

Object Oriented Programming

Structures

Mr. Usman Wajid

usman.wajid@nu.edu.pk

February 10, 2023



National University
of Computer & Emerging Sciences

What is a Structure?

Structure

A structure is a collection of related data items that can be referenced with a single name

- Also known as records

What is a Structure?

Structure

A structure is a collection of related data items that can be referenced with a single name

- Also known as records
- The data items in structure are called members

What is a Structure?

Structure

A structure is a collection of related data items that can be referenced with a single name

- Also known as records
- The data items in structure are called members
- A struct is heterogeneous as its data members can be of different types

What is a Structure?

Structure

A structure is a collection of related data items that can be referenced with a single name

- Also known as records
- The data items in structure are called members
- A struct is heterogeneous as its data members can be of different types
- Whereas, array is homogeneous as it can only store items of same data types

① Structure Definition

```
struct <identifier>{  
  <data-type> <identifier_list>;  
  <data-type> <identifier_list>;  
  .....  
};
```

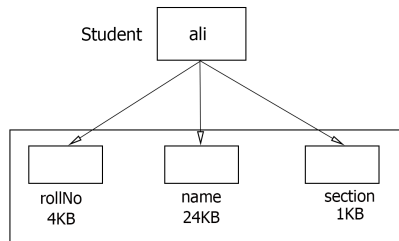
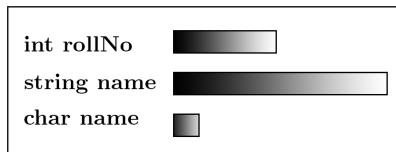
- structure definition acts like a blueprint and doesn't hold any space in memory

Structure Example

```
struct Student{  
    int rollNo;  
    string name;  
    char section;  
};  
  
Student ali;
```

- When a struct type variable is declared, a space in memory is reserved just like variable of other data types
- The memory structure for the above structure variable, i.e., ali is depicted below

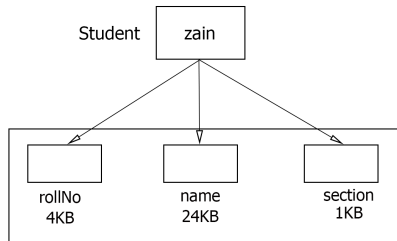
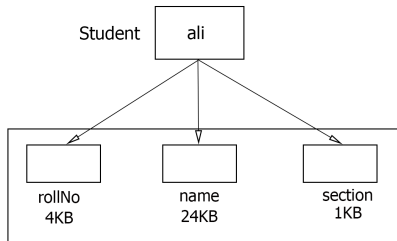
Student ali



Declaration of a variable of struct type

- Example continued,

```
Student ali, zain;
```



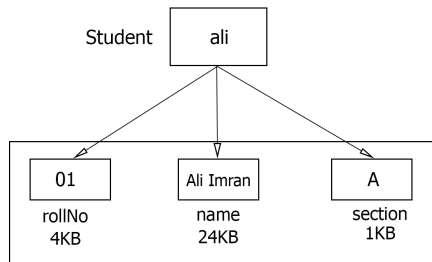
Accessing struct members

- The member of a struct type variable are accessed with the dot (.) operator
- Syntax,

`<struct-variable>.<member_name>;`

```
#include <iostream>
using namespace std;
struct Student{
    int rollNo;
    string name;
    char section;
};
Student ali;
int main(){
    ali.rollNo = 01;
    ali.name = "Ali Imran";
    ali.section = 'A';

    cout<<"Roll no: "<<ali.rollNo<<
        endl;
    cout<<"Name: "<<ali.name<<endl;
    cout<<"Section: "<<ali.section<<
        endl<<endl;
}
```



