of the actual parameter is stored. Use pass by reference when you are changing the parameter passed in by the client program.

Type of arguments cannot be passed by value :-
A Variant argument will accept a value of any built-in data type; and any list, array, or object. A variant argument will not accept a value of a user-defined type.
Lists, arrays, objects, and user-defined types cannot, and therefore should not, be passed by value.

PAGE:
DATE: / /

E1-Megha Lohar- 49398

Question 8

Which are function calling conventions in C ? What is its significance ?

- Calling conventions specify how arguments are passed to a function, how return values are passed back out of a function, how the function is called, and how the function manages the stack and its stack frame.
- In short, the calling convention specifies how a function call in C into assembly language.

Calling Conventions.

	Calling conventions	Argument push order	Stack cleanup
1)	cdecl	right to left	calling function
2)	pascal	left to right	called funⁿ
3)	stdcall	right to left	called funⁿ

Question 9

What do you mean by structure member alignment, padding and data packing?

PAGE:
DATE: / /

E4 - Megha Lohar - 49398

Structure Member Alignment

- Data structure alignment is the way data is arranged and accessed in computer memory.
- It consists of two separate but related issues: data alignment and data structure padding.
- When a modern computer reads from or writes to a memory address. it will do this in word sized chunks. (e.g. 4 byte chunks on a 32- bit system). or larger.
- Data alignment means putting the data at a memory address equal to some multiple of a word size, which increases the system's performance due to the way the CPU handles memory.
- To align the data, it may be necessary to insert some meaningless bytes between the end of the last data structure and the start of the next. which is data structure padding.
- Packing or data packing.
  Packing. on the other hand prevents compiler from doing padding means remove the unallocated space allocated by structure.

PAGE:
DATE: / /

E1 - Megha Lohar - 49398

Question 10

What is Array ? Which are limitations of it ? Why array index begins with zero ?

Array
- Array is collection of similar data elements in contiguous memory locations.
- It is a collection of similar type elements.

Limitations
The limitations of an array are explained here below.

I) An array which is formed will be homogeneous. That is, in an integer array only integer values can be stored. while in a float array only floating value and character array can have only characters. Thus, no array can have values of two data types.

II) While declaring an array. passing size of an array is compulsory. and the size must be a constant. Thus, there is either shortage or wastage of memory.

III) Shifting is required for insertion or deletion of elements in an array.

IV) An array doesn't check boundaries : In C language, we cannot check, if the values entered in an array are exceeding the size of that array or not.

PAGE:
DATE: / /

E1- Megha Lohar - 49398

Array index begins with zero.
An array arr [i] is interpreted as *(arr+i). Here, arr denotes the address of the first array element or the 0 index element. So *(arr+i) means the element at i distance from the first element of the array.

Question 11

Write a function to simulate string reverse and check whether string is palindrome or not ?.

Given a string, write a c function to check if it is palindrome or not.
A string is said to be palindrome if reverse of the string is same as string.
For example, "abba" is palindrome, but "abbc" is not palindrome.

```
# include <stdio.h>
# include <string.h>

 // A function to check if a string str is palindrome.
void isPalindrome (char str [])

 // start from leftmost and rightmost corners of str
int l = 0;
int h = strlen (str) - 1;

 // Keep comparing characters while they are same
```

```
while (h>1)
  {
     if (str [l++] ! = str [h--])
     {
       printf ("%s is Not Palindrome", str);
       return;
      }
  }
 printf ("%s is palindrome", str);
}

// Driver program to test above funⁿ.
int main()
  {
     isPalindrome ("abba");
     isPalindrome ("abbccbba");
     isPalindrome ("geeks");
     return 0;
  }
```

Output :-

```
   abba is palindrome.
   abbccbba is palindrome
   geeks is Not Palindrome.
```

Question 12

What is the difference between macro and function?

PAGE:
DATE: / /

E1 - Megha Lohar - 49398

	Macro	Function.
1)	Macro is Preprocessed	Function is Compiled
2)	No Type Checking.	Type Checking is Done.
3)	Code length Increases	Code length remains same.
4)	Speed of Execution is Faster.	Speed of Execution is Slower.
5)	Before Compilation macro name is replaced by macro value.	During function call, Transfer of control takes place.
6)	Useful where small code appears many time	Useful where large code appears many time.
7)	Generally Macros do not extend beyond one line	Function can be any number of lines.
8)	Macro does not check Compile Errors.	Function Checks Compile Errors.

Question 13.

What is the difference between structure and union?

PAGE:
DATE: / /

EY - Megha Lohar - 49398

Structure	Union
1) Access Member We can access all the members of structure at anytime.	1) Access Member Only one member of union can be accessed at anytime.
2) Memory Allocation Memory is allocated for all variables.	2) Memory Allocation. Allocates memory for variable which variable require more memory.
3) Initialization All members of structure can be initialized.	3) Initialization Only the first member of a union can be initialized.
4) Keyword 'struct' keyword is used to declare structure.	4) Keyword. 'union' keyword is used to declare union.

5) Syntax
struct struct-name
{
structure element 1;
structure element 2;
------ 	
----.
structure element n;
} ~~int~~ struct_var_nm;
	5) Syntax.
	union union-name
	{
	union element 1;
	union element 2;
	----- 	
	---- 	
	union element n;
	} union_var_nm;

PAGE:
DATE: / /

FY_Megha Lohar - 49398

Question 14.

What is function pointer ? With example, explain how to declare and define function pointer ?

Function Pointer
- Function Pointers point to code like normal pointers.
- In Functions Pointers, function's name can be used to get function's address.
- A function can also be passed as an arguments and can be returned from a function.

Declaration
function_return_type (*Pointer_name) (function argument list)

Example
```
# include <stdio.h>
int subtraction (int a, int b)
  {
    return a-b;
  }
  int main ()
  {
   int (*fp) (int, int) = subtraction;
   // calling function using function pointer
   int result = fp (5,4);
   printf ("Using function pointer we get the result: %d",
           result);
   return 0;
  }
```