String

Strings are used to manipulate text such as words and sentences. They are also called charcter array. A string constane is a 1-dimensional array terminated by a null charcter(\0). The ASCII value is '\0' is 0.

Eg: char name[]={'h','a','p','\0');

Declaration of String

Syntax :- char string_name[string size];

Initialization of string

Declare without size --- char name[]="news";

Declare with string size --- char name[30]="news";

Declare string of characters --- char name[]={'n','a','m','e'};
char name[10]={'n','a','m','e'};

Reading and printing a line of text

```
#include<stdio.h>

void main()

{
char name[100];

printf("ENter the words");

gets(name);

printf("the words %s",name);

puts(name);

}
```

Output

Enter the word abcd

The word abcd

Abcd

To Read a line of Text

fgets() to read a line of text and puts() to display a line of string. While using scanf() the programme terminate scanning when a space is detected

```
#include<stdio.h>
int main()
{
char name[30];
printf("enter the name");
fgets(name,sixeof(name),stdin);
printf("Name: ");
puts(name);
return 0;
}
```

Output

Enter the name abcd

Name: abcd

Standard library in strings

The header files used in these functions are **string.h**

Strlen() - Used to calculate the lenth of the string

Strcpy() - used to copy the content of one string to another

Strcat() - combines to strings. In it passes the base address of the two strings. It takes two arguments I.e. two strings or charcter arrays and stores the resultant concatenated string in the first array.

Strcmp() - it compares to array and check for any miss match if tehre is any missmatch the difference between the ASCII value of that particular point is displayed.

Programme

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

```
void main()
{
   char str1[20]="abcd";
   char str2[20]="efgh";
   char str3[20]="abCd";
   char str4[20];
   int len,res;
   len = strlen(str1);
   printf("the length of str1 = %d \n",len);
   strcpy(str4,str1);
   printf("copied text \n");
   puts(str4);
   strcat(str4 ,str1);
   printf("combination \n");
   puts(str4);
   printf("comparisoned text %d",strcmp(str1,str3));
}
```

Output

```
the length of str1 = 4
copied text
abcd
combination
abcdabcd
comparisoned text 1
```

Functions

Modular programming -: subdividing a programme into separate sub-progamms. The function which is used always is named as the main().

Advantages

- Development can be divided it divides the programme into different section for easy developments
- Programming errors are easy to be detected
- Re-use of codes can be done
- Improves Manageability They are easy to test, implement or design
- Collaboration in which small teams can do smaller set of programming

Breaking programme into smaller section is called **structure programming.**

Every C programme have one mandatory function **main()** from where the execution of the programme starts

Types of Functions

- Standard library function
- User-Defined Functions

Standard Library Functions

These are built-in functions which are defined by the header files. They are stored in what is known as library.

Eg. Printf() is a function of #include<stdio.h>

Sqrt is function of math.h

User-Defined Functions

These functions are defined by the user and are created on the need of the user.

How user defined functions work.

#include<……h>

Void functionname()

{

.

.

}

Int main()

{

.

.

}

Functionnaame()

{

.

.

}

C programme begins with the function main and it continue its execution upon the oder of the function calls in the main(). It doen't matter how many functions are declared in the programm. When the main function ends the programme also ends.

RULES

```
#include<stdio.h>

Int addnumber(int a, int b);   //function protoype

Int main(

{

Int n1,n2,sum;

Printf("enter two numbers \n");

Scanf("%d %d", &n1,&n2);

Sum = addnumber(n1,n2);    //function call

Printf("sum = %d",sum);

Return 0;

}

Int addnumber(int a, int b)  // function definition

{

  Int result;

Result = a+b;

Return result;    // return statment

}
```

Function Prototype

It is a function declaration that specifies function name, argument and return type. It doen't contain the function body. It tells the compiler that the function may be called upon later. It is not needed if the user defined function is called before the main() function.

Syntax :- returntype function_name(type1 argument1,type2 argument2,…….)

Calling a function

Control of the programme is transferred to the user-defined function by calling it.

Syntax:- functionname(argumnt1,argument2,…..)

Function Definition

It contains the block of code to perform a task. When we call a function the control is transferred to the function definition

Syntax :- returntype functionname(type1 argument1, type2 argument2,….)

{

body

}

Passing arguments to a function

Argument refers to the variable passed to the function These arguments are called **actual parameters** of the function. The parameters a and b accepts the passed arguments in the function definition which are known as formal parameters

The both of the data type of the arguments should be the same. A function can also be called without passing arguments.