Initializing an object of struct variable

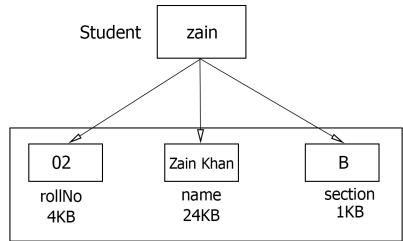
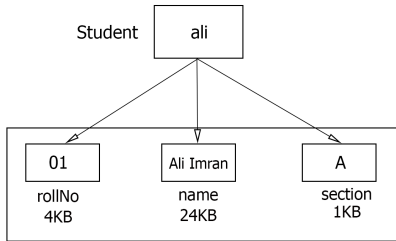
- An object of a struct variable can be initialized at once by using curly braces
- The sequence of member types must be the same as declared in the struct definition
- Example,

```
struct Student{
    int rollNo ;
    string name;
    char section;
};

Student ali;

int main(){
    ali = {01, "Ali Imran", 'A'};
    Student zain = {02, "Zain Khan", 'B'};
}
```

Initialization of struct variables



struct with default values

- assigning values to variables while defining structure member variables refers to default values
- while declaring structure variable, the default values will be automatically assigned to its members
- Syntax,

```
<struct> <identifier>{  
    int <identifer> = 00 ;  
    string <identifier> = "N/A";  
    char <identifier> = '-';  
};
```

struct default values example

```
#include <iostream>
using namespace std;
struct Student{
    int rollNo = 0;
    string name = "N/A" ;
    char section = '-';
};
Student ali;
void displayMembers(Student x){
    cout<<"Roll no: "<<x.rollNo<<endl
    ;
    cout<<"Name: "<<x.name<<endl;
    cout<<"section: "<<x.section<<
    endl<<endl;
}
int main(){

    ali = {1, "Ali Imran", 'A'};
    Student zain;

    displayMembers(ali);
    displayMembers(zain);
}
```

```
PS C:\Users\p18-0405\Documents\structures> ./a
Roll no: 1
Name: Ali Imran
section: A

Roll no: 0
Name: N/A
section: -

PS C:\Users\p18-0405\Documents\structures> █
```

copying struct variable object

- The struct variable object can be initialized by copying from other struct variable object. Example,

```
struct Student{
    int rollNo = 00;
    string name = "N/A" ;
    char section = '-';
};

Student ali;

void displayMembers(Student x){
    cout<<"Roll no: "<<x.rollNo<<
        endl;
    cout<<"Name: "<<x.name<<endl;
    cout<<"section: "<<x.section<<
        endl<<endl;
}

int main(){

    ali = {01, "Ali Imran", 'A'};
    Student zain = ali;

    displayMembers(ali);
    displayMembers(zain);
}
```

```
PS C:\Users\p18-0405\Documents\structures> ./a
Roll no: 1
Name: Ali Imran
section: A

Roll no: 1
Name: Ali Imran
section: A

PS C:\Users\p18-0405\Documents\structures>
```

Array in Structures

```
struct Student{
    int rollNo = 0;
    string name = "N/A";
    char section = '-';
    int marks[5]={0,0,0,0,0};
};

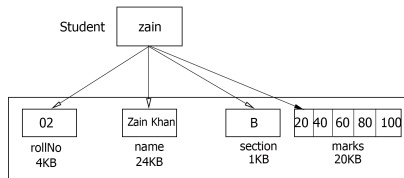
Student ali;
void displayMembers(Student x){
    cout<<"Roll no: "<<x.rollNo<<endl;
    cout<<"Name: "<<x.name<<endl;
    cout<<"section: "<<x.section<<endl;
    for (int count = 0; count < sizeof(x.marks)/sizeof(int); count++)
        cout<<"marks in subject["<<count<<"] = "<<x.marks[count]<<endl;
}

int main(){
    ali.rollNo = 01;
    ali.name = "Ali Imran";
    ali.section = 'A';
    ali.marks[0] = 10;
    ali.marks[1] = 20;
    ali.marks[2] = 30;
    ali.marks[3] = 40;
    ali.marks[4] = 50;
    Student zain = {2, "Zain Khan", 'B', {20, 40, 60, 80, 100}};
    displayMembers(ali); cout<<endl;
    displayMembers(zain);
}
```

