	SVKM's NMIMS **Mukesh Patel School of Technology Management & Engineering / School of Technology Management & Engineering**

B. Tech/MBA Tech	**Workbook**	**Academic Year- 2023-24**
Year:-First	**Subject:-** Programming for Problem Solving	**Semester: - First**

Experiment: 8

PART A

(PART A: TO BE REFERRED BY STUDENTS)

Aim: Programming using structure and pointer

Learning Outcomes: The learner would be able to

1. Understand the syntax structure & pointers.
2. Solve problems using structure
3. Implement programs using pointers

Theory:

Structures

- **Structure** is user defined data type.
- Structure is used to store heterogeneous data under unique name.
- Keyword 'struct' is used to declare structure.
- In other word structure is a convenient tool for handling a group of logically related data items of different data types.
- Following are the different instances of structure

Syntax 1.

```
struct structure_name{
        <data-type> member1;
        <data-type> member2;
        - - - - - - - - - - -
        - - - - - - - - - - -
        <data-type>member_n;
```

Example of Syntax 1 :

```
struct student{
        int rollno;
        char nm[10];
        float wt,ht;
};
```

For this structure we can declare structure

Syntax 2.

```
struct structure_name{
        <data-type> member1;
        <data-type> member2;
        - - - - - - - - - - -
        - - - - - - - - - - -
        <data-type>member_n;
}struct_var_list;
```

Example of Syntax 2 :

```
struct student{
        int rollno;
        char nm[10];
        float wt,ht;
}s1,s2,s3,s4,s5;
```

- Above syntax shows, structure definition with structure variable declaration.
- Struct_var_list is list of structure variables.

Syntax 3.

```
struct{
        <data-type> member1;
        <data-type> member2;
        - - - - - - - - - - -
        - - - - - - - - - - -
        <data-type>member_n;
}struct_var_list;
```

- Above syntax shows, structure definition with structure variable declration.
- Struct_var_list is list of structure variables as we have structure variable that's why we can omit structure name.

Example of Syntax 3 :

```
struct{
        int rollno;
        char nm[10];
        float wt,ht;
}s1,s2,s3,s4,s5;
```

Accessing Members of a Structure

- C++ provides two operators to access structure members viz dot & arrow.
- Dot(**.**) is an operator which is used to work with structure variable.
- Arrow (- >) is another operator, to work with structure with pointer.
- **Syntax:-**

  Structure_variablename.member;
- **Example:-**

  s1.rollno;

Assigning Values to structure members:

- With the help of assignment operator we can assign value to the structure members.
- **Syntax:-**

  Structure_variable.member=value;
- Example:-

```
s1.rollno=50;
s1.name="Rohit";
s1.wt=68.00;
s1.ht=5.7;
```

