Lab Guide for Introduction to Object Oriented Programming using C++ - Part I

Education and Research

Infosys®
POWERED BY INTELLECT
DRIVEN BY VALUES

Author(s)	
Authorized by	
Creation/Revision Date	June-2009
Version	1.0

COPYRIGHT NOTICE

Education and Research Department
Infosys Technologies Limited
Electronic City
Hosur Road
Bangalore - 561 229, India.

Tel: 91 80 852 0261-270
Fax: 91 80 852 0362
www.infy.com
mailto:E&R@infy.com

Document Revision History

Version	Date	Author(s)	Reviewer(s)	Comments
0.0	Mar-2007	Vani Vasudevan	Amit Tiwary	Initial Draft
0.0	May-2007	Vani Vasudevan	Amit Tiwary	Review comments have been incorporated
1.0	June-2009	Payal Jain, Smita Khode, Priya Chandan Kudchadker, DODDE_GOWDA	Archana Pralhadrao Dhande	
1.1	August-2009	Payal Jain, Smita Khode, Priya Chandan Kudchadker, DODDE_GOWDA		Review comments implemented

Contents

Context

This document contains assignments to be completed as part of the hands on for the subject Introduction to Object Oriented Programming using C++ - Part I (Course code: LA83).

Note: In order to complete the course, assignments in this document must be completed in the sequence mentioned.

Day 1 Assignments

Assignment 1: Use of Structures

Objective: To understand the use of structures

Problem Description: Write a C++ program which accepts 5 student records .The records of each student will include student id, name and total marks, grade and percentage. We should display the details of all the students at the end.

Estimated time: 30 mins

Step 1: We can store the student details in a structure. We can use a character array to store the names i.e the character array will be a member of the structure.

Step 2: Create an array for 5 structure variables to access the information for 5 Students .

Step 3: Accept the information from the user to store the details of the students.

Step 4: Use a display function which will display the details.

Summary of this assignment:
In this assignment, you have learned

- How to declare a structure
- How to initialize the members within a structure
- How to create a structure variable

Assignment 2: Array of Structures, Union and Enum

Objective: To understand Array of structure, union and enum

Problem Description: Write a C++ program that will accept the following information for each for employees in an organization. The employee can be a trainee or a Lateral.

General Information
1. Employee name
2. Employee Number
3. Employee Date of Birth
4. Employee address

For a Trainee, add the following information:
1. Trainee Stream
2. Trainee Batch
3. Training Type

For a Lateral, add the following information:
1. Previous employer's name
2. Years of work experience

Enter this information for 10 employees from the organization. Then print the details of all the employees. Store the information in an array of structures, where each array elements (i.e. each structure) contains the information for an employee. We can store the address and date of birth in a structure within the structure. Make use of a union to represent the variable information (either a trainee or a lateral) that is included as a part of the structure. This union should itself contain two structures, one for trainee and other for lateral.

Step 1: Create a structure declaration for the Employee in an organization. The members of the structure should include the general information.

Step 2: For representing date of birth we can have another structure Date and create a variable of the structure Date as a member of Employee.

Step 3: We can include a union as member of the structure to represent Trainee or Lateral. This union will contain two structures one for Trainee and one for Lateral.

Step 4: Create an array of structures by creating an array of 10 structure variables where each element contains the information for an employee.

Step 5: Use enum to compare the menu options in a switch case

Estimated time: 45 mins

Note: use enum type in comparing menu options.

Assignment 3: Two Dimensional Array and Pointers

Objective: To implement 2-D array and manipulate it using pointers

Problem Description:

To find the transpose of a square matrix, display the sum of principle elements and secondary diagonal elements of the matrix.

Estimated time: 30 mins

Step 1: Pass the order of the matrix (Number of Rows and Columns) from command line.

Step 2: Read the matrix elements

Step 3: Find the transpose of the matrix

Step 4: Use pointer arithmetic to calculate and display the sum of principle and secondary diagonal elements of the matrix.