Return statement

It immediately transfers the control to the calling program. There can be any number of return statement not only that the function may not be present at the end of the programme and a function can return only one value at a time. The type value in both the function and function prototype must be same.

Syntax I:- return (expression);

Call by value and call by reference

Call by value:- It copies the value of an argument into the formal parameter of that function. In this this the change made to the value inside the function is not reflected in the actual value because they are stored in different memory locations.

Call by Reference :- the value of actual argument is passed to the formal argument but in here any change in the formal argument will reflect in the change of value at the actual argument also. Here *variable name(*a) is used to pass the address of the argument.

Recursion

A function is called recursive if the body of a function calls the same function. if else statement can be used where one branch makes the recursive call and the other doesn't.

Array as function parameters

One dimensional array :- passing array elements to a function is similar to passing variables to a function.

returntype variablename(arrayname[]) //in the case of 1-D array

Multi dimensional array :- In this case the name of array is treated to the function.

In C we cannot return type an array so we use print directly in the statemenet

<u>Important Questions</u>

1. **What are predefined functions?**

   These are the functions which already have a definition in header files(stdio.h,string.h,etc). These function include printf(),scanf(),etc.

2. **Difference between predefined and user defined functions?**

   Predefined functions are functions that are already declared and comes under header files like string.h,stdio.h,etc. The functions include printf(),gets()

   User-defined functions are function defined by the users to perform the desired user task. They are called by functionname().

3. **What is function declaration, function call and function definition?**

Function decleration

It is a function declaration that specifies function name, argument and return type. It doen't contain the function body. It tells the compiler that the function may be called upon later. It is not needed if the user defined function is called before the main() function.

Syntax :- returntype function_name(type1 argument1,type2 argument2,…….)

Function call

Control of the programme is transferred to the user-defined function by calling it.

Syntax:- functionname(argumnt1,argument2,…..)

Function Definition

It contains the block of code to perform a task. When we call a function the control is transferred to the function definition

Syntax :- returntype functionname(type1 argument1, type2 argument2,….)

{

body

}

4. **Write syntax for function declaration, function call and function definition?**

Function decertation

Syntax :- returntype function_name(type1 argument1,type2 argument2,…….)

Function call

Syntax:- functionname(argumnt1,argument2,…..)

Function definition

Syntax :- returntype functionname(type1 argument1, type2 argument2,….)

{

body

}

5. **Difference between call by reference and call by value?**

Call by value:- It copies the value of an argument into the formal parameter of that function. In this this the change made to the value inside the function is not reflected in the actual value because they are stored in different memory locations. Simply, the actual address is not shared

Call by Reference :- the value of actual argument is passed to the formal argument but in here any change in the formal argument will reflect in the change of value at the actual argument also. Here *variable name(*a) is used to pass the address of the argument. Simply the actual address is shred and change take place in original value scene in return multiple elements.

6. **Difference between actual parameters and formal parameters?**

Actual parameters are parameters as they appear in function calls.

Formal parameters are parameters as they appear in function declarations.

Eg: int main()

{

Int a,b;

Sum(a,b); // actual parameters

}

Int sum(int a,int b) //formal parameters

{

.

.

}

7. **What is the advantage of call by reference method?**

In call by reference different memory locations are used there by which reduces the size of the programme and actual change takes place in the original value and useful in the case of multiple value returning

8. **What is function prototype ?**

It is a function declaration that specifies function name, argument and return type. It doen't contain the function body. It tells the compiler that the function may be called upon later. It is not needed if the user defined function is called before the main() function.

Syntax :- returntype function_name(type1 argument1,type2 argument2,…….)

Github Link for all the programme :- https://github.com/Amarjith-c-k/different-type-of-functions

9. **Write a programme to check whether number is odd or even using function?**

Github link :- https://github.com/Amarjith-c-k/different-type-of-functions/blob/master/check%20odd%20or%20even.c

//number is odd or even

```
#include<stdio.h>
int check(int a);
int main()
{
   int a;
   printf("enter a number \n");
   scanf("%d",&a);
   check(a);
   return 0;
}
int check(int a)
{
   int s;
   s = a%2;
   if(s=0)
      printf("number is even %d",a);
   else
      printf("number is odd %d",a);
}
```

