**Chapter 13**

**Basic Classes**

**Overview**

* class: like a struct (allows bundling of related variables), but variables and functions in the class can have different properties than in a struct
* object: an instance of a class, in the same way that a variable can be an instance of a struct
* attributes: members of a class (identical to struct)
* methods or behaviors: member functions of a class - integration of code/data

What are the advantages classes provide over structured programming (e.g. struct and functions)?

Information Hiding

Encapsulation (Private data/Public Methods)

Inheritance

Run-time Polymorphism

**Step 1. Class Declaration**

The declaration for a class appears similar to that of a structure, but there are major differences. Consider the declaration from example9.cpp:

class Employee

{

private: //These are the private data fields

int id;

int dept\_id;

int yrs\_svc;

public: //Prototypes for methods

Employee(); //Default initializer - called *constructor*

void setId(int id); //setter

int getId(); //getter

void setDeptID(int dept\_id);

int getDeptId();

int getYrsSvc();

bool detSvcAward();

};

Private data can only be accessed within the methods. In other words, you cannot access the private data using the dot operator in functions not related to the class - like **main**. Consider the following main (from example9.cpp)

int main()

{

//Variable Declaration/Initialization (Default syntax identical to struct)

Employee emp1;

Employee emp2;

//emp1.id = 9; Will not compile now

//cout << emp1.id << endl; Will not compile either

*The central principal of information hiding is that private data is never accessed outside the class - they can only be accessed within the methods of the class.*

Information Hiding allows for maintainability - changes to class are only made **within** the methods of the class - not **everywhere** the class happens to be used!

We will access the private data by using setters/getters methods:

Setters set the private data

Getters get the private data

We re-write the above main using

int main()

{

//Variable Declaration/Initialization (Default syntax identical to struct)

Employee emp1;

Employee emp2;

emp1.setId(9);

cout << emp1.getId() << endl;

**Methods are similar to functions, but can automatically access all private data fields of the class.**  At a **minimum**, every class must have the following methods:

Default Constructor - We’ll discuss this in more detail on Monday

Setter for each data field

Getter for each data field

**What is the purpose of the Setter/Getter**

The reason for using setters/getters is so we don’t access the data fields outside the class by the dot (.) operator. They provide information hiding. Any underlying changes to the data structure can be changed in only one place - these methods - instead of 1000’s of lines of code by accessing the data fields directly!