innovative ● entrepreneurial ● global

ONLINE LEARNING

Introduction to Abstract Data Type & C++

SCSJ2013 Data Structures & Algorithms

Nor Bahiah Hj Ahmad & Dayang Norhayati A. Jawawi

Faculty of Computing

Objectives

In this lesson, students are expected to:

- Understand Abstract Data Type concept
- Master C++ programming
  - Class declaration
  - Creating constructor and destructor
  - Pass object as function parameter
  - Return object from a function.
  - Array of class

What is Abstraction?

- Make calls and receive calls
- Take photos
- Send and receive messages
- Access internet

Abstraction

Functional abstraction

- The purpose of a module is separated from its implementation
- Separates the purpose of a module from its implementation

Data Abstraction

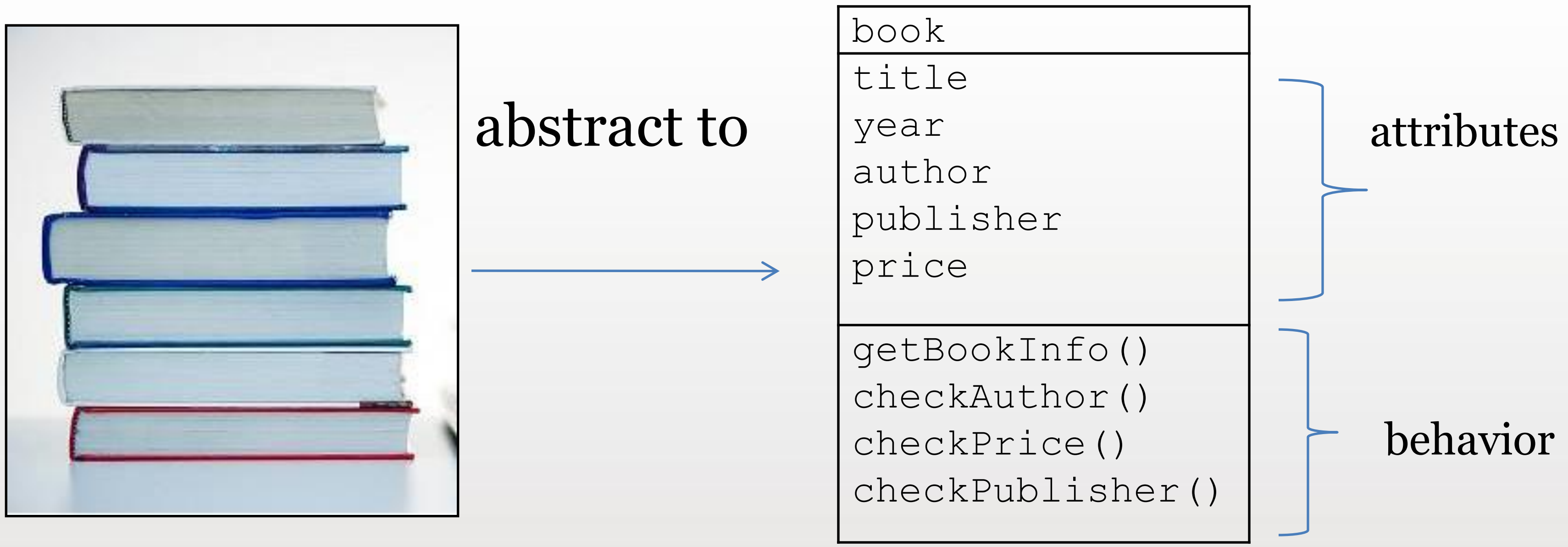
- Focuses on the operations of data (*what* you can do to a collection of data)
- And not on the implementation of the operations (*how* you do it)
- Develop each data structure independently from the rest of the solution

Abstract Data Type

Abstract data type (ADT)

- A collection of data and a set of operations on the data
- Given the operations' specifications, the ADT's operations can be used without knowing their implementations or how data is stored,

Abstract Data Type - Example



Abstraction of a book

Encapsulation

- The process of combining data and functions into a single unit called class.
- The programmer cannot directly access the data. Data is only accessible through the functions present inside the class.
- Data encapsulation is an important concept of data hiding.

