CSE-281: Data Structures and Algorithms

Classes and Objects

Ref. Book: *Object Oriented Programming, C++*
By E. Balagurusamy

Introduction

- ✓ The most important feature of C++ is the **"class".**
- ✓ A class is an extension of the idea of **structure** used in C.
- ✓ It is a new way of creating and implementing a **user-defined data type**.

C Structures Revisited

- ✓ A structure is a convenient tool for handling a group of logically related data items.
- ✓ It is a user-defined data type with a template that serves to define its data properties.
- ✓ Consider the following declaration:

```
struct student
{
    char    name [20] ;
    int     roll_number;
    float   total_marks;
 };
```

Continue….

- ✓ These fields are known as structure members or elements.
- ✓ The identifier **student**, which is referred to as structure name or structure tag, can be used to create variables of type student:

```
struct student A;          // C declaration
```

- ✓ Member variables can be accessed using the dot or period operator as follows:

```
strcpy(A.name, "John");
A.roll_number = 999;
```

- ✓ Structures can have arrays, pointers or structures as members.

.

Limitation of C Structures

- ✓ The standard C does not allow the struct data type to be treated like built-in types.
- ✓ consider following structure:

```
struct complex
          {
                    float x;
                    float y;
          };
     struct complex c1, c2, c3;
```

- ✓ We cannot add two complex numbers or subtract one from the other. For example,

  `c3 = c1 + c2;    //illegal in C.`

- ✓ They do not permit data hiding. In other words, the structure members are public members.

Extensions to Structures

- ✓ C++ supports all the features of structures as defined in C.
- ✓ It attempts to bring the user-defined types as close as possible to the built-in data types, and also provides a facility to hide the data which is one of the main precepts of OOP.
- ✓ Inheritance, a mechanism by which one type can inherit characteristics from other types, is also supported by C++.
- ✓ In C++, a structure can have both variables and functions as members.
- ✓ It can also declare some of its members as 'private' so that they cannot be accessed directly by the external function.

Continue….

- ✓ In C++, the structure names are stand-alone and can be used like any other type names:

  student A;

- ✓ C++ incorporates all these extensions in another user-defined type known as **class**.
- ✓ The only difference between a structure and a class in C++ is that, by default, the members of a class are private, while, by default, the members of a structure are public.

Specifying a Class

- ✓ A class is a way to bind the data and its associated functions together.
- ✓ It allows the data (and functions) to be hidden, if necessary, from external use.
- ✓ When defining a class, we are creating a new abstract data type that can be treated like any other built-in data type.
- ✓ Generally, specification has two parts:

  1. Class declaration

  2. Class function definitions

- ✓ The class declaration describes the type and scope of its members.
- ✓ The class function definition describe how the class functions are implemented.

Continue….

- The general form of a class declaration is:

```
class class_name
{
        private:
                variable declarations;
                function declarations;
        public:
                variable declarations;
                function declarations;
};
```

Continue….

- ✓ The keywords **private** and **public** are known as visibility labels followed by a colon.
- ✓ Private members can be accessed only from within the class.
- ✓ Public members can be accessed from outside the class also.
- ✓ The use of keyword private is optional. By default, the members of a class are private.
- ✓ If both the labels are missing, then, by default, all the members are private.
- ✓ The variables declared inside the class are known as **data members** and the functions are known as **member functions**.

Continue….

- ✓ Only the member functions can have access to the private data and private functions.
- ✓ the public members (both functions and data) can be accessed from outside the class.
- ✓ The binding of data and functions together into a single class-type variable is referred to as **encapsulation**.
- ✓ This is illustrated in next Figure:

Continue….