Summary of this assignment:
In this assignment, you have learnt

- How to declare and use 2-D array
- How to declare and initialize the pointer
- How to carry out pointer arithmetic

Note: In 3x3 matrix , the principle diagonal elements are (0,0),(1,1),(2,2) and the secondary diagonal elements are (0,2),(1,1),(2,0)

Assignment 4: Array of Function pointers

Objective: To understand Array of Function pointer

Problem Description:
Declare an array of function pointers. The function pointers should point to a function accepting two integers and returning an integer.

Implement the following functions:

```
int  fnAdd(int,int);
int fnMultiply(int,int);
int fnDivide(int,int);
int fnSubtract(int,int);
```

Initialize the array with the address of these four functions and call the functions using the function pointers in the array.

Hint: Declare an array of function pointers having the same prototype as:
`int (*fnptr[4])(int,int);`

Assign the corresponding fnAdd, fnMultiply, fnDivide and fnSubtract function's addresses to the elements of the function pointer array

Estimated time: 45 mins

Summary of this assignment:

In this assignment, you have learnt

- How to declare and use array of function pointers

Assignment 5: Class and Object

Objective: To understand the concept of ADT used for achieving encapsulation and abstraction.

Problem Description: A courier company "SpedFast" wants to automate their process of collecting and dispatching shipments. The company has a set of customers, many of whom are already in the company's database. Each customer has a unique Customer ID. Each customer has an address as well, which is used as "Sender's address" for shipments.

Customers approach the Dispatchers at SpedFast counters to send shipments. One customer can send as many shipments as required. The Dispatcher initiates a shipment after the payment from the customer.

Each shipment has a Shipment ID associated with it. A shipment also has a Recipient's address, priority, weight of the shipment, shipping date and date of delivery. The amount is calculated based on the weight of the shipment and priority. Currently SpedFast provides "LOW, NORMAL and URGENT" priorities with different rates on shipments.

SpedFast also intends to have a Greeting Shipment service for customers to send Gifts for celebrations. This is similar to the normal shipments, but it also carries a "Greeting message" from the customer.

Step 1: Identify all the 'nouns' (type of objects or classes) in the problem requirement.
Step 2: Identify all the member variables and methods the class should have.

Step 3: Identify the member variables and methods (getter and setter methods for each and every attribute) for the rest of the classes.

Step 4: Ensure that the class is fully independent of other classes and contains all the attributes and methods necessary

Step 5: Declare a class "**Customer**" and store it in **Customer.h**

Step 6: Define all the member methods of **Customer** Class and store it in **Customer.cpp**

Step 7: Define a main function and store it in **Customer _Main.cpp**

a. In this function, read the Customer details for and store it in the temporary variables.
b. Define an object of type Customer and invoke the corresponding setter methods to set the member variables with the given values stored in temporary variables.
c. Using getter methods, display the Customer details.

Estimated time: 60 mins

Summary of this assignment:

In this assignment, you have learnt

- How to implement a class in C++
- How to implement getter and setter methods
- How to create object and pass messages

Programs

1. Write a program to multiply two square Matrices .Accept the input for the array elements from the user and display the resultant matrix.
2. Write a C++ program to display the largest element of a matrix. Accept the input for the array elements from the user and find the largest element in a matrix.
3. Write a C++ program to implement to create pointer to structures. Create a structure Movie including members name, year. Create a pointer to a structure for accessing the members through the pointer variable. Accept the details for the Movie name and year from the user and print the details.
4. Write a C++ program to store information about friends like name, email, phonenumber. Implement it using a class and have set and get methods store and retrieve the information.
5. Write a C++ program to illustrate the use of namespaces.

Day 2 Assignments

Assignment 1: UML - Class Diagram

Objective: To analyze and design a UML class diagram for the given problem.

Problem Description:

A courier company "SpedFast" wants to automate their process of collecting and dispatching shipments. The company has a set of customers, many of whom are already in the company's database. Each customer has a unique Customer ID. Each customer has an address as well, which is used as "Sender's address" for shipments.

Customers approach the Dispatchers at SpedFast counters to send shipments. One customer can send as many shipments as required. The Dispatcher initiates a shipment after the payment from the customer.

