CLASS WORK

Constructors, Copy Constructors, constructor overloading, function overloading

Lecture 4

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

What is a constructor?

- It is a member function which initializes a class.
- A constructor has:

  (i) the same name as the class itself

  (ii) no return type

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

Comments on constructors

- A constructor is called automatically whenever a new instance of a class is created.
- You must supply the arguments to the constructor when a new instance is created.
- If you do not specify a constructor, the compiler generates a default constructor for you (expects no parameters and has an empty body).

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

What is a copy constructor?

- It is a member function which initializes an object using another object of the same class.
- A copy constructor has the following general function prototype:

class_name (const class_name);

- It is an another way to initialize an object: you can initialize it with *another object of the same type*.
- We don't need to create a special constructor for this; one is already built into all classes.
- It's called the *default copy constructor*.
- It's a one argument constructor whose argument is an object of the same class as the constructor.

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

Copy Constructor Example

```
#include<iostream.h>
#include<conio.h>
//------------------------------------------------------------------------
//copy constructor class test
class copyconst
 {
 private:
 int a;
 public:
       copyconst(){   }
       copyconst(int  x)  {
       a=x;
       }
       void disp_sq() {
       cout<<"\nSquare of "<<a<<"="<<a*a<<endl;
       }
       ~copyconst() {
       cout<<"destructor"<<endl;}
       };
```

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

CLASS WORK

```
int main()
{
copyconst obj(5);
copyconst obj2(obj);
copyconst obj3=obj2;
obj.disp_sq();
obj2.disp_sq();
obj3.disp_sq();
system("PAUSE");
	return 0;
}
```

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

Constructor Overloading

Constructor overloading refers to the use of more than one constructor with same name but different number of arguments. Each constructor is called according to its signature matching

Example program:

```
#include<iostream.h>
#include<conio.h>
class constructor
{
private:
```

```
int x;
public:
        constructor()
        {
                x=0;
        }
        constructor(int  a)
        {
                x=a;
        }
        constructor(float a)
        {
                x=a;
        }
```

CLASS WORK

Tabir Nawaz
Assistant Professor
UOL Sargodha Campus

```
void input(int y)
{
        x=y;
}
void display()
{
cout<<x<<"\n";
}
};
```

CLASS WORK

```
int main(void){
constructor obj1;
constructor obj2(10);
Constructor obj3(4.5);
cout<<“After invoking constructors value of x for obj1 and obj2”;
obj1.display();
obj2.display();
Obj3.display();
cout<<“After invoking input() value of x is for obj1 and obj2”;
obj1.input(50);
obj2.input(80);
Obj3.input(50.5);
obj1.display();
obj2.display();
Obj3.display();
getche();
Return 0;
}
```

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

CLASS WORK

Lab Task

- Write a program which input records of ten employees and then to print it out on the screen. The task should only be done by using constructors.
- Write a class using constructors to find the factorial of any given integer value.

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

Functions Overloading

- C++ enables several functions of the same name to be defined, as long as they have different signatures. This is called function overloading.
- The C++ compiler selects the proper function to call by examining the number, types and order of the arguments in the call.
- Function overloading is used to create several functions of the *same* name that perform similar tasks, but on *different* data types.

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

- Overloaded functions are distinguished by their signatures.
- A signature is a combination of a function's name and its parameter types (in order). The compiler encodes each function identifier with the number and types of its parameters.

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

Example of overloaded Function without classes

```
#include <iostream.h>
 using namespace std;
int square(int  x )
{
cout << "square of integer " << x << " is ";
return x * x;
}
float square( float y )
{
cout << “Square of float " << y << " is ";
return y * y;
}
 int main(void)
 {
 cout << ;
 cout << square(7)<<endl;
 cout << square(7.5)<<endl;
 }
```

Function overloading example in a class.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
//------------------------------------------------------------------------
class ovrlod
{
private:
int a;
float b,d;
string c;
public:
void get_data(int x)
{
a=x;
}
```

```
void get_data(float y,float z)
{
b=y;
d=z;
}
void get_data(string z)
{
c=z;
}
void display_sq()
{
cout<<"square of "<<a<<"="<<a*a<<endl;
}
void display_float()
{
cout<<"add float values "<<b<<"+"<<d<<"="<<b+d<<endl;
}
```

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

