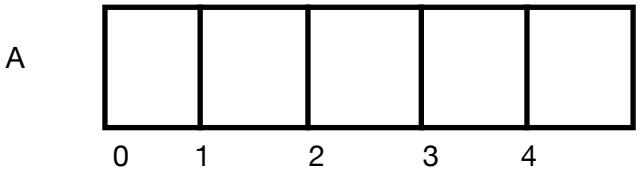
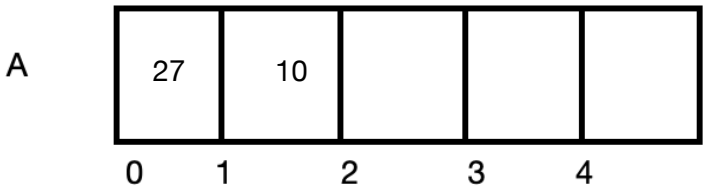
Arrays

- Arrays is defines as collection of similar data types.



- *Declaration* of arrays is as follows int A[5] ;
- Each location can be *accessed* as A[0] = 27;
  A[1] = 10;



- The array will be created in the *main memory* if inside a function
- The Declaration and Initialisation of array is as follows

  Int B[5] = { 2, 9, 6, 8, 10 } ;

- You can also access each element of array using *for loop*

DataStructures :

Structures

- Collection of data members under one name is structure
- Data members can be of similar type or non similar type
- When structures is called in the main( ) program then it will consume space accordingly to the data members types it contains in the memory

An example of structure is a program of a rectangle

```
Struct Rectangle

{
        int length;
        int breath;

}

Int main( )

{

struct Rectangle r ;                    - Declaration

Struct Rectangle r = { 10, 5 } ;        - Declaration + Initialisation

r.length = 15 ;                         - . Is is used to access a member

r.breath = 10 ;


Printf( " Area of rectangle is %d" , r.length * r.breath ) ;  - Accessing the members

}
```

Use of structures :

Structures is used to combine data under one name , thus some example usage of structures is

- In Complex numbers
- In student details
- In Employee Details
- Bank Details etc
- Defining Shapes etc…

Pointers

- Pointer is a address variable that is meant for storing address of not data itself
- They are used for indirect access of data
- For a program to use heap memory , pointers is used
- To access heap memory and resources outside the main memory like internet, keyboard , monitor etc pointers is used
- Pointers are also used for parameter passing

Example :

```
Int main( )
{
Int a = 10 ;                    - data variable
Int *p ;                        - declaration
P = & a ;                       - Assignment / Initialisation
printf( “% d ” , a ) ;
printf(“ %d ” , * p ) ;         - dereferencing
}
```

- Accessing Heap memory through pointer

```
#include<stdio.h>

Int main( )

{

Int * p;

P = new int[5];

}
```

Reference

- A reference is a just another name for the same the variable

Example :

```
Int main( )
{
Int a = 10;
Int &r = a;        - syntax of reference
Count << a ;
r++;
Cout<< r ;
Cout << a ;
}
```

- This is use for parameter passing
- For writing small functions we use reference

Pointer to a Structure

Syntax - 1

- When variable is already ***existing*** , the we can use pointer to structure like

```
Struct Rectangle

{
        int length;
        int breath;

}

Int main( )

{

Struct Rectangle r = { 10, 5 } ;

Struct Rectangle *p = &r;

r.length = 15 ;
```

P-> length = 20 ;

Or

(*p).length = 20 ;

Syntax - 2

- ***Dynamically*** object created in heap and pointer pointing there

```
Struct Rectangle

{
        int length;
        int breath;

};

Int main( )

{

Struct Rectangle *p;

(Struct rectangle * ) malloc (sizeof (struct rectangle));

P -> length = 10 ;

P -> breath = 5;

}
```

Functions

- Function is a piece of code which performs a specific task
- Grouping instructions is called function
- They are called as modules or procedures
- The main task can be divided into several small task in the form of functions this type of programming is also called as modular or procedural programming
- It is easy for development
- A group of programmers can work on a single project using functioning
- Functions provide reusability of code
- It can be used in other software projects as well
- you can group function into library

Example :

```
int add( int a , int b )

{

int c;

c = a + b;

return c;

}
```

```
Int main ( )
{

int x,y,z;

x =10;

y = 5;

z = add x, y;

printf(“sum is  %d” , z);

}
```