Each shipment has a Shipment ID associated with it. A shipment also has a Recipient's address, priority, weight of the shipment, shipping date and date of delivery. The amount is calculated based on the weight of the shipment and priority. Currently SpedFast provides "LOW, NORMAL and URGENT" priorities with different rates on shipments.

SpedFast also intends to have a Greeting Shipment service for customers to send Gifts for celebrations. This is similar to the normal shipments, but it also carries a "Greeting message" from the customer.

Estimated time: 45 mins

Step 1: Identify all the 'nouns' (type of objects or classes) in the problem requirement.

Step 2: Identify the commonalities between classes (Generalization) if it is obvious.

Step 3: Identify all the member variables and methods the class should have.

Step 4: Identify the member variables and methods (getter and setter methods for each and every attribute) for the rest of the classes.

Step 5: Ensure that the class is fully independent of other classes and contains all the attributes and methods necessary.

Step 6: Keep all data members private (-) or protected (#).
Step 7: The methods in class should completely abstract the functionality. The methods in the class should not be a force fit of procedural code into a class.

Step 8: Inherit and extend classes from the base class ONLY IF situation has scope for it. Shift the commonalities among classes to the base class

Step 9: Define the "Has-A" and "Uses-A" relationship among classes
(Is-A has been defined already in previous step)

Step 10: Keep the number of classes in your application under check (Do not create any unnecessary classes)

Step 11: REVIEW your design

Summary of this assignment:

In this assignment, you have learnt

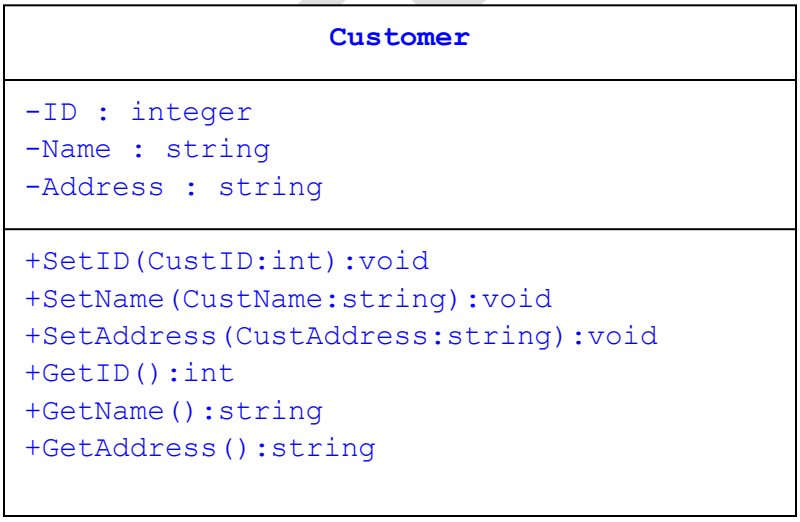
- How to design a class diagram for the given problem statement.

Assignment 2: Class and Object

Objective: To understand the concept of ADT used for achieving encapsulation and abstraction.

Problem Description:

Implement the following class diagram using C++.



Note: 1. All string data types should be declared as char array.
2. Using a class , you can create any number of objects

Estimated time: 45 mins

Step 1: Declare a class "**Customer**" and store it in **Customer.h**

Step 2: Define all the member methods of Customer Class and store it in **Customer.cpp**

Step 3: Define a main function and store it in **Customer_Main.cpp**

d. In this function, read the customer details for and store it in the temporary variables.
e. Define an object of type **Customer** and invoke the corresponding setter methods to set the member variables with the given values stored in temporary variables.
f. Using getter methods, display the customer details.