Initialization of structure: -

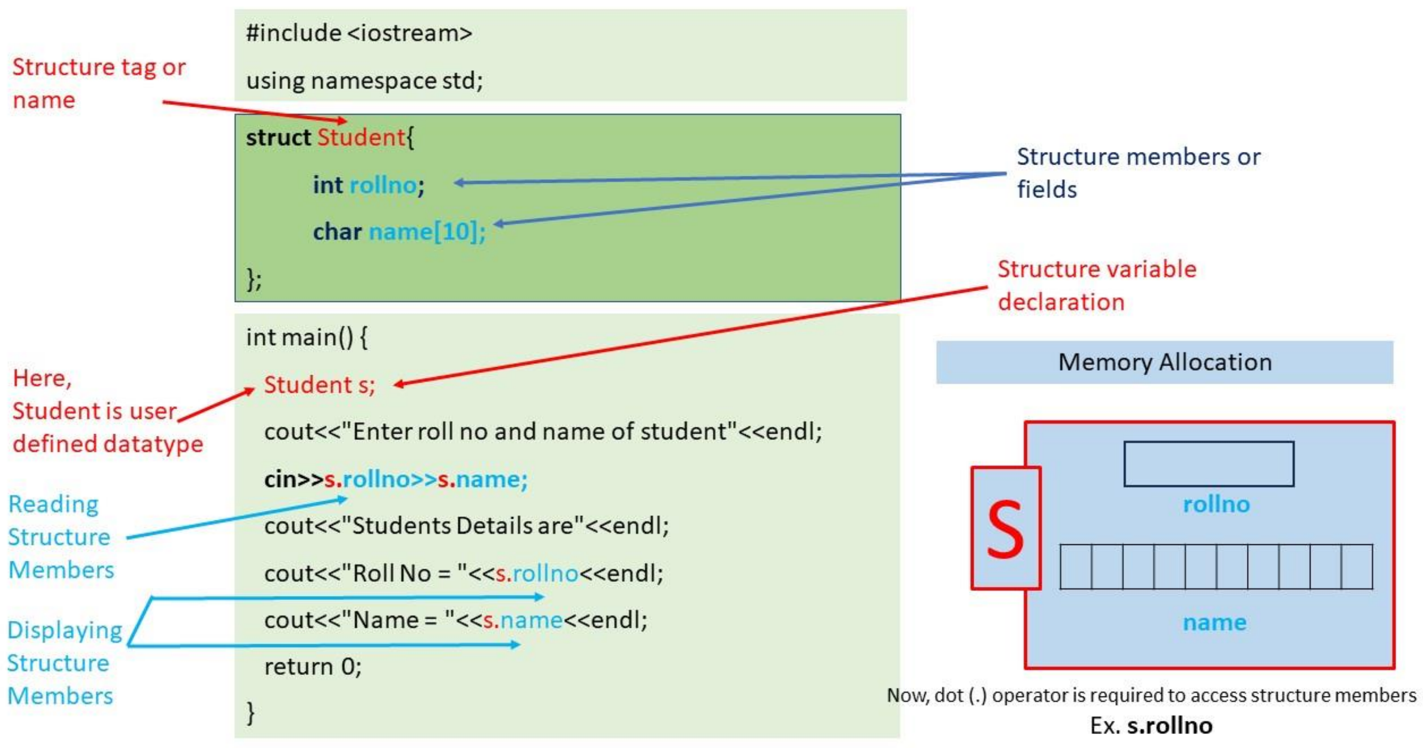
- We can initialize structure as like as array by specifying list of values in curly bracket.
- Syntax:

  struct structure_name variable_1= { list of values separated by comma };

  Example:-

  struct student s1={50,"Rohit",68.00,5.7};

Write a program to read and display Student information like roll number & name using structure...



```
#include <iostream>
using namespace std;
struct Student{
        int rollno;
        char name[10];
};
int main() {
   Student s;
   cout<<"Enter roll no and name of student"<<endl;
   cin>>s.rollno>>s.name;
   cout<<"Students Details are"<<endl;
   cout<<"Roll No = "<<s.rollno<<endl;
   cout<<"Name = "<<s.name<<endl;
   return 0;
}
```

Structure tag or name
Structure members or fields
Structure variable declaration
Here, Student is user defined datatype
Reading Structure Members
Displaying Structure Members
Memory Allocation
S
rollno
name

Now, dot (.) operator is required to access structure members
Ex. **s.rollno**

Array With in Structure:-

- We can have array within a structure
- For example, if we have structure members student_name, & student_marks etc.
- Example:-

```
struct student{
        int rollno;
        char nm[10];
        int marks[5];
};
```

- In above example nm and marks is array within structure

Array of Structure: -

- We can have array of structure
- Syntax
  struct structure_name arrayname[size];
- Example
  struct student s[100];

Write a structure cricket with information like Player Name, Team Name & Batting average. Write a program to read & display information of 11 players of a team.

```
#include <iostream>
using namespace std;
struct Cricket{
    char pname[10];
    char tname[10];
    float bavg;
};
int main() {
    Cricket c[11];

    cout<<"Enter information of 11 players "<<endl;
    for(int i=0;i<2;i++)
        cin>>c[i].pname[i]>>c[i].tname>>c[i].bavg;

   cout<<"Plyer info:"<<endl;
    for(int i=0;i<2;i++){
        cout<<"Plyer "<<c[i].pname<<endl;
        cout<<"Team "<<c[i].tname<<endl;
        cout<<"Avg "<<c[i].bavg<<endl;
    }

    return 0;
}
```