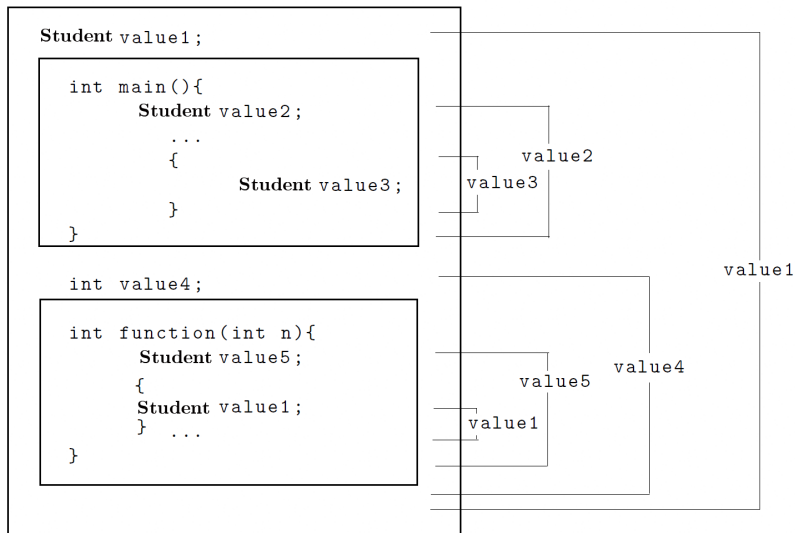
Array in Structures

```
PS C:\Users\p18-0405\Documents\structures> ./a
Roll no: 1
Name: Ali Imran
section: A
marks in subject[0] = 10
marks in subject[1] = 20
marks in subject[2] = 30
marks in subject[3] = 40
marks in subject[4] = 50

Roll no: 2
Name: Zain Khan
section: B
marks in subject[0] = 20
marks in subject[1] = 40
marks in subject[2] = 60
marks in subject[3] = 80
marks in subject[4] = 100
PS C:\Users\p18-0405\Documents\structures>
```



struct variable object scope



struct variable object scope example

```
struct Student{
    int rollNo ;
    string name;
    char section;
};

Student std1 = {01, "Ali", 'A'};

void displayMembers(Student x){
    cout<<"Roll no: "<<x.rollNo<<endl;
    cout<<"Name: "<<x.name<<endl;
    cout<<"section: "<<x.section<<endl
    <<endl;
}

int main(){

    displayMembers(std1);
    Student std1 = {02, "Zain", 'B'};
    displayMembers(std1);
    //displayMembers(std3); // cause
    error

    Student test();
    displayMembers(test());
}
Student std3 = {03, "Umer", 'C'};
Student test(){
    return std3;
}
```

```
PS C:\Users\p18-0405\Documents\structures>
PS C:\Users\p18-0405\Documents\structures> ./a
Roll no: 1
Name: Ali
section: A

Roll no: 2
Name: Zain
section: B

Roll no: 3
Name: Umer
section: C
```

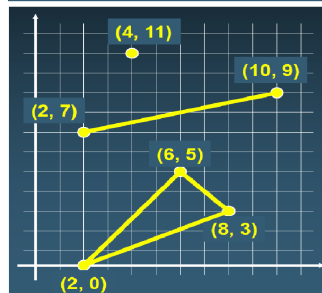
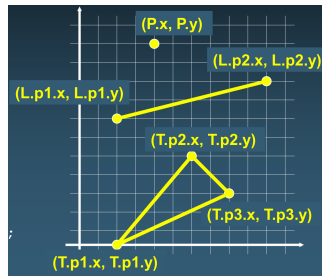
Nested Structures

```
struct point{
    int x, y;
};

struct line{
    point p1, p2;
};

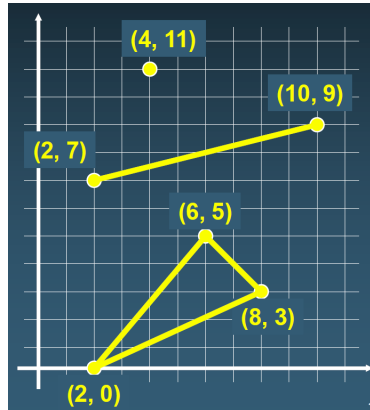
struct triangle{
    point p1, p2, p3;
};

int main(){
    point P = {4,11};
    //P.x = 4;
    line L = {{2,7}, {10,9}};
    //L.p1.x = 2; L.p1.x=7; L.p2.x=10; L.p2.y
    //      =0;
    triangle T = {{2,0},{6,5},{8,3}};
    //T.p1.x = 2;
}
```



