Lecture 4

structs

Bundling multiple data types

```
Data representing a patient:
  string name;
  float height;
  string bloodGroup;
```

```
Data representing a flower:
  string name;
  int numOfPetals;
```

Bundling multiple data types

```
Data representing a student:
  string name;
  int ID;
  float cgpa;
```

```
Data representing a rectangle:
  int length;
  int width;
```

struct

```
struct student{
        string name;
        float height;
};

//Compare the two data instance:
student s;
int x;
```

structs using static memory

```
int main(){
  student s;
  //Write to s
  s.name = "motiur";
  s.height = 5.7;

  //Display value of s
  cout<<s.name<<endl;
  cout<<s.height<<endl;
  return 0;
}
```

structs using dynamic memory

```
int main(){
  student *s = new student;
  //Write to s
  s->name = "motiur";
  s->height = 5.7;

  //Display value of s
  cout<<s->name<<endl;
  cout<<s->height<<endl;
  return 0;
}
```

Overview of structs

1. Static Memory
    a.structs using function for single element
    b.structs using function for multiple element
2. Dynamic Memory
    a.structs using function for single element
    b.structs using function for multiple element

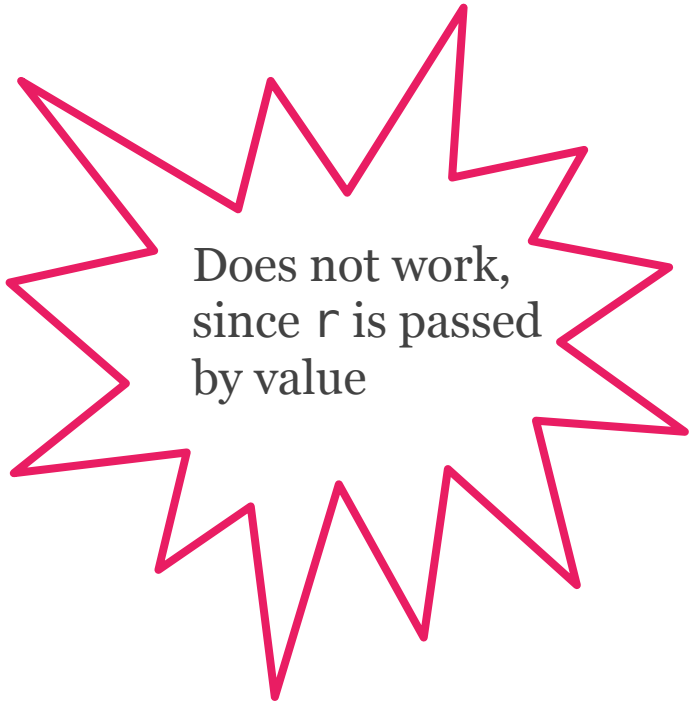
1a.structs and functions using static memory

```
//Single Element
#include <iostream>
using namespace std;

struct rectangle{
   float height;
   float width;
};
```

```
void setLengthWidth(rectangle r){
    r.length = 5.0;
    r.width = 6.0;
}

void getArea(rectangle r){
    cout<<r.height*r.width<<endl;
}
```