10. **Write a programme to check whether number is prime or not ?**

Github Link :- https://github.com/Amarjith-c-k/different-type-of-functions/blob/master/primef.c

```
//check prime or not
#include <stdio.h>
int checkPrimeNumber(int n);

int main()
{
   int n, flag;

   printf("Enter a positive integer: ");
   scanf("%d",&n);

   flag = checkPrimeNumber(n);

   if(flag == 1)
      printf("%d is not a prime number",n);
   else
      printf("%d is a prime number",n);

   return 0;
}

int checkPrimeNumber(int n)
```

```
{
   int i;

   for(i=2; i <= n/2; ++i)
   {
      if(n%i == 0)
         return 1;
   }

   return 0;
}
```

11. Write a programme to check whether a number is amstrong or not using function?

Githublink :- https://github.com/Amarjith-c-k/different-type-of-functions/blob/master/amstrongf.c

```
//amstrong or not
#include<math.h>
#include<stdio.h>
int am(int n);
int main()
{
 int n, flag;
   printf("Enter a positive integer: ");
   scanf("%d", &n);
   flag = am(n);
   if (flag == 1)
      printf("%d is an Armstrong number.", n);
   else
      printf("%d is not an Armstrong number.", n);
   return 0;
}

int am(int num) {
   int original, rem, n = 0, flag;
   double result = 0.0;
   original = num;
   while (original != 0) {
      original /= 10;
      ++n;
   }
   original = num;
   while (original != 0) {
      rem = original % 10;
      result += pow(rem, n);
      original /= 10;
   }
```

```
    if (result == num)
        flag = 1;
    else
        flag = 0;
    return flag;
}
```

12. Write a programme for finding average of three numbers using

a. Function without argument without return statement

Github link :-
https://github.com/Amarjith-c-k/different-type-of-functions/blob/master/avg1.c

```
//function no argument no return
#include<stdio.h>
int avg();
int main()
{
   avg();
   return 0;
}
int avg()
{
   int a,b,c,sum=0;
   float avg=0;
   printf("enter 3 numbers \n");
   scanf("%d %d %d",&a,&b,&c);
   sum=a+b+c;
   avg=sum/3;
   printf("the avg of 3 no. %.1f \n",avg);

}
```

b. Function with argument without return value

Github link :-
https://github.com/Amarjith-c-k/different-type-of-functions/blob/master/avg3.c

```
//function with argument and without return value
#include<stdio.h>
int avg(int a,int b,int c);
int main()
{
   int a,b,c;
   printf("enter 3 numbers \n");
   scanf("%d %d %d",&a,&b,&c);
```

```
   avg( a, b, c);
   return 0;
}
int avg(int a,int b,int c)
{
   int sum;
   float avg;
   sum=a+b+c;
   avg=sum/2;
   printf("average of 3 numbers %.1f",avg);
}
```

c. Function without argument with return value

Github link :-
https://github.com/Amarjith-c-k/different-type-of-functions/blob/master/avg2.c

```
//function withno argument with return value
#include<stdio.h>
int avg();
int main()
{
   float a;
   a=avg();
   printf("the average value %.1f",a);
   return 0;
}
int avg()
{
   int abscise=0;
   float avg=0;
   printf("enter 3 numbers \n");
   scanf("%d %d %d",&a,&b,&c);
   sum=a+b+c;
   avg=sum/3;
   return avg;
}
```

d. Function with argument and return value

Github link :-
https://github.com/Amarjith-c-k/different-type-of-functions/blob/master/avg4.c

```
//Function with argument and return value
#include<stdio.h>
int avg(int a,int b,int c);
int main()
{
```

```
    int a,b,c;
    float av;
    printf("enter 3 numbers \n");
    scanf("%d %d %d",&a,&b,&c);
    av=avg(a,b,c);
    printf("the average value %.1f",av);
    return 0;
}
int avg(int a,int b,int c)
{
    int sum=0;
    float avg=0;
    sum=a+b+c;
    avg=sum/3;
    return avg;
}
```

RECURSION

A function is called recursive if the body of a function calls the same function. if else statement can be used where one branch makes the recursive call and the other doesn't.

Syntax

void recurse() //invoked function

{

… .. …

recurse(); //again continues the execution of the function

… .. …

}

int main()

{

… .. …

recurse(); //this calls the recurse function

… .. …

}

The recursive function is controlled by using if else statement by putting in some condition to satisfy the recursion.

Advantages and Disadvantages of Recursion

Recursion makes program elegant. However, if performance is vital, use loops instead as recursion is usually much slower. That being said, recursion is an important concept. It is frequently used in data structure and algorithms.

IMPORTANT QUESTIONS

1. What is recursion ?

   A function is called recursive if the body of a function calls the same function.

2. What is the advantage of recursive function?

   Its easy for one to call the function to execute it again. It reduces the size of the programme. Easy to solve out problems.