Summary of this assignment:

In this assignment, you have learnt

- How to implement ADT in C++.
- How to use conditional compilation directives
- How to implement getter and setter methods
- How to create object and pass messages

Assignment 3: Constructors and Destructor

Objective: To understand the concept of Default Constructor/Destructor, Parameterized Constructor and Copy Constructor

Problem Description:

Enhance the Assignment No.2 and do the necessary modification as mentioned below:

Customer
-ID : integer -Name : string -Address : string
+Customer ( ) **+Customer (CustID: int, CustName: string, CustAddress: string)** **+Customer(Customer& oCustomer)** +SetID(CustID:int):void +SetName(CustName:string):void +SetAddress(CustAddress:string):void +GetID():int +GetName():string +GetAddress():string **+~Customer ()**

Estimated time: 45 mins

Step 1: Declare a class "Customer" and store it in Customer.h

Step 2: Define all the member methods of Customer Class and store it in Customer.cpp

1. **Customer()** : to initialize the member variables to 0/NULL.
2. **Customer (CustID: int, CustName: string, CustAddress: string)**: To initialize the member variables with the parameters received.
3. **Customer(Customer& oCustomer)** : To initialize an object with another object.
4. **~Customer()** : To display a message during object destruction.

Step 3: Define a main function and store it in Customer_Main.cpp

1. In this function, read the Customer details and store it in the temporary variables.
2. Define 3 objects which uses the 3 types of constructors defined in Customer.cpp
3. Using getter methods, display all the Customer details.

Summary of this assignment:

In this assignment, you have learnt

- How to implement different type of constructors and destructor
- How to use the constructors for initializing the objects

Assignment 4: 'this' pointer

Objective: To understand the usage of 'this' pointer

Problem Description:

Modify Assignment 2 of Day 3.

Estimated time: 30 mins

Step 1: Use "this" pointer to return the Address of the invoked object in GetRecipientAddress() and GetSenderAddress() methods
Step 2: Use "this" pointer to return the Date of the invoked object in GetDateOfShipment() and GetDateOfDelivery() methods
Step 3: Use Cascaded method call to get the individual member data of Address/Date object.

Summary of this assignment:

In this assignment, you have learnt

- How to use "this" pointer

Assignment 5: Complete Class Diagram Implementation

Objective: To understand the complete Class Diagram implementation in C++.

Problem Description:
A Customer wants to send a courier to the intended recipient through the dispatcher.The Dispatcher accomplishes this task through Shipment.

Estimated time: 120 mins

Step 1: Create a class "Dispatcher" as below:

Dispatcher
-Name: string (char *)
-DispatcherID :int -oCustomer:Customer -oShipment:GreetingShipment
+Dispatcher () +SetName(Name:string):void +GetName():string +SetID(DispatcherID:int):void +GetID():int +SetCustomer(oCustomer:Customer):void +GetCustomer():Customer +SetShipment(oShipment:GreetingShipment):void +GetShipment(oShipment:GreetingShipement):void

Step 2: In main function,

a. Create an object of type Dispatcher.
b. Get the details of the Customer and set the oCustomer member variable of the Dispatcher with the corresponding values.
c. Get the details of the Shipment including the type of shipment (Normal / Greeting). The type of shipment can be stored in a temporary variable.
    i. If it is normal Shipment, set GreetingMessage member variable of GreetingShipment to NULL otherwise, set it to the read value.
d. Display all the details in a neat format including the calculated amount of the shipment.

Summary of this assignment:

In this assignment, you have learnt

- The complete implementation of Class diagram in C++

Day 3 Assignments

Assignment 1: static members and constant methods

Objective: To understand the concept of static members, const members.

Problem Description:

Modify the Assignment 4 of Day2 to get the total number of Customer objects and make all the getter methods as constant.

Estimated time: 30 mins

Static members:

Step 1: Add static data member "m_iCustomerCount" to store the total number of customer objects in the Customer Class.
Step 2: Add static member method "GetCustomerCount () to return the "m_iCustomerCount".
Step 3: In default constructor Customer (), Increment the m_iCustomerCount.
Step 4: In destructor ~Customer (), Decrement the m_iCustomerCount.

Const methods and Constant return types:

Step 1: Make all the getter methods as constant.
Step 2: If the return type of the getter method is string (char *), make it as constant return type.

Summary of this assignment:

In this assignment, you have learnt

- How to use static and constant keywords in C++

Assignment 2: friend Classes

Objective: To understand the concept of aggregation and inline function

Problem Description:

Create three classes as Address , date and shipment where shipment is declared as a friend class to the both of address and date class .