Abstraction Implementation in C++

- In C++, class defines a new data type
- In a class there are data members and methods, which are called member functions.
- By default, all members in a class are private
  - But it can also be specified as public
- An object of the datatype is an instance of a class.

How to define a Class in C++?

```
class clasName
{
public:
    list of data member declaration;
    list of function member declaration;
private:
    list of data member declaration;
    list of function member declaration;
};  // end class definition
```

$\}$ class member declarations: data member and function member

public : members that are accessible by other modules

private : members that are hidden from other modules and can only be accessed by function member of the same class.

Class Definition for Book

```
class book
{ private:
  // data member declaration as private
    float price;
    int year;
    char author[20], title[25];
  public:
    book();        // Default constructor
    // Constructor with parmeter
    book(char *bkTitle,double bkPrice);
    book(int = 2000);
    // C++ function
    void getData();
    void print( );
    float checkPrice( )const;
    char * getAuthor();
    ~book() ;   // destructor
};  // end book declaration
```

Attribute declarations

Constructor

Function Member Declaration

Destructor

Constructors

- Constructors
  - Used to create and initialize new instances of a class
  - Is invoked when an instance of a class is declared
  - Have the same name as the class
  - Have no return type, not even `void`
- A class can have several constructors
  - However, compiler will generate a default constructor if no constructor is defined.

Destructor

- Destroys an instance of an object when the object's lifetime ends
- Each class has one destructor
  - The compiler will generate a destructor if the destructor is not defined
- Example: `~book();`

```
book::~book()
{ cout << "\nDestroy the book with title "
          << title;
}
```

Function Member Implementation

```
void book::getData()
{ cout << "\nEnter author's name : ";
  cin >> author;
  cout << "\nEnter book title : ";
  cin >> title;
}
```

How to call the member function?

- You can call the function from `main()` or non-member function:

```
book myBook;
cout << myBook.getData() << endl;
```

const member function – cannot alter value

```
float book::checkPrice( )const
{   return price;   }
```

Classes as Function Parameters

- Class objects can be passed to another function as parameters
- 3 methods of passing class as parameter to function
  - Pass by value
  - Pass by reference
  - Pass by const reference

Passing a class object by Value

Any change that the function makes to the object is not reflected in the corresponding actual argument in the calling function.

Pass by value

```
class subject
{
private:
     char subjectName[20];
     char kod[8];
     int credit;
public:
     subject (char *,char *,int k=3);
     void getDetail();
     friend void changeSubject(subject);
};
subject:: subject (char *sub,char *kd,int kre)
{    strcpy(subjectName,sub);
     strcpy(kod,kd);
     credit = kre;
}
void subject:: getDetail()
{
cout << "\n\nSubject Name  :  " << subjectName;
cout << "\nSubject Code   :  " << kod;
cout << "\nCredit hours   :  " << credit;
}
```

friend function is used to pass object as parameter and allow non-member function to access private member.

Pass by value Continued…

```
// friend function implementation that receive object as
parameter
void changeSubject(subject sub); // receive object sub
{ cout << "\nInsert new subject name: ";
  cin >> sub.subjectName;
  cout << "\nInsert new subject code: ";
  cin >> sub.kod;
  cout << "\n Get new information for the subject.";
  sub. getDetail();
}
main()
{ subject DS("Data Structure C++","SCJ2013");
  DS.getDetail();
  changeSubject(DS); // pass object DS by value
  cout << "\n View the subject information again: ";
  DS.getDetail();  // the initial value does not change
  getch();
};
```

Access class member, including private data member from sub.

