Lecture-3

- Concepts of class/object
- Data encapsulation
- Constructor and destructor
- public, private and protected members
- friend functions

# C++ — Philosophically different from C

- **High level features of C++**
  - Uses concepts of "object oriented programming" (OOP)
  - Everything that works in C works in C++
    - C syntax, operators, structures, control statements, etc. work in C++
    - Reverse is NOT true
- **Object Oriented Programming**
  - Concept of class/object, methods, inheritance, encapsulation, abstraction, polymorphism
  - Key concepts in this
   - Separation of data and methods

Data encapsulation

- Provide access restrictions to member data and functions

    - From other classes and functions.

- Implemented y using access modifiers

    - **public**, **private** and **protected**

- Other classes, functions need to know what methods are implemented

    - Not how they are implemented

Data encapsulation ... contd.

User

Hidden from user

Data → **Method** → **Output**
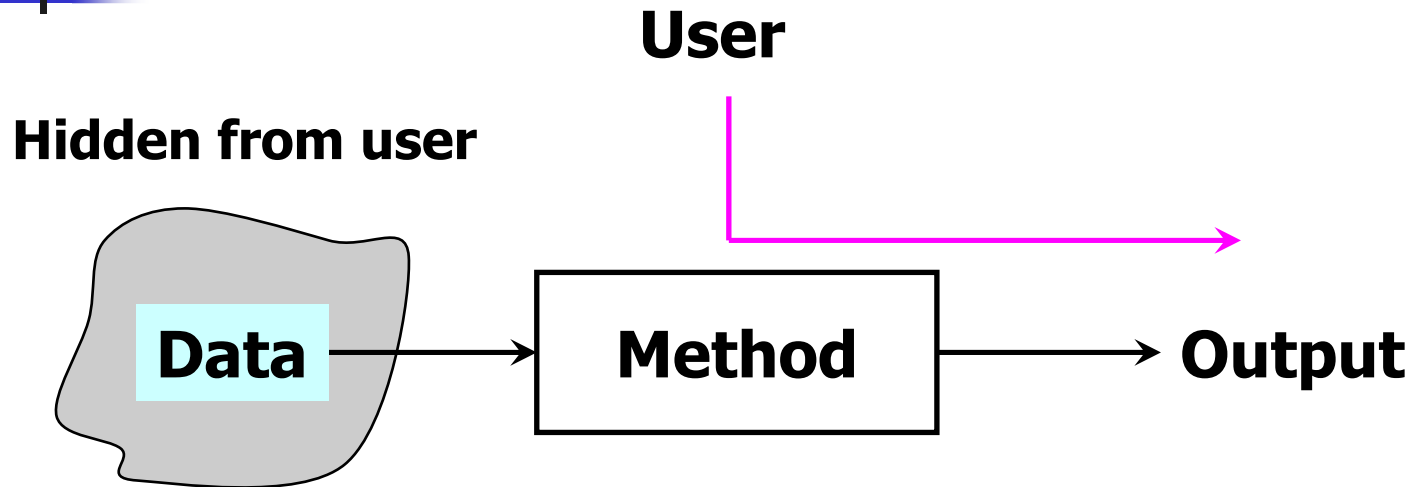
- Methods act on data to provide output.
- User needs to see only method, not data.
- User should not be affected by
  - Implementation details of methods.
  - Changes in implementation of methods.
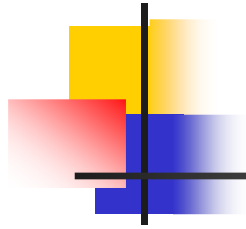
A simple "account" example

```
class account
{
  private:
      int user_SSN;              // data
      int accountNumber;         // data
  public:
      void withdrawMoney (int amount);   // method
      void depositMoney (int amount);    // method:
      void computeInterest( );           // method
};
account x;  // x is an object of class "account"
```

Account example … contd.

- class has both "data" and "methods".
- Attributes and methods are "members" of a class
- An instance of a class is an object.
- A class should typically correspond to some meaningful entity.
- A class uses methods to interact with other classes/functions.
- private members accessible only to the class (and friends)
- public members are accessible to every class and functions

Back to data encapsulation

- How can data be hidden?
  - Only class should have access to data
  - Class methods use data
- Define every class member to be one of
  - public – accessible to every class, function
  - private – accessible only to class and friends
  - protected – accessible only to class, friends and children

Data encapsulation in account example

- ## In an object of account
  - user_ssn and accountNumber are declared private
    - Accessible only to account objects (and friends)
  - Methods are public
    - Anyone can access them.

- ## Example

```
void function1 ( ) // function, not defined in Account
{
        account x;
        x.user_ssn = 123; // Will NOT work
        x.computeInterest ( ); // Will work
}
```

How do we initialize and cleanup objects?

```cpp
class account
{
    private:
        int user_SSN;
        int accountNumber;
    public:
        account( ); // constructor – to initialize account object
        account(int ssn, int acctNum); // constructor
        ~account( );    // destructor – used to cleanup resources
        void withdrawMoney (int amount);
        void depositMoney (int amount);
};
```

Constructor and destructor ... contd.

Constructor

- o A function with the same name as the class
- o Called when an object is created
- o A class can have more than one constructor

Destructor

- o Called when an object is cleaned up (goes out of scope)
- o One class can have only one destructor

Examples

account x; // constructor code is called

account *y = new account; // constructor code is called

delete (y); // destructor code is called

Constructor and destructor

Constructor code

```
account::account( )
{ user_ssn = -1; accountNumber = -1; }


account::account( ) : user_ssn (-1),
                         accountNumber(-1) { }
account::account (int ssn, int acctNum)
{
     user_ssn = ssn;
     accountNumber = acctNum;
}
```

Destructor code

```
account::~account( )
{  // Any memory/resource cleanup, etc. }
```

Class methods

Syntax:

```
<ret_type> class::functionName(args)
{
    // code
}
```

Method code can be present in class definition
- Outside the class definition
- In a separate file

Example

```
void account::withdrawMoney (int amount)
{
    // code
}
```

friend functions

- What if a function genuinely needs to have access to private data?
    - E.g. showAccountInfo (Account acct)
- Need to give access ONLY to that function, not others.
- Use friend function definition
- friend functions of a class have access to private members of the class.

Example – friend function

```
class account
{
private:
    int user_SSN;
    int accountNumber;
public:
    void deposit (int amount)
    void withdraw (int amount);
friend showAccountInfo
    (class Account)
};
```

```
void showAccountInfo
    (Account acct)
{
    cout << user_SSN << endl;
    cout << accountNumber <<
                endl;
}
```

This is valid.

Friend function can access private members.

friend class

- Concept of friend can be extended to a class from a function.
- A class gives access to its private members to its friend classes.

```
class account                 class bank
{                             {
 …                                          …
    friend class bank         }
}
```

Members of bank have access to private members of account

Examples

- Valid usage in an external function
  - account acct(123456, 5672);
  - checkingAccount ca;
  - acct.deposit (700);
  - acct.withdraw (300);
  - ca.deposit (1000);
  - ca.showAllChecksCleared( )
- Invalid usage in derived class
  - ca.user_SSN = 1234; // Can't access user_SSN
  - ca.accountNumber = 567;