Estimated time: 90 mins

Step 1: Implement the class "Address" with the following details.
(Address.h and Address.cpp)

Address
-DoorNumber: string -StreetName : string -Locality : string -City:string -PostalCode : int
+Address (DoorNumber: string, StreetName: string, Locality: string, City: string, PostalCode: int) +GetDoorNumber():string +GetStreetName():string +GetLocality():string +GetCity():string +GetPostalCode():int Friend class shipment;

Step 2: Implement the class "Date" with the following details.
(Date.h and Date.cpp)

Date
-Day:int -Month: int -Year:int
+Date (Day: int, Month: int, Year: int) +GetDay():int +GetMonth():int +GetYear():int Friend class shipment;

Step 3: Implement the class "Shipment" with the following details.
(Shipment.h and Shipment.cpp , In Shipment.h include Date.h and Address.h)

Shipment
-RecipientAddress: Address - SenderAddress:Address -DateOfShipment:Date -DateOfDelivery:Date -Priority:enumDataType -Weight: float
+Shipment(RecipientAddress:Address,SenderAddress:Address,DateOfShipment:Date,DateOfDelivery:Date,Priority:enumDataType,Weight:float) +GetRecipientAddress():Address +GetSenderAddress():Address +GetDateOfShipment():Date +GetDateOfDelivery():Date +GetPriority():enumDataType +GetWeight():float

Step 4: Create enumDataType for assigning values for the priority.
Priority Values: LOW=0, NORMAL and URGENT.

Step 5: Create a non-member function to calculate the Amount to be paid for shipment.

a. if the weight is less than 1 kg and priority is LOW then
    i. Amount = Weight * 40
b. if the weight is less than 1 kg and priority is URGENT then
    i. Amount – Weight * 80.75
c. If the weight is greater than 1 kg and priority is LOW then
    i. Amount = Weight * 100.50
d. If the weight is greater than 1 kg and priority is URGENT then
    i. Amount = Weight * 225.50
e. If the priority is NORMAL, add Rs.10 with the calculated amount of LOW priority
f. Make the non-member function as a inline function

Step 6: In the main function, create an object of type shipment and invoke the member methods and non-member function to calculate the amount and display the shipment details.

Summary of this assignment:

In this assignment, you have learnt

- How to implement friend classes

Assignment 3: Friend Function

Objective: To understand friend functions

Problem Description:

Create three classes as vehicle and define a friend function sort () to this class .

Estimated time: 30 mins

Step 1: Implement the class "vehicle" with the following details.
(vehicle h and vehicle.cpp)

Vehicle
-Count: int - > static **-ID : int** **-Name : char[]** **-Type : char[]** **-Cost : double** **-Mileage : float**
+*Vehicle* () **+*Vehicle* (int,char [],char[],double, float)** **+Display():void** **+Friend Sort (Car[]): void**

Step 2: Implement the function sort()to read an array of vehicle objects, sort it in ascending order of vehicle ID, and display the vehicle details in sorted order by invoking Display() member function.

Summary of this assignment:

In this assignment, you have learnt

- How to implement friend function .

Assignment 4: Function Overloading

Objective: To understand compile-time polymorphism: Function overloading

Problem Description:

Write a C++ program that will implement the following functionalities:

- void Add(int,int);
  - To Add the given 2 integer numbers and print the sum.
- void Add(char[],char[]);
  - To Concatenate the given 2 character array (strings) and print the concatenated string.

Estimated time: 30 mins

Summary of this assignment:

In this assignment, you have learnt

- How to implement function overloading

Assignment 5: Operator Overloading

Objective: To understand compile-time Polymorphism: Operator Overloading

Problem Description:

Implement the following class:

String
-Name : string (char *)
+String() +SetName(Name:string):void +GetName():string +Operator +(oString:String):String

Estimated time: 30 mins

In the main function,

Step 1: Create 3 objects of type String

Step 2: Set values for 2 objects by invoking SetName ().