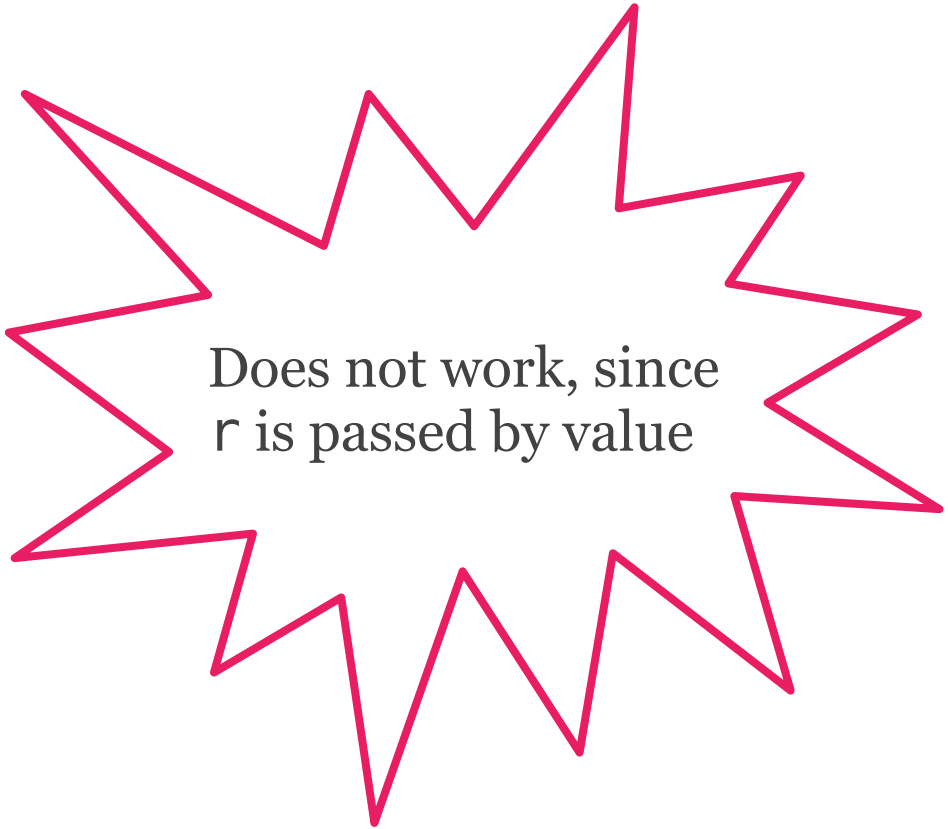
```
int main(){

    rectangle r;

    setLengthWidth(r);

    getArea(r);

    return 0;
}
```



1a.structs and functions using static memory

```
//Single Element
#include <iostream>
using namespace std;

struct rectangle{
  float height;
  float width;
};
```

```
void setHeight(rectangle &r){
    r.length = 5.0;
    r.width = 6.0;
}

void getArea(rectangle &r){
    cout<<r.height*r.width<<endl;
}
```

```
int main(){

    rectangle r;

    setLengthWidth(r);

    getArea(r);

    return 0;
}
```

1b.structs and functions using static memory for multiple elements

```
//Multiple Element
#include <iostream>
using namespace std;

struct rectangle{
  float length;
  float width;
};
```

```
void setLengthWidth(rectangle r[],int n){
    for(int i = 0; i<n; ++i){
        r[i].height = 5.0;
        r[i].width=4.0;
    }
}

void getArea(rectangle r[], int n) {
    for(int i=0; i<n; ++i){
    cout<<r[i].height*r[i].width<<endl;}
}
```

```
int main(){

  rectangle r[3];

  setLengthWidth(r,3);

  getArea(r,3);

  return 0;
}
```

2a.structs and functions using dynamic memory

```
//Single Element
#include <iostream>
using namespace std;

struct rectangle{
  float length;
  float width;
};
```

```
int main(){

    rectangle *r = new rectangle;

    setLengthWidth(r);

    getArea(r);

    return 0;
}
```

```
void setLengthWidth(rectangle *r){
    r->length = 5.0;
    r->width = 4.0;
    //(*r).length = 5.0;
    //(*r).width = 4.0;
}

void getArea(rectangle *r) {
    cout<<r->length*r->width<<endl;
    //cout<<(*r).length*(*r).width<<endl;
}
```

2b.structs and functions using dynamic memory for multiple elements

```
//Multiple Element
#include <iostream>
using namespace std;

struct rectangle{
  float length;
  float width;
};
```

```
int main(){

   rectangle *r = new rectangle[3];

   setLengthWidth(r,3);

   getArea(r,3);

   return 0;
}
```

```
void setLengthWidth(rectangle *r, int n){
   for(int i =0; i<n; ++i){
      r[i].length = 5;
      r[i].width = 6;}
}

void getArea(rectangle *r, int n) {
   for(int i =0; i<n; ++i)
      cout<<r[i].length*r[i].width<<endl;
}
```

Summary

1. Static Memory

    a. structs using function for single element

```
student s; s.name = “hello”; s.age=12;
```

    b. structs using function for multiple element

```
fruit f[3];
f[0].name = “banana”;  f[0].age=10;
f[1].name = “apple”;   f[1].age=20;
f[2].name = “pineapple;f[2].age=30;
```

Summary

2.Dynamic Memory

a.structs using function for single element

```
student s = new student;
s->name = “motiur”; s->age=12;
```

b.structs using function for multiple element

```
fruit *f = new fruit[3];
f[0].name = “banana”;   f[0].age=10;
f[1].name = “mango”;    f[1].age=20;
f[2].name = “pineapple”;f[2].age=30;
```