3. What is the necessity of exit condition in recursion?

   Exit condition is needed in recursion because if exit condition is not there the programme will continue to be executed based on the memory of the computer and sometimes we may not be getting the result as well.

Github Link for all the below programs :- https://github.com/Amarjith-c-k/recursion-factorial-and-fibonacci

4. Write a programme to find factorial of a given number using recursion and without using recursion?

   Without recursion

   https://github.com/Amarjith-c-k/recursion-factorial-and-fibonacci/blob/master/factorial.c

```
//factorial of a number without using recursion
#include<stdio.h>
int fct(int n);
int main()
{
   int n,f;
   printf("enter the number to find factorial \n");
   scanf("%d",&n);
   f=fct(n);
   printf("factorial of %d is %d",n,f);
   return 0;
}
int fct(int n)
{
   int f=1,i;
      for(i=1;i<=n;++i)
         {
         f *=i;
         }
   return f;
}
```

   With recursion

   https://github.com/Amarjith-c-k/recursion-factorial-and-fibonacci/blob/master/factorialr.c

```
//factorial of a number using recursion
#include<stdio.h>
int fct(int n);
```

```
int main()
{
   int n,f;
   printf("enter the number to find factorial \n");
   scanf("%d",&n);
   f=fct(n);
   printf("factorial of %d is %d",n,f);
   return 0;
}
int fct(int n)
{
   if(n>1)
      return n*fct(n-1);
   else
      return n;
}
```

5. Write a programme on Fibonacci series using recursion and not

Using recursion

https://github.com/Amarjith-c-k/recursion-factorial-and-fibonacci/blob/master/fibonacci.c

```
//fibonacci series using recursion
#include <stdio.h>
int fb(int n)
{
   if ( n == 0 )
     return 0;
  else if ( n == 1 )
     return 1;
  else
     return ( fb(n-1) + fb(n-2) );
}
int main() {
   int c,i=0, n;
   printf("Enter the number of terms: ");
   scanf("%d", &n);
   printf("Fibonacci series\n");
   for ( c = 1 ; c <= n ; c++ )
   {
      printf("%d, ",fb(i));
      i++;
   }
   return 0;
}
```

Without recursion

https://github.com/Amarjith-c-k/recursion-factorial-and-fibonacci/blob/master/fibonaccir.c

```
//fibonacci without usng recursion
#include <stdio.h>
int fb(int n)
{
      int i,t1 = 0, t2 = 1, nextTerm;
      printf("Fibonacci Series: ");
      for (i = 1; i <= n; ++i)
   {
      printf("%d, ", t1);
      nextTerm = t1 + t2;
      t1 = t2;
      t2 = nextTerm;
   }
    return 0;
}
int main() {
   int i, n;
   printf("Enter the number of terms: ");
   scanf("%d", &n);
   fb(n);
   return 0;
}
```

STORAGE CLASSSES

Declaration of variables contains two part: -

Data type of the variable it tells the type of information represented by a variable, e.g., integer number, floating number, character etc

Storage class refers to the portion of the program over which the variable is recognized. There are basically two types of location **Memory and CPU register** and the variable of the storage class determine the location of the stored value.

Storage class of the variable defines

- Place where the variable should store
- Default or initial value of the variable
- Scope of the variable, the function where the value of variable is available
- Life of the variable how long it exists

Storage classes in C 4 types

- Automatic storage class.
- register storage class
- Static storage class.
- external storage class

Automatic storage class

Synatax: **-** auto int x;

In here it exist only under the variable in which it is declared. The features of such storage class are

- **Storage:** Memory
- **Default initial value:** An unpredictable value, garbage value
- **Scope:** local to the variable in which it is declare
- **Life:** Control remains within the block in which it is declared

e.g., for(int i=1;i<6;i++)

{ printf(“hello world);

Printf(“%d”,i);

In the above program an error that I is not declared will be out because it is not declared outside the function for.

Register storage class

***Syntax:* -** register int x;

It is used in the case where we need to use register memory. These register memories are special storage unit inside the CPU and basically the arithmetic and logical operation comprising a programme are carries out here. These are basically the used to execute data faster and it should be used limited as it may be used by other applications within the computer and are found very less in the CPU.