Step 3: Invoke the + method as follows.

a. oString3 = oString1 + oString2

Step 4: Display the concatenated value of oString3 by invoking GetName()

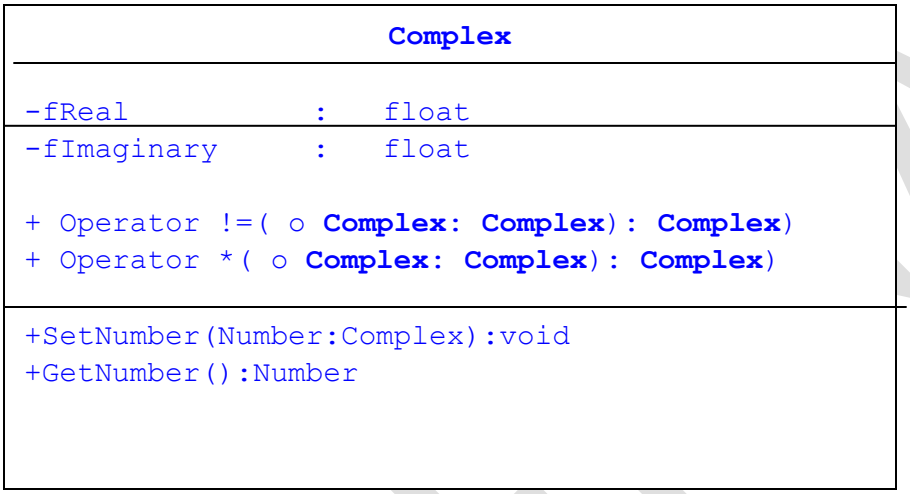
Summary of this assignment:

In this assignment, you have learnt

- How to implement operator overloading.

Assignment 6: Operator Overloading

Implement the following class:



Estimated time: 30 mins

In the main function,

Step 1: Create 3 objects of type Complex

Step 2: Set values for 2 objects by invoking SetNumber ().

Step 3: Invoke the != method as follows.

oString3 = oString1 != oString2

b. Invoke the * method as follows

oString3 = oString1 * oString2

Step 4: Display the multiplied values as of oComplex3 by invoking GetNumber()

Summary of this assignment:

In this assignment, you have learnt

- How to implement operator overloading.

Day 4 Assignments

Assignment 1: memory management using malloc () and free()

Objective: To learn how to allocate and release memory using malloc() and free ().

Problem Description: Type the following programs separately, compile, run and observe the output:

Estimated time: 20 Minutes

```
/************************************************************
* Filename: MallocEx1.cpp
* Date:     May 2009
* Author:   E&R, Infosys
* Description: Program to demonstrate memory allocation using malloc()
************************************************************/

#include <iostream>
using namespace std;

/************************************************************
* Function main
* DESCRIPTION: Entry point for the program
* PARAMETERS:
*   int argc - no of cmd line parameters
*   char **argv - cmd line parameters
* RETURNS: 0 on success,   Non-Zero value on error
************************************************************/




/* Employee Calss definition */
class Employee
{
private:
        int empId;
        char empName[30];
public:

       /* Constructor */
        Employee() {
```

```
        cout<<"Constructor is getting executed "<<endl;
        }

         /* Function to input data */

       void getData()          {
        cout<<"enter empId and empname\n";
        cin>>empId>>empName;
       }

       /* Function to output data */

       void putData()        {
         cout<<empId<<"    "<<empName<<endl;
       }
};

int main(int argc, char ** argv) {

       /* dimention for toy integer array */
        int intArraySize = 5;

       /* loop index counter */
        int i;

       /* pointer to our array */
        Employee *empArray;

        empArray = (Employee *) malloc(intArraySize*sizeof(Employee));

       /* loop over the arary, and do some computations */
        for(i=0; i<intArraySize; i++) {
              empArray[i].getData();
        }
       /* loop over the arary, and do some computations */
        for(i=0; i<intArraySize; i++) {
                  empArray[i].putData();
        }
        /* free malloc'd memory and end program */
        free(empArray);
        return 1;
}
```

Summary of this assignment:

You have learnt how to allocate and de allocate memory in C++ using malloc() and free () functions.