Pass by reference

- Any changes that the function makes to the object will change the corresponding actual argument in the calling function.
- Function prototype for function that receive a reference object as parameter: use operator &

```
functionType functionName(className & classObject)
{
      // body of the function
{
```

Pass by Reference

```
// pass by reference
// friend function that receive object as parameter
void changeSubject(subject &sub);  // operator & is used
{ cout << "\nInsert new subject name: ";
  cin >> sub. subjectName;
  cout << "\nInsert new subject code: ";
  cin >> sub.kod;
  cout << "\n Get new information for the subject.";
  sub. getDetail();
}
main()
{ subject DS("Data Structure C++","SCJ2013");
  DS.getDetail();
  changeSubject(DS); // pass by reference
  cout << "\n View the subject information again: ";
  DS.getDetail();  // the value within the object has changed
  getch();
};
```

Class as Return Value from Function

- Syntax for declaring function that return a class object

```
className functionName(parameter list)
{
     // function body
}
```

- Syntax to call function that return a class

```
objectName = functionName();
```

where,

- `objectName`, an object from the same class with the type of class return from the function. This object will be assigned with the value returned from function
- `functionName()`: function that return class

Class as Return Value from Function

Function that return a class object, Point

```
Point findMiddlePoint(Point T1, Point T2)
{
    double midX, midY;
    midX = (T1.get_x() + T2.get_x()) / 2;
    midY = (T1.get_y() + T2.get_y()) / 2;
    Point middlePoint(midX, midY);
    return middlePoint;
}
```

Return type is a class

Create instance of Point

Return instance of Point

Statement that call function that return a class

```
Point point1(10,5), point2(-5,5);
Point point3; // use defult argumen
// point3 is the point in the middle of point1 and point2
point3 = findMiddlePoint(point1,point2)
```

Call `findMiddlePoint` that return object and assign to `point3`

Array of class

- A group of objects from the same class can be declared as array of a class
- Example:
  - Array of class students registered in Data Structure class
  - Array of class lecturer teaching Data Structure Subject
  - Array of class subjects offered in Semester I.
- Every element in the array of class has it's own data member and function member.
- Syntax to declare array of objects :

```
    className arrayName[arraySize];
```

Array of class

```
class staff {
    char name[20];
    int age ;
    float salary;
public:
    void read_data() ;
    { cin >> name >> age >> salary;
    void print_data()
    { cout << name << age << salary; }
} ;

main()
{
     staff manager[20];
     // declare array of staff
}
```

Declare 20 managers from class staff. Each element of manager has name, age and salary.

Array of class

How to call member function for `manager` array?

1. **By using array subscript in order to access manager in certain location of the array.**

```
cin >> n ;
manager[n].read_data() ;
cout << manager[n].name << manager[n].age ;
manager[n].print_data() ;
```

2. By using loop in order to access a group of managers.

```
// read information for 10 managers
for ( int x = 0 ; x < 10; x++ )
  manager[x].read_data();
// print information of 10 managers
for ( int y = 0 ; y < 10; y++ )
   manager[y].print_data();
```

Pointer to Object

- Pointer can be used to store address of an object.
- Example statement to create instance of student

```
student student1;
```

- Example statement to create a pointer variable, named `studentPtr`

```
student* studentPtr = &student1;
```

- The pointer can be initialized with the address of instance `student1`

```
studentPtr = &student1;
```

- The 2 statements above can be combined as:

```
student* studentPtr = &student1;
```

Pointer to Object

2 methods to access class member through pointer variable `studentPtr` :

1. `(*studentPtr).print()`

   or

2. `studentPtr->print()`

Pointer to Object

- Operator `new` can also be used to allocate memory for a pointer variable.
- Operator `delete` destroys memory for a pointer variable.

```
void main()
{
    student *ptr = new student("Ahmad", 123123);
    ptr -> print();
    delete(ptr);
    ptr = new student("Abdullah", 234234);
    ptr ->print();
    delete(ptr);
}
```

Conclusion and Summary

- Abstract Data Type is a collection of data and a set of operations on the data.
- Abstraction implements information hiding and encapsulation, whereby other modules cannot tamper with the data.
- In C++, abstraction is implemented by using class.

Thank You

http://comp.utm.my/