Slide Set 2

Constructor and Destructor: Constructors, Parameterized Constructors, Constructor Overloading, Constructors in array of objects, Constructors with default arguments, Dynamic Initialization, Pointer to objects, this pointer, Dynamic memory allocation, Array of pointer to objects, Copy Constructor, Static objects, Friend function, and Friend classes.

Constructors

- It is a public function with the same name as class
- Used to initialize the values of object variables
- Called automatically
- No return

Example - constructor

```
#include<iostream>
using namespace std;
class A{
public:
A(){cout<<"In constructor";}
};
int main(){A a; }
```

Parameterized constructor

```
class A{
int i;
public: A(int a){i=a;}
void show()  {  cout<<"i = "<< i << endl;  }
};
int main(){
A a(19);
a.show(); }
```

Constructor Overloading

```
class A{
int i;
public:
A(){i=0;}
A(int a){i=a;}
void show(){cout<<"i = "<<i<<endl;}
};
int main(){
A a(19),b;
a.show();
b.show();}
```

Constructors in array of objects

```
#include<iostream>
using namespace std;
class A{
public: A(){cout<<"In constructor"<<endl;}
};
int main(){
A a[5];
}
```

Constructors with default arguments

```
class A{
int i,j;
public:
A(int a=-99, int b=-88){ i=a; j=b;}
void show(){cout<<"i="<<i<<" j = "<<j<<endl;}
};
int main(){
A a, b(22),c(1,2);
a.show(); b.show(); c.show();
}
```

Dynamic Initialization

```
int i=5;
char name[i];
```

OR

```
int const b = factorial(8);
```

Pointer to object

```
class Test {
int i;
public: Test(int a){i=a;}
void show(){cout<<"i = "<<i<<endl;}
};
int main() {
Test t(10);  //object created
Test *p;   // just a pointer, obj not created
p = &t;  p->show();   // OR t.show();
}
```

this pointer is the address to current object

```
class Test {
int x;
public: Test(int x) { this->x = x; }
void show(){ cout << "x = " << x << endl; }
};
int main() {
Test obj(20);
obj.show();
}
```

Dynamic memory management

```
int main() {
int *i = new int;   // allocate one int space
cout<<"Enter integer value: ";
cin>>*i;
cout << endl << "*i = " << *i << endl;
delete i;   // clear int space
}
```

Dynamic memory allocation

```
int main() {
char *st = new char[20];   // array of 20 characters
cout<<"Enter string less than 20 char: ";
cin.getline(st, 20);
cout<<endl<<"You entered: "<<st<<endl;
delete []st;
}
```

Dynamic memory allocation

```
int main() {
int n;
cout<<"How many char?"; cin>>n; cin.sync();
char *st = new char[n];
cout<<"\nEnter string of atmost n char: ";
cin.getline(st, n);
cout<<endl<<"You entered: "<<st<<endl;
delete []st;
}
```

Array of pointer to objects

```
class Test {
int i;
public: void setData(int a){i=a;}
void show() {cout<<"i = "<<i<<endl;}
};
int main() {
Test *t = new Test[2];    // create array of 2 objects
t->setData(44); (t+1)->setData(88); // set both objects
 t->show(); (t+1)->show();  //show both objects
delete []t;   //delete both objects
}
```

Copy constructor

```
class Test {
int i;
public:
Test(int a){i=a;}
Test(Test & t){i=t.i;}    // & is used in the syntax of copy constructor
void show(){cout<<"i = "<<i<<endl;}
};
int main() {Test t(99),u(t); u.show();}
```

What is the output?

```
class Test  {
public: Test() {cout << "In constructor"<<endl; }
    ~Test() { cout << "In destructor"<<endl; }
};
void myfunc() {  Test obj; }
int main() {
cout << "Start main()"<<endl;   myfunc();
cout << "End main()"<<endl;
}
```

Answer:

Start main()

In constructor

In destructor

End main()

Now using static object

```
class Test  {
public: Test() {cout << "In constructor"<<endl; }
    ~Test() { cout << "In destructor"<<endl; }
};
void myfunc() {  static Test obj; }
int main() {
cout << "Start main()"<<endl;   myfunc();
cout << "End main()"<<endl;
}
```

Answer: Static objects never die easily

Start main()

In constructor

End main()

In destructor

Friend function can access private data

```
class Test {
int i;
public: Test(){i=12;}
friend void show(Test t);
};
void show(Test t){cout<<"i="<<t.i<<endl;}
int main(){
Test t; show(t);
}
```

Friend classes

```
class A{
int i;
public: A(){i=5;}
friend class B;
};
class B{
public: void show(A a){cout<<"i="<<a.i<<endl;}
};
int main(){A a; B b; b.show(a);}
```