```
void display_char()
{
cout<<"entered char "<<c<<endl;
}
};
```

CLASS WORK

```
int main()
{
   ovrlod obj;
   obj.get_data(5);
   obj.get_data(5.5,5.5);
   obj.get_data("Tahir");
   obj.display_sq();
   obj.display_float();
   obj.display_char();
   getch();
   return 0;
}
```

CLASS WORK

`static` Class Data

- `static` data member
  - Only one copy of a variable shared by all objects of a class
    - "Class-wide" information
    - A property of the class shared by all instances, not a property of a specific object of the class
  - Declaration begins with keyword `static`

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

`static` Class Members

- `static` member function
  - Is a service of the *class*, not of a specific object of the class
- `static` applied to an item at file scope
  - That item becomes known only in that file
  - The `static` members of the class need to be available from any client code that accesses the file
    - So we cannot declare them `static` in the `.cpp` file—we declare them `static` only in the `.h` file.

CLASS WORK

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

`static` Class Members

- Use `static` data members to save storage when a single copy of the data for all objects of a class will suffice.
- A class's `static` data members and `static` member functions exist and can be used even if no objects of that class have been instantiated.

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus

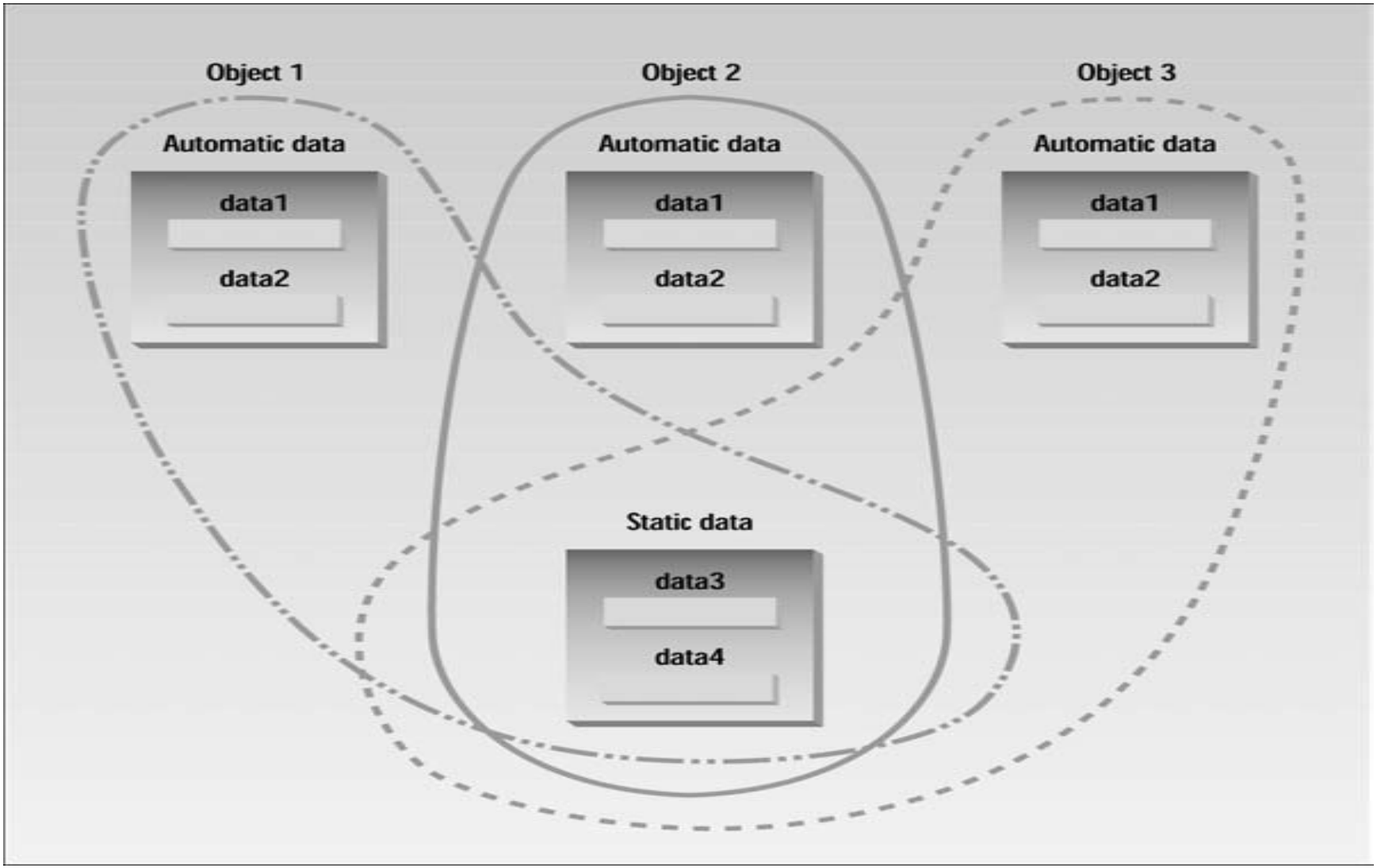
An Example of Static Class Data

Here's an example, STATDATA, that demonstrates a simple static data member:

```
// statdata.cpp
// static class data
#include <iostream>
using namespace std;
class sta
{
private:
static int count;
```

```
public:
sta()
{ count++; }
int getcount()
{ return count; }
};
//--------------------------------------------------------
int sta::count = 0;
////////////////////////////////////////////////////////////
int main(void)
{
Sta f1, f2, f3; //create three objects
cout << “count is “ << f1.getcount() << endl;
cout << “count is “ << f2.getcount() << endl;
cout << “count is “ << f3.getcount() << endl;
return 0;
}
```


Jabir Nawaz
Assistant Professor
UOL Sargodha Campus



- *Static Versus automatic member variables*

CLASS WORK

Assignment 1

- Write a program in C++ by using a class to copy one string into another string and also to find out the length of the string without using any built-in function.
- Write a class which can convert temperatures. Such as if we provide Centigrade to Fahrenheit converter or Fahrenheit to Centigrade without using built in functions.

Submission Date:18/12/2013

Time 1:00pm to 1:30pm

Note: No late submission will be accepted

Jabir Nawaz
Assistant Professor
UOL Sargodha Campus