C Programming Assignment-02

By Tajim Md.Niamatullah Akhund ;   IIT (5ᵗʰbatch) ;   JU(43batch) ;   Roll-1916

1. What are the storage classes a variable may have? What is the scope of a variable in each storage class? Give example.

Answer: The storage classes of a variable are given below:

  1. Automatic   2. External   3. Static   &   4. Register

 1.Automatic: scope of the variable confined to that function.

 Example:
 #include <stdio.h>
 void func_1()
 {auto int x=500;
 x++;
 …….}

2.Static: scope of the variable confined to that function.

Example:
#include<stdio.h>
{static int x=500;
x++;
……..}

3.Register: scope of the variable confined to that function.

Example: Iterative variable for loop or array index variable.
…..
register int i=0;
for(i=0; i<n; i++)
{…..array [i]=…..}…

4.External: scope of the variable confined to that function.

Example:
#include<stdio.h>
int x=100;
void func_1()
{x++;
……}

2. What is an Array variable? How can we declare and initialize an array variable in C?

Answer: An array is a variable containing multiple values. There is no maximum limit to the size of an array, nor any requirement that number variable be indexed or assigned continuously. Arrays are zero based: the first element is indexed with the number zero.
Declaring and initializing an array variable in C:

storage_class data_type array[expression];
  ➢ static int myArray[25];
Initializing:
  ❖ static int myArray[5]={100,24,50,70,200};
  ❖ static int myArray[]={10, 25, 20}; // array with 3 elements
  ❖ int myArray[5]={20,10}; //other values will be 0

  ❖ Good programming practice- declare arrays as global or static
    ➢ Automatically initialized to 0 or null
    ➢ Easy to use among different functions
Array_Name= address of 1st element of the array (&myArray[0])

3. What is the difference between the following two arrays:
char myName[5]="RAHIM" , yourName [ ]="RAHIM";

Answer: Difference between char myName[5]="RAHIM", myName[ ]="RAHIM":
First statement has 5 elements,
Second statement has 6 elements.
myName[0]= 'R'
myName[1]= 'A'
myName[2]= 'H'
myName[3]= 'I'
myName[4]= 'M'
extra element in second statement to indicate end of string
myName[5]= '\0' //null character.

4. How can we pass an array to a function? What type of passing it is?

Answer:

…
```
int main()
{    int  T;

float Avg, myArray[20];
....
Avg=average(T, myArray);
....
return 0;
}
float   average(int T, float myArray[])
{   float A, sum=0.0;
....
A= sum/T;
return A;
}
```
Any change in the array at function also will be affected in the main function –which is pass by reference.

5. What is pointer? Show how pointer can be used to pass arguments by reference to a function.

Answer: Pointer: A pointer is a variable whose value is the address of another variable, direct address of the memory location.

```
#include <stdio.h>
void  func( int *m )
{
*m=200;
return ;
}
int  main( )
{ int  x, *p;
x=3000;
p=&x;
func(p);
return 0;
}
```

x = 3000	0x9ff
…	…
m=0x9ff	0x8f1
…	…
p=0x9ff	0x7e7
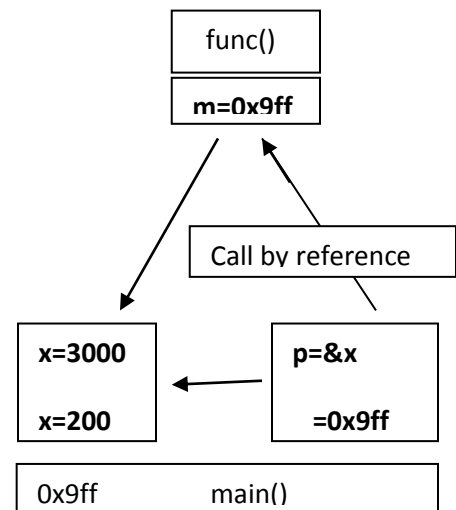
func()

m=0x9ff

Call by reference

x=3000

x=200

p=&x

=0x9ff

0x9ff   main()

6. What are the differences between call by value and call by reference while passing arguments in a function. Give example with explanation.

Answer: Two ways of passing argument to function in C language:
1. Call by value &
2. Call by reference.

1.Example of call by value:
```
#include<stdio.h>
void interchange (int number 1, int number 2)
{    int temp;
temp=number 1;
number1=number2;
number2=temp;
}
int main()
{    int number1= 50, number2=70;
interchange(number1, number2);
printf("\n number1 : %d", number1);
printf("\n number2 : %d", number2);
return 0;}
```
Output:
1. Number1: 50
2. Number2: 70

Explanation:
1. While passing parameters using call by value, xerox copy of original parameter is created, and passed to the called function.
2. Any update made inside method will not affect the original value of variable in calling function.

2.Example of call by reference:
```
#include<stdio.h>
void interchange (int *number1, int *number2)
{    int temp;
temp= *number 1;
*number1=*number2;
*number2=temp;
}
int main()
{    int number1= 50, number2=70;
interchange(number1, number2);
printf("\n number1 : %d", number1);
printf("\n number2 : %d", number2);
```

```
return 0;
}
```
Output:
1. Number1: 70
2. Number2: 50.

Explanation:
1. While passing parameters using call by reference, we are passing the actual address of the variable to the called function.
2. Any updates made inside the called function will modify the original copy since we are directly modifying the context of the exact memory location.