B. Tech/MBA Tech	**Workbook**	**Academic Year- 2023-24**
Year:-First	**Subject:-** Programming for Problem Solving	**Semester: - First**

Pointers: -

- Pointer is variable, which holds memory address of another variable
- **Pointer Operators:-**
    - **&** - **Address of or Direction or referencing operator.**
      Address of Operator (&) returns the address allocated to the variable.
      Syntax:-
      ```
      & variable_name;
      ```
      Example:-
      ```
      &i;
      ```
    - ***** - **Value at address or Indirection** or **dereferencing pointer operator**
      Value at address (*) operator returns the value stored inside address
      '*' is used to declare pointer variable.
      Syntax:-
      ```
      *memory_location;
      ```
      Example:-
      ```
      *(&i);
      ```
- Suppose we have variable declaration

  int i=5; ⟵ Normal Variable

  int *p ⟵ Pointer Variable

  p=&i; /*Assigning address of variable i to pointer p*/
  then following three things will take place, for each variable in memory.

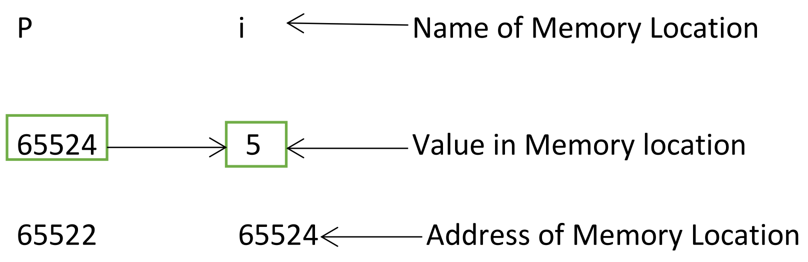


Fig:- Memory allocation for variable

In above fig, p is pointer variable (contains address of i) & that points to variable i.

Pointer Variables Declaration:

- Like normal variable declaration, we can declare pointer variable using value at address operator(*)

  Syntax:-

  ```
  data_type *variable_name;
  ```

  Example:-

  ```
  int *ptr;
  float *ptr2;
  char *ch;
  ```

What information pointer variable contains?

- Pointer may be initialized to zero, NULL or an address(if we know).
- Initializing pointer to zero is equivalent to initializing a pointer to NULL.
- NULL is symbolic constant available in "stdio.h".
- Example:-
  1. int *p=0;
  2. int *p=NULL;
  3. int *p=&i;

     In example 3 'i' is normal variable & address of i is assign to p.

Programming Example to show use of pointer:

```
#include<stdio.h>

main( ){                        ←———————— Declaration of normal variable

        int i=5;                ←———————— Declaration of pointer variable.

        int *p;                 ←———————— Assigning address to pointer variable.

        p=&i;

        cout<<p<<endl;    ←———————— Display value of P.

        cout<<*p;         ←———————— Display value at address, see fig 2.

}
```

Output:-

```
65524
5
```

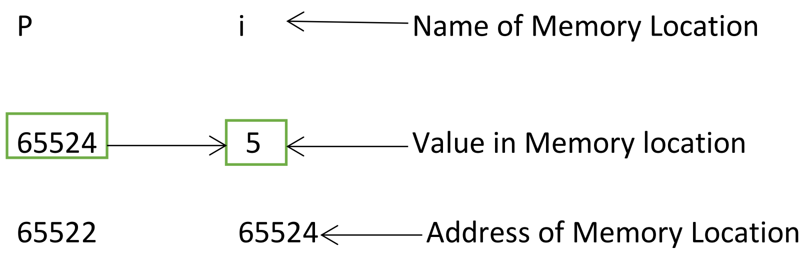


Fig 2:- Memory allocation for i & p

Pointer Arithmetic:-

- Integer constant can be added or subtracted from pointer
- Pointer can be incremented or decremented

  Note:- We considered int as 2 byte, in c++ int is of 4 byte, so difference will be of 4.

