Chapter-1

Structures and Pointers

Structure is a user defined data type which representsa collection of logically related data items.

Defining a structure

To define a structure you use the struct statement.The format of structure statement is

struct [struct tag]

{

   Member definition;

   Member definition;

};

The members of a structure is also called structure elements.

For Example:

struct student

  {

     int rollno;

     char grade;

     float percentage;

  };

Accessing elements of a structure

The elements of a structure are accessed using the dot (.) operator or Period operator.The syntax for accessing elements is,

[Object name][Operator][Member variable Name]

For Example: s.name.

```
#include<iostream>

using namespace std;

struct student

{

    int rollno;

    int m1,m2,m3;

};

int main()

{

    student s;    //Object declared

    cout<<"Enter the marks in Three subjects";

    cin>>s.m1>>s.m2>>s.m3;

    cout<<"Total="<<s.m1+s.m2+s.m3;

return(0);

}
```

Nested Structure

A Structure placed inside another structure is called a nested structure.

Difference between Array and Structure

Array	Structure
An array is a collection of similar data type(Homogeneous)	A structure is a collection of different data types(Heterogeneous)
The elements of an array are accessed using an index	The elements of a structure is accessed using dot(.) operator
Array is derived data type	Structure is user defined data type

Pointer

The data stored in computer occupies a memory cell.Each cell has a unique address.A pointer points to the address of memory location.It holds the address of the memory location.

Declaring a pointer

We can declare a pointer,similar to a variable.The syntax is

Data_Type *Pointer_Variable;

For Example:

 int *ptr;

Here ptr is a pointer variable and points to an integer data type.

Initializing a pointer
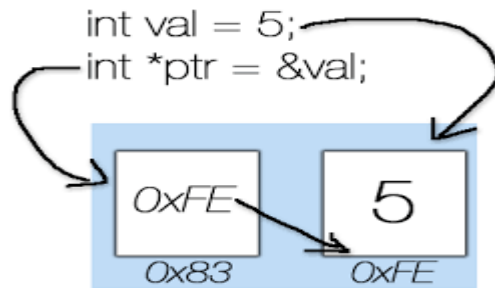
We can initialize a pointer as,

Ptr=&a;

Here the pointer variable **ptr** contains the address of variable a.

The content of or deferencing operator or indirectionoperator (*****) is used to get the value stored at the location held by a pointer variable.The * operator is also known as value at operator.

The address of operator(**&**) is used to retrieve the address of a variable.


```
#include<iostream>

using namespace std;

int main()

{

int a=10;

int *Ptr=&a;

cout<<"The value of a="<<a<<"\n";

cout<<"The address of a="<<Ptr;

return 0;

}
```

Output:

The value of a=10

The address of a= oxffoo;

Memory allocation

Allocation of memory for a variable during compilation time is known as static memory allocation.Once the memory is allocated during compile time it cannot be expanded or compressed.

For example:

   int a[10] as integer;

During compilation the compiler will allocates 20 bytes .

The process of allocating memory during execution time is called **dynamic memory allocation**.This is done using **'new'** and **'delete'** operators.

Memork leak is a situation where once memory is dynamically allocated to a variable and if later the memory is not de-allocated after its purpose is satisfied,the memory gets lost.This problem where memory is dynamically allocated but not released ,hence not accessible to any problem is called memory leak.

new and delete operators

The *new* operator is used to allocate memory during execution(run-time).It is a unary operator.The syntax is

   pointer_variable=new data_type;

The *delete* operator is used to free memory allocated using new operator.The syntax is

   delete pointer_variable;

#include<iostream>

using namespace std;

int main()

{

```
int i,*p;

 p=new int[10];

 cout<<"Enter the elements";

for(i=0;i<10;i++)

{

cin>>p[i];

}

for(i=0;i<10;i++)

{

cout<<p[i]<<"\t";

}

delete[]p;

return 0;

}
```

Pointers and array

There is a close relationship between pointers and array.C++ treats the name of array as a pointer.The name of array is a pointer pointing to the first element of the array.

For example:

char ch[10],* ptr;

ptr=ch;

here ptr points to the address of first array element in ch.To access fifth element you can use

ch[4] ;   or   *(ptr+4);

Note:Currently ptr points to address of first element in the array ch.

Operations on pointers

Arithmetic operations can be performed on a pointer.

For Example;

int a[50];

int *ptr;

ptr=&a[0] ; //ptr refers to the base address of array a.

ptr - - or   - - ptr;

Moves the pointer to the previous address.(If the base address was 1000 the operation – ptr moves the pointer to 998 memory location ie,1000 – 2.

cout<<ptr; //Displays 1000 ie, the address of a[0](Base address);

cout<<*ptr;//Displays the value at a[0];

Program to find average marks of students

```
#include<iostream>

using namespace std;

int main( )

{

int n,sum=0,*mark_ptr,i;
```

```
float avg;

mark_ptr=new int;

cout<<"Enter the No of Students";

cin>>n;

for(i=0;i<n;i++)

 {

  cin>>*(mark_ptr + i);

  sum=sum+*(mark_ptr+i);

 }

avg=(float)sum/n;

cout<<"Average="<<avg;

return 0;

}
```

Note: Only equality**(= =)** and not equality**(! =)** operators can be applied to a pointer.
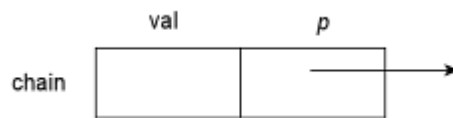
Self referential structures

When a member of a structure is declared as a pointer to the structure itself,then the structure is called as self referential structure.

For example:

```
struct  chain {

 int val;

chain *p;
```

};

The structure has two members val and p.The member p is a pointer to a structure of type chain.

Some examples of self referential structures are **linked list,stack,trees** etc.

Dynamic array

A dynamic array is an array created during run time using new operator.The synatx for creating a dynamic array is

Pointer= new data_type[size];

For example:

P=new int[10]; //declares a dynamic array of size 10.

Pointer and String

A string is an array of characters,and array name can be considered as a string variable.Every string in C++ is terminated by a Null character**(\0).**For example the string *hello* is represented in memory as

Intializing an array

The above array can be initialized as

char ch[ ]="hello";

or

char ch={'h','e','l','l','o','\0'};

We can use a character pointer to reference the array.

char s[10];//character array declaration

 char *ptr;//character pointer declaration;

 cin>>s;

 ptr=s;//Copying contents of s to ptr

Example:

int main()

 { char a[10];

 strcpy(a, "hello");//copies the string hello to a

 cout << a; return(0);

 }

Output:

hello

Advantages of character pointer

- Strings can be managed by optimal memory space.
- Assignment operator can be used to copy strings.
- No wastage of memory.