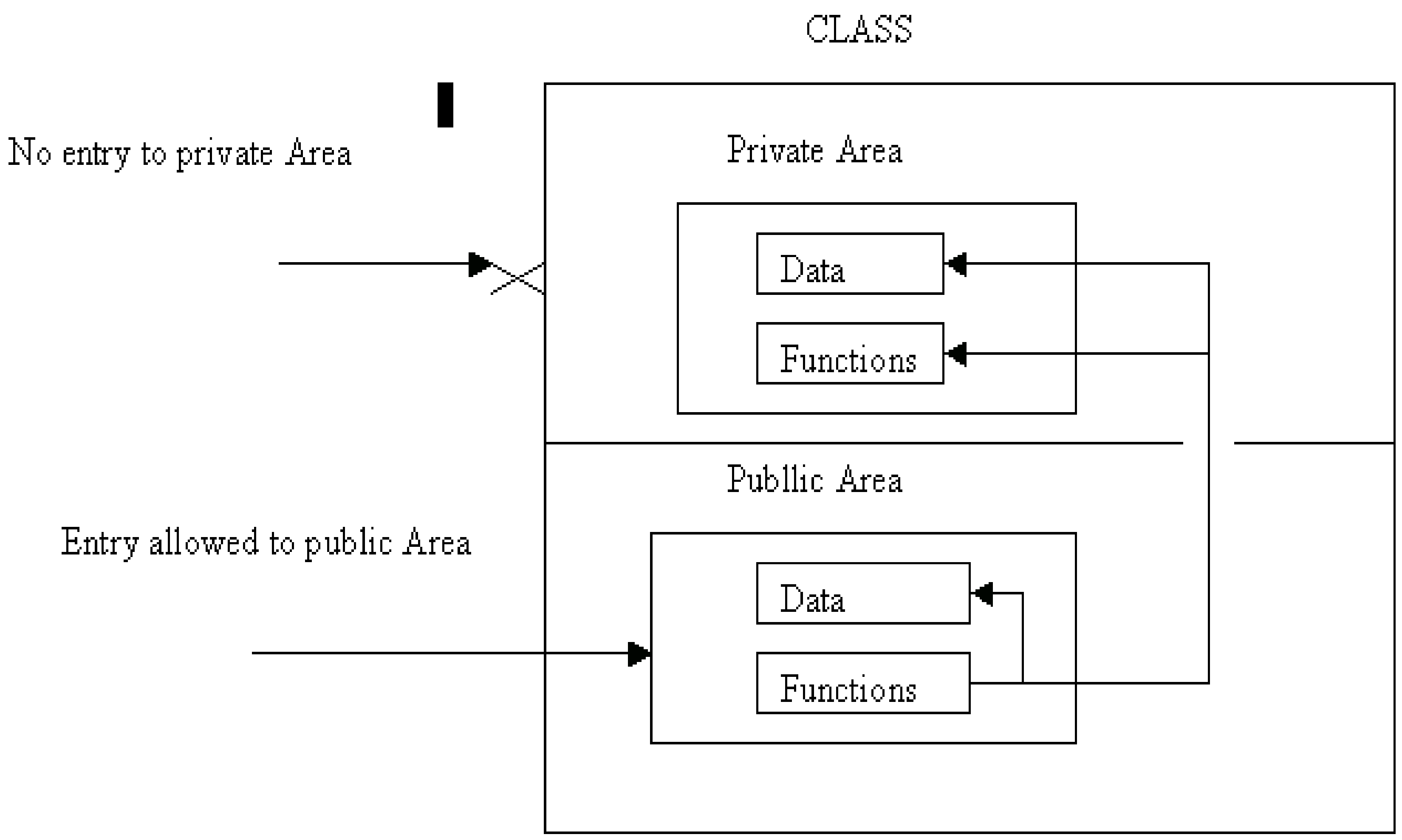


Fig.- Data hiding in classes

A Simple Class Example

- A typical class declaration would look like:

```
class item
 {
    int number;      // variables declaration
    float cost;      //private by default
  public:
    void getdata (int a, float b);//functions declaration
    void putdata (void);          // using prototype
 };
```

- The functions are declared, not defined. Actual function definition will appear later in the program.
- The data members are usually declared as private and the member functions as public.