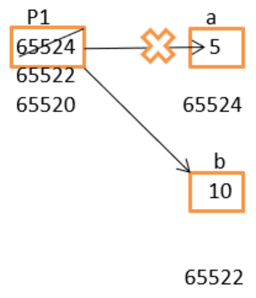
```
main(){
int a=5,b=10;
int *p1;
p1=&a;
cout<<*p1;
p1=p1-1;
cout<<*p1;
}
```

Pointer Points to 65524 that is a=5.

Now, Pointer Points to 65522 that is b=10

Output:-

```
5
10
```



- Pointer assignment

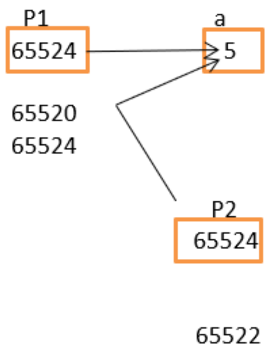
```
main(){
int a=5,;
int *p1,*p2;
p1=&a;
p2=p1
cout<<*p1;
cout<<*p2;
}
```

Pointer Points to 65524 That is a=5.

Pointer Assignment, Now both pointers points to same memory location.

Output:-

```
5
5
```



- One pointer can be subtracted from another pointer.
- Pointers can be compared.
    - o Two pointers can be compared using relational & logical operators like(<,>,<=,>=,==,!=)

1. We cannot add two pointers.
    - o Two pointers cannot be added.
2. Cannot be multiplied or divided by integer constant.
    - o Pointer cannot be multiplied and/or divided by integer constant.
    - o Two pointers cannot be multiplied or divided.

Pointer to Pointer (Chain of a Pointer).

- One pointer variable contains address of another pointer variable.
- **General syntax is,**

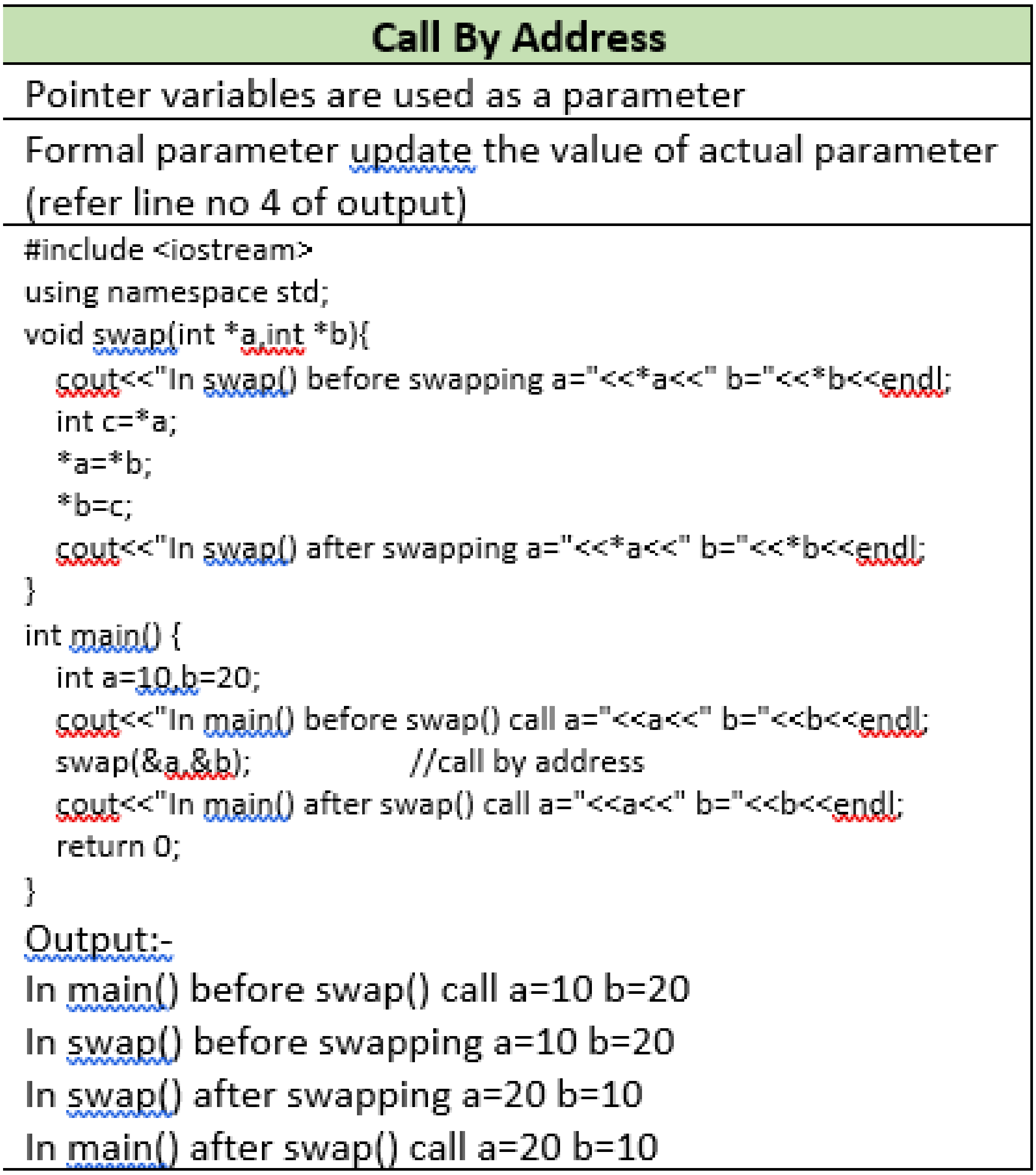
```
Data_type **pinter_to_pointer_var_name;
```