Nested Structures continued ...

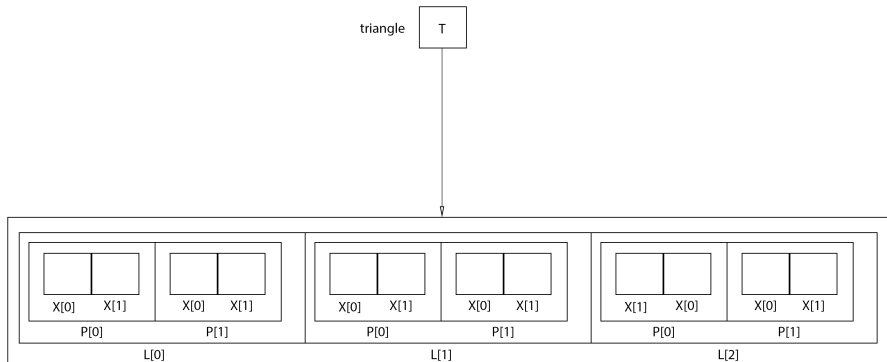
- 1 Construct the same triangle with struct line nested inside struct triangle.
- 2 Construct a square with struct line nested inside the struct square
- 3 Construct a rectangle with struct line nested inside the struct rectangle. Where, the length is double the size of width of the rectangle



Array of structures

Array of Structure

It is an array of a struct variable, where each index is an object and holds an address to the location of a structure in memory



Array of Structures continued ...

- Re-constructing the triangle using the nested array of structures

```
struct point{
    int x[2];
};

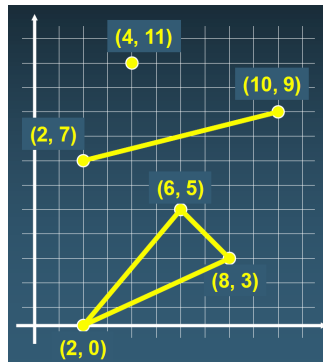
struct line{
    point p[2];
};

struct triangle{
    line l[3];
};

int main(){

triangle T= { {      {{2, 0}, {6, 5}} } ,
               {{6, 5}, {8, 3}} } ,
               {{8, 3}, {2, 0}} } } ;

}
```



① Pointing to Existing Object

```
Student ali = {1, "Ali Imran", 'A'};  
Student *ptr = &ali;
```

② Pointing to new Object

```
Student *ptr = new Student;
```

struct Pointers Example

```
struct Student{
    int rollNo = 0;
    string name = "N/A";
    char section = '-';
};

int main(){

    Student ali = {1, "Ali Imran", 'A'};

    Student *ptr;
    ptr = &ali;

    cout<<"\nrollNo: "<<ptr->rollNo<<endl;
    cout<<"Name: "<<ptr->name<<endl;
    cout<<"Section"<<ptr->section<<endl;

    Student *ptr1 = new Student;

    cout<<"\n\nrollNo: "<<ptr1->rollNo<<endl;
    cout<<"Name: "<<ptr1->name<<endl;
    cout<<"Section"<<ptr1->section<<endl;

}
```


struct Pointers Applications

① Single linklist:

```
struct Node{  
    int val = 0;  
    Node *nextNode;  
};
```

② Double linklist:

```
struct Node{  
    int val = 0;  
    Node *nextNode;  
    Node *previousNode;  
};
```