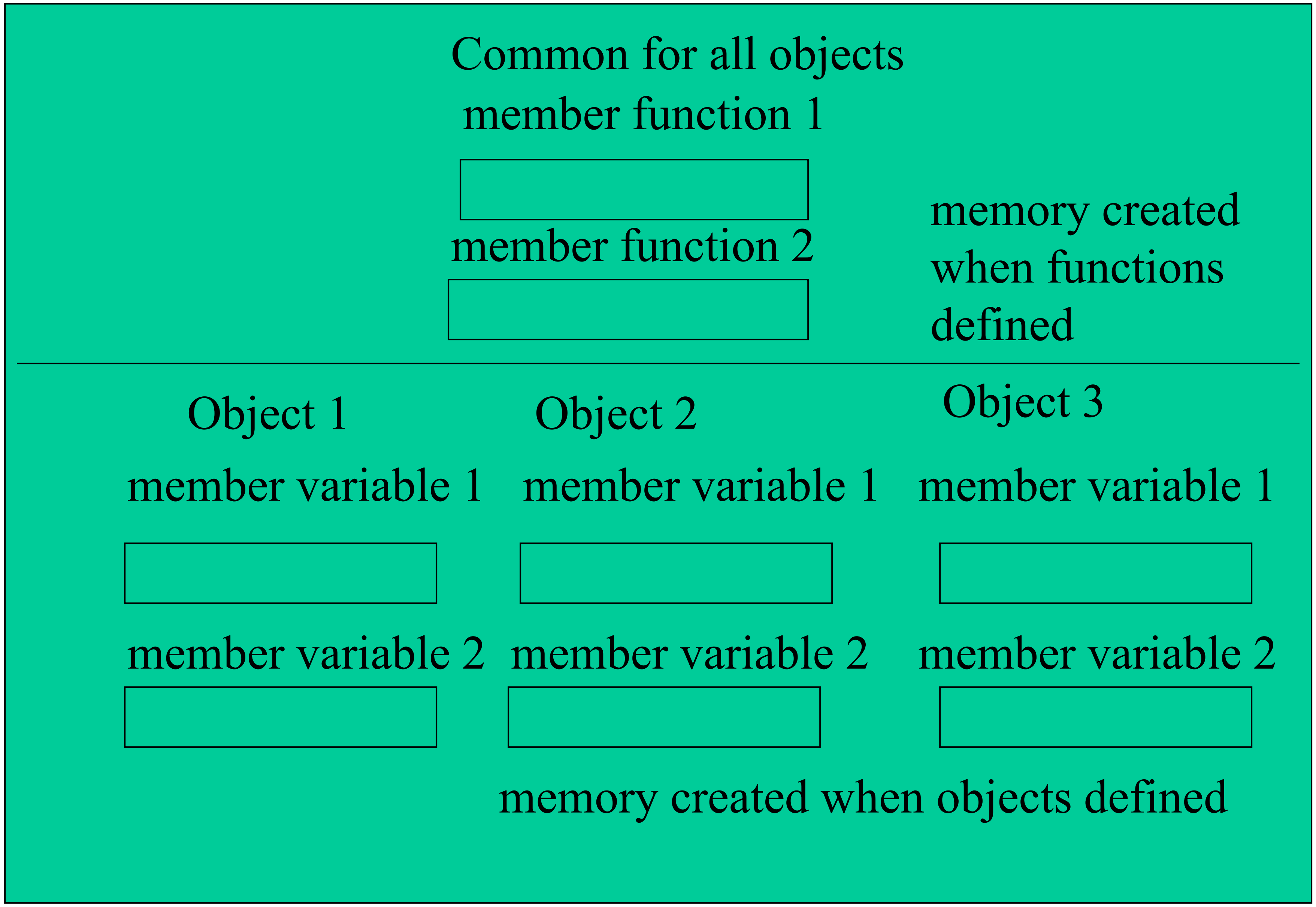
Creating Objects

- The declaration of item only specifies what they will contain.
- Once a class has been declared, we can create variables of that type by using the class name (like any other built-in type variable). For example:

```
item x;     // memory for x is created
```

- In C++, the class variables are known as **objects**. Therefore, **x** is called an object of type **item**.
- The necessary memory space is allocated to an object at this stage.
- Class specification, like a structure, provides only a template and does not create any memory space for the objects.

Objects in memory



Continue….

- Objects are also be created when a class is defined by placing their names immediately after the closing brace, as we do in the case of structures:

```
class item
        {
        ..........
        ..........
        } x, y, z;
```

Accessing Class Members

- ✓ The main () cannot contain statements that access number and cost.
- ✓ The format for calling a member function:
- ✓ **object_name.function_name(actual_arguments);**
- ✓ For example,

```
x.getdata(100, 75.5);
x.putdata( );
```

- ✓ The assignments occur in the actual function.
- ✓ A member function can be invoked only by the object (of the same class).

Continue….

- The statement like

```
getdata(100, 75.5);
```

  has no meaning.
- It may be recalled that objects communicate by sending and receiving messages. This is achieved through the member functions. For example:

```
x.putdata();
```

- A variable declared as **public** can be accessed by the objects directly.

Defining Member Functions

- Member functions can be defined in two places:
    - **Outside the class definition.**
    - **Inside the class definition.**
- Irrespective of the place of definition, the function should perform the same task.
- The code for the function body would be identical in both the case.
- However, there is a subtle difference in the way the function header is defined.

Outside the Class Definition

- ✓ Member functions that are declared inside a class have to be defined separately outside the class.
- ✓ They should have a function header and a function body.
- ✓ An important difference between a member function and a normal function is that a member function incorporates a membership 'identity label' in the header.
- ✓ This 'label' tells the compiler which class the function belongs to.

Continue….

- The general form of a member function definition is:

```
return-type class-name :: function-name (argument declaration)
        {
                function body
        }
```

- The scope of the function is restricted to the class-name in the header line.
- Example:

```
void item : : getdata (int a, float b)
{
    number = a;
    cost  = b;
}
```

```
void item : : putdata (void)
{
 cout<<number <<"\n";
 cout<<cost  <<"\n";
}
```

Continue….

- Some special characteristics of Member functions are
  - Several different classes can use the same function name. The 'membership label' will resolve their scope.
  - Member functions can access the private data of the class. A non-member function cannot do so.
  - A member function can call another member function directly, without using the dot operator.

Inside the Class Definition

- Another method of defining a member function is to replace the function declaration by the actual function definition inside the class:

```
class item
{
        int number;
        float cost;
    public:
        void getdata(int a, float b);   //declaration
           // inline function
        void putdata(void)   //definition inside the class
           {
                cout<<number<<"\n";
                cout<<cost<<"\n";
           }
}
```

Continue….

- ✓ When a function is defined inside a class, it is treated as an inline function.
- ✓ Therefore the restrictions and limitations that apply to an inline function are also applicable here.
- ✓ Normally, only small functions are defined inside the class definition.

- A C++ program with class: Self

Thank You