7. How pointer variable can be used instead of array? Give example, What is the benefit of doing so?

Answer: Pointer variable can be used instead of array by specifying the array's name without an index. For example, the following program fragment passes the address of I to func 1();
```
Int main (void)
{int i[10];
Func 1[i];
/*…..*/}
```
If a function receives a pointer to a single dimension array, we can declare it's formal parameter as a pointer. For example, to receive i, a function called func1() can be declared as
```
void func1(int *x)/*pointer*/
{/*…..*/}
```
Benefit: The benefit of doing so is the length of the array does not matter as far as the function is connected.

8. Briefly describe what types of operation can be done with a pointer variable.

Answer: Pointer Usages:-
1. Passing data items to back and forth between function and calling part(call by reference)
2. Alternate way to access to the element of an array.

9. What is difference between an array and a structure?

Answer: Difference between array & structure:

Structure	Array
Structure elements are of different data type.	Array elements are homogeneous.
Structure is not a pointer.	Array is a pointer to the first element of it.
For accessing the number of a structure.	Accessing elements of an array.

10. Write the syntax of a structure. What is the difference between a union and a structure?

Answer: Syntax of a structure:

defining a structure *Outside* of a function is *global* and *Inside* of a function is *local*.

struct   tag_name
{ data_type   member_1;
data_type   member_2;

// more if needed
} ;

Declaring structure type variables
struct   tag_name   X, Y;

// Members can be accessed as normal variable using period (.) symbol
X.member_1 = …
X.member_2 = …
Y.member_1 = …
Y.member_2 =…

Difference between structure and union:

Structure	Union
The keyword struct is used to define a structure.	The keyword union is used to define a union.
Each memory with a structure is assigned unique storage area of location.	Memory allocated shared by individual members of union.
Altering the value of a member will not affect other members of the structure.	Altering the value of any of the member will alter other member values.
Individual member can be accessed at a time.	Only one member can be accessed at a time.
Several members of a structure can initialize at once.	Only the first member of a union can be initialized.

11. What is difference between "." and "->" symbols used to refer a member variable of a structure?

Answer: The difference between "." And "->" symbols used to refer a member variable is given below through an example:
…
struct   Books   a;
struct   Books   *p;
p= &a;
…
gets(a.title);
gets(a.author);
scanf("%f",   &a.price);
scanf("%d",   &p->book_id);
…
printf("Book title is %s",   a.title);
printf("Book title is %s",   ->title);

12. With example show how can we use user defined data type in C.

Answer: We can use user defined data types to manage their code easily. User defined data types are equivalent to existing data types (int, float etc.). If it is big program and if in future all types of marks have to be changed to float, it would be very difficult. This problem can be solved by defining a new data type (example M) & declaring all the marks as M type.
Example:
```
#include<stdio.h>
#include<string.h>
typedef struct Bppks
{        char title[50];
         char author[50];
         char subject[100];
         int book_id;
}Book;
int main()
{Book book;
strcpy(book.title, "C Programming");
strcpy(book.author, "Unmatched Tajim");

strcpy(book.subject, "Ennovative Circuits");
```

strcpy(book.book_id=3456798_);
printf("Book title : %s\n", book.title);
printf("Book author : %s\n", book.author);
printf("Book subject : %s\n", book.subject);
printf("Book book_id : %d\n", book.book);
return 0;
}

13. Write the basic syntax of using a file for input/output. Mention the meaning of different file type specifications used in C file handling.

Answer: The basic syntax of using a file for input/output:
Syntax of fopen():
fp=fopen(fileName, fileType);
here,   fp= file pointer
fileName= a string (example: "input.txt")
fileType=a string (example: "w+")
If any error occurs while opening a file, fp=0.

Each opened file must be closed at the end of the program (before return statement)

fclose(fp) ;   //here, fp=file pointer.

The meanimg of different file type specification:
        For new file (old file will be deleted)
            ❖ "w"   = write only.
            ❖ "w+"   = read and write
        Existing old file (if no file exists, there will be an error. FilePointer=0)
            ❖ "r"   = read only
            ❖ "r+"   = read and write
        Appending data in existing old file (if no file exists, new file will be created)
            ❖ "a"   = append only (add data at the end of file)
            ❖ "a+"   = read and append
            ❖ File types may be written as "wb", "wb+", "rb", "rb+" etc. The character 'b' has no effect, but is allowed for ISO C standard conformance

14. What are the syntax and function of following functions:
fopen(), fclose(), fscanf(), fprintf(), fwrite(), fread(), fputc(), fgetc(), fputs(), fgets

Answer:

Syntax	Function
fopen(s1, s2)	Open a file named sl of type s2. Return a pointer to the file.
fclose(f)	Close file f. Return 0 if file is successfully closed.
fscanf(f, …)	Enter data items from file **f**
fprintf(f, …)	Send data items to file **f**
fwrite(s,i1, i2, f)	Send **i2** data items, each of size **il** bytes, from string s to file f
fread(s, i1, i2, f)	Enter **i2** data items, each of size **il** bytes, from file f to string s
fputc(c, f)	Send a single character to file f
fputs(s, f)	Send string **s** to file f
fgetc(f)	Enter a single character from file f
fgets(s, I, f)	Enter string **s**, containing icharacters, from file f