- **Storage:** CPU registers
- **Default intial value:** garbage value
- **Scope:** local to which the variable is declared
- **Life:** Till the control is defined

Static storage class

***Syntax:* -** static int x;

It's a single file programme and the value of the static variable exist throughout the programme. Its value gets stored with the conditions its called up and throughout the programme. It is used when we want the value to be persist between different function call

- **Storage: -** Memory
- **Default initial value: -** zero
- **Scope: -** local to the function where it is declared
- **Life: -** Value of variable persist between different function call.

e.g., main()

{ display();

Display();

Void display()

{

Static int c=1

C +=5;

Printf("%d ",c)

}

Output

6 11

We can see that the first function call makes the value of c to 6 and in the next call the changed value 6 is used that is this call changes the value of variable declared

External storage class

It is written outside all the function and it is also known as global variable. And any storage class specifiers is not used. Should be used in the case where it is urgent. And unnecessary declaring variable as global may result in wastage of storage space.

- **Storage: -** memory
- **Default initial value: -** zero
- **scope: -** global
- **life: -** as along the program as execution doesn't come to end

e.g., int n=5;// global variable

int main()

{ ++n;

Display();

}

Int display()

++n;

Printf("n=%d",n);

}

Output

n = 7

Storage classes	**Storage**	**Default initial value**	**Scope**	**Life**
Automatic	memory	Garbage value	Local to the function or block	Till the function is terminated
Register	CPU register	Garbage value	Same as above	Same as above
Static	Memory	Zero	Same as abo0ve	Value passes between different function calls
External	Memory	Zero	Global	As long as the program's execution ends

Important questions

1. **What does the storage class of a variable refer to?**
   Storage class refers to the portion of the program over which the variable is recognized. There are basically two types of location **Memory and CPU register** and the variable of the storage class determine the location of the stored value.
2. **What does storage class of a variable define?**
   - Place where the variable should store
   - Default or initial value of the variable
   - Scope of the variable, the function where the value of variable is available
   - Life of the variable how long it exists

3. **Where is the values of variables defined inside a program stored?**
   Basically, there are 2 locations where the variable declared in a storage can be stored memory and CPU registry.

4. **Name the storage classes used in C.**
   - Automatic storage class.
   - register storage class
   - Static storage class.
   - external storage class

5. **Write down the syntax for a)Automatic storage class b)Register storage class c) Static storage class d)External storage class**
   a, auto int x;
   b, register int x;
   c, static int x;
   d, int n=5; // declared outside the programme
6. **What is the default storage class?**
   Default storage classes are the storage class assumption made by the compiler when we don't make any specifications.
7. **Write down the features of variables defined inside a)Automatic storage class b)Register storage class c) Static storage class d)External storage class**

Storage classes	**Storage**	**Default initial value**	**Scope**	**Life**
Automatic	memory	Garbage value	Local to the function or block	Till the function is terminated
Register	CPU register	Garbage value	Same as above	Same as above
Static	Memory	Zero	Same as abo0ve	Value passes between different function calls
External	Memory	Zero	Global	As long as the program's execution ends

8. **What is the advantage of declaring a variable as a register variable?**
   These are faster than local variables these are special unit present in the CPU. And the arithmetic and logical operations inside a programme is carried out here.
9. **Which type of variables are to be declared inside register storage class?**
   only the variables that are used very often in a program should be declared in register class as there are only few CPU registers available.
10. **Applications of register storage class variables**
    Loop counters because it get used a number of times in a program.

11.What is the difference between auto and static variables?

The value of auto variable exist only in the function block and its default initial value is garbage value, While in the case of static variable the value exist throughout the programme

12.Which type of variables are to be declared as extern? Why?

The variables that need to be accessed throughout the programme within all the functions

13. **What is the disadvantages of using external variables?**
    Declaring all the variables as extern will result in the wastage of memory space as they remain active throughout the programme

14. What is the output of the following program?

#include int main()

{ static int a = 3;

printf("%d", a --);

return 0; }

a) 0

b) 1

c) 2

d) 3

the answer is 2

15.What will be the output of the following program?Justify your answer.

#include

void main()

{ fun();

Return 0; }

void fun()

{

auto int I = 1;

register char a = ‘D’;

static int p = 0;

printf(“%d %d %ld”, I, a, p);

}

a) 1 D 0

b) 1 0 0

c) 0 D 1

d)1 68 0

The answer is 1 68 0 as in this case a is taken as character and the integer value of ‘D’ is asked to print which will print the ASCII value all othe variables they are integer variables.

16. What will be the output of the program? Justify your answer.

extern int i = 5;

main()

{ int i=7;

printf("%d",i);

}

A. 5

B. compiler error

C. 7

D. garbage value

7 is the answer because a value is declared inside the function and the printf is inside that function of the value declared. 5 will be displayed only if the I is not initialised and simply called in the function.