Assignment 2: Memory management using calloc()

Objective: To learn how to allocate memory using calloc().

Problem Description: Write a program to allocate memory for an array of Employee class using calloc() Input the array size and implement data input and output operations.
Note: Employee class in the first assignment can be used.
Estimated time: 20 Minutes

Summary of this assignment:

You have learnt how to use cllaoc().

Assignment 3: Memory allocation using realloc()

Objective: To learn how to enhance the memory size pointed to by some pointer.

Problem Description: Type the following programs separately, compile, run and observe the output:

Estimated time: 30 Minutes

```
/*************************************************************
* Filename: ReallocEx.cpp
* Date:     May 2009
* Author:   E&R, Infosys
* Description: Program to demonstrate the use of realloc()
*************************************************************/

#include <iostream>
#include <iomanip>
using namespace std;

/*************************************************************
* Function main
*
* DESCRIPTION: Entry point for the program
* PARAMETERS:
*   int argc - no of cmd line parameters
*   char **argv - cmd line parameters
* RETURNS: 0 on success,   Non-Zero value on error
*************************************************************/
```

```
int main (int argc, char ** argv) {
  int intData,iN;
  int intCount=0;
  int * iPtr  = NULL;

  do {
     cout<<"Enter an integer value (0 to end): ";
     cin>>intData;
     intCount++;
     iPtr = (int*) realloc (iPtr, intCount * sizeof(int));
     if (iPtr==NULL)
       { puts ("Error (re)allocating memory"); exit (1); }
     iPtr[intCount-1]=intData;
  } while (intData!=0);

  cout<<"Numbers entered: "<<endl;
  for (iN=0;iN<intCount;iN++)
     cout<<iPtr[iN]<<endl;
  free (iPtr);

  return 0;
}

}

/**************************************************************
* End of Realloc.cpp
**************************************************************/
```

Summary of this assignment:

You have learnt how to use realloc()

Assignment 4: Memory Management using new and delete

Objective: To learn how to allocate memory and release memory using new and delete respectively.

Problem Description: Type the following programs separately, compile, run and observe the output:

Estimated time: 30 Minutes

```
/*****************************************************************
* FileName: NewEx1.cpp
* Author: E&R Department, Infosys Technologies Limited
* Date: May 2009
*****************************************************************/
#include <iostream>
#include <string>

using namespace std;

int main() {

// Allocating memory for one integer
  int *pIntO= new int;
  *pIntO=100;
  cout << *pIntO << endl;
  delete pIntO;

/* Allocating memory for one integer and initializing it to 500 */
  int *pIntTo= new int(500);
  cout << *pIntTo << endl;
  delete pIntTo;

// Allocating memory for 6 characters
char *pChar= new char[strlen("Jumpy")+1];
  strcpy(pChar,"Jumpy");
  cout << pChar << endl;
  delete []pChar;
// Deallocating pChar

  return 0;
}

/*****************************************************************
* End of file NewEx1.cpp
*****************************************************************/
```

```
/*****************************************************************
* FileName: NewEx2.cpp
* Author: E&R Department, Infosys Technologies Limited.
* Date: May 2009
*****************************************************************/
```

```
#include <iostream>
#include <string>

using namespace std;

class Trainee {
  char *Name;
public:
   Trainee(char str[]);
  ~Trainee();
  void Details() {
     cout << Name ;
  }
};

Trainee::Trainee( char str[]) {
  Name= new char[strlen(str)+1];
  strcpy(Name,str);
}

 Trainee :: ~Trainee() {
   if(Name !=NULL)  delete []Name;
}
/*************************************************************
* Function main
*
* DESCRIPTION: Entry point for the program
* PARAMETERS:
*   int argc - no of cmd line parameters
*   char **argv - cmd line parameters
* RETURNS: 0 on success,   Non-Zero value on error
*************************************************************/

int main() {
  Trainee *poT = new Trainee("Jumpy");
  poT->Details();
  delete poT;
  return 0;
}

/*************************************************************************
* End of file NewEx2.cpp
*************************************************************************/
```

Summary of this assignment:

You have learnt how to allocate memory and release memory through new and delete.