- **Example:-**

```
int **p;
```

- **Example:-**

int a;	-	Normal/ordinary variable
int *p	-	pointer variable
int **p	-	pointer to pointer
int ***p	-	pointer to pointer to pointer



Call By Address

Pointer variables are used as a parameter

Formal parameter update the value of actual parameter (refer line no 4 of output)

```
#include <iostream>
using namespace std;
void swap(int *a,int *b){
   cout<<"In swap() before swapping a="<<*a<<" b="<<*b<<endl;
   int c=*a;
   *a=*b;
   *b=c;
   cout<<"In swap() after swapping a="<<*a<<" b="<<*b<<endl;
}
int main() {
   int a=10,b=20;
   cout<<"In main() before swap() call a="<<a<<" b="<<b<<endl;
   swap(&a,&b);                //call by address
   cout<<"In main() after swap() call a="<<a<<" b="<<b<<endl;
   return 0;
}
```

Output:-

```
In main() before swap() call a=10 b=20
In swap() before swapping a=10 b=20
In swap() after swapping a=20 b=10
In main() after swap() call a=20 b=10
```

Tasks:

1. Define structure Student having data members: Roll no, name, address, branch and percentage. WAP to read and display information of a student.
2. Define structure called cricket that will describe the following data Player name, country name, no of matches played & batting avg. Develop a program that store information of 10 players and display names of players having batting average greater than 50.
3. There are 50 computers in an office. Every computer has following information CPU type, hard disk size. WAP to store details of all 50 computers and then print details of computers having hard disk size greater than 8 GB.
4. WAP to print array using pointer
    1. WAP to print string in reverse order using pointers
5. WAP to copy one string to another using pointer and display copied string-using pointer.
6. WAP to find the number of vowels in entered string using pointer [eg **- i/p** India **o/p**→ A-1, E-0, I-2, O-0, U-0]

	SVKM's NMIMS Mukesh Patel School of Technology Management & Engineering / School of Technology Management & Engineering	
B. Tech/MBA Tech	**Workbook**	**Academic Year- 2023-24**
Year:-First	**Subject:-** Programming for Problem Solving	**Semester: - First**