Assignment 5: How to detect Memory Leaks

Objective: To learn how to detect memory leaks using valgrind debugging tool.
Problem Description: Type the following code snippet and follow the steps provided here.

Estimated time: 20 Minutes
Step 1: Type the following code and compile this program as shown below.

g++ -o final -g MemLeak.cpp

```
/*****************************************************************
* FileName: MemLeak.cpp
* Author: E&R Department, Infosys Technologies Limited
* Date: May 2009
*****************************************************************/

 #include<iostream>
 using namespace std;

/*************************************************************
* Function main
*
* DESCRIPTION: Entry point for the program
* PARAMETERS:
*   int argc - no of cmd line parameters
*   char **argv - cmd line parameters
* RETURNS: 0 on success,   Non-Zero value on error
*************************************************************/

  int main(void) {

 /* dimention for toy integer array */

 int intArraySize = 10;

/* pointer to our array */
  int* intArray;

 /* dynamically declare some memory for our toy integer array */
```

```
    intArray = new int[intArraySize];

    /* loop over the arary, and do some computations */

    for(int i=0; i<intArraySize; i++) {
                intArray[i] = i*10;
        }

    intArray = new int[intArraySize];



    /* loop over the arary, and do some computations */

    for(int i=0; i<intArraySize; i++) {
     intArray[i] = i*20;
}

  /* free malloc'd memory and end program */

      delete intArray;

   return 1;
}




/*****************************************************************
* End of file MemLeak.cpp
*****************************************************************/
```

Step 2: Use the valgrind tool to debug the program for memory leak and observe the output

valgrind ./final

Summary of this assignment:

You have learnt how to debug programs to check for memory leaks using valgrind tool

Assignment 6: To understand Law of Big three in C++

Objective: To know the importance of implementation of 3 special functions in C++

- Copy constructor
- Copy Assignment Operator
- Destructor

Problem Description: Type the following programs separately, compile, run and observe the output:

Note: First program produces error.

Estimated time: 30 Minutes

```
/*************************************************************
* Filename: Big_Three1.cpp
* Date:     May 2009
*************************************************************/
#include <iostream>
using namespace std;

class Course
{
 private:
      char courseName[30];
};
class Faculty{
private:
  Course* cPtr;
public:
  Faculty() : cPtr(new Course())
 {
  cout<<"creating Faculty and allocating course"<<endl;
}
~Faculty() {
  delete cPtr;
  cout<<"Deleting Faculty and closing the course"<<endl;
}
};
/*************************************************************
* Function main
*
* DESCRIPTION: Entry point for the program
* PARAMETERS:
*   int argc - no of cmd line parameters
*   char **argv - cmd line parameters
* RETURNS: 0 on success,   Non-Zero value on error
*************************************************************/

int main()
```

```
{
 Faculty f1;
 Faculty f2;
 f2=f1;
}
```

```
/*************************************************************
* Filename: Big_Three2.cpp
* Date:     May 2009
*************************************************************/
#include <iostream>
using namespace std;
class Course
{
 private:
      char courseName[30];
};
class Faculty{
private:
  Course* cPtr;

public:
  Faculty() : cPtr(new Course())
 {
  cout<<"creating Faculty and allocating course"<<endl;
}

~Faculty() {
  delete cPtr;
  cout<<"Deleting Faculty and closing the course"<<endl;
}

Faculty& operator=(const Faculty& other) {

if (this==&other)
    return *this;
  *cPtr=*cPtr;
return *this;
}

};

/*************************************************************
* Function main
```

```
*
* DESCRIPTION: Entry point for the program
* PARAMETERS:
*   int argc - no of cmd line parameters
*   char **argv - cmd line parameters
* RETURNS: 0 on success,   Non-Zero value on error
*****************************************************/
int main()
{
 Faculty f1;
 Faculty f2;
  f2=f1;
}
```

Summary of this assignment:

You have learnt that whenever you implement one of the three special functions Copy constructor, Copy Assignment Operator or Destructor, then you must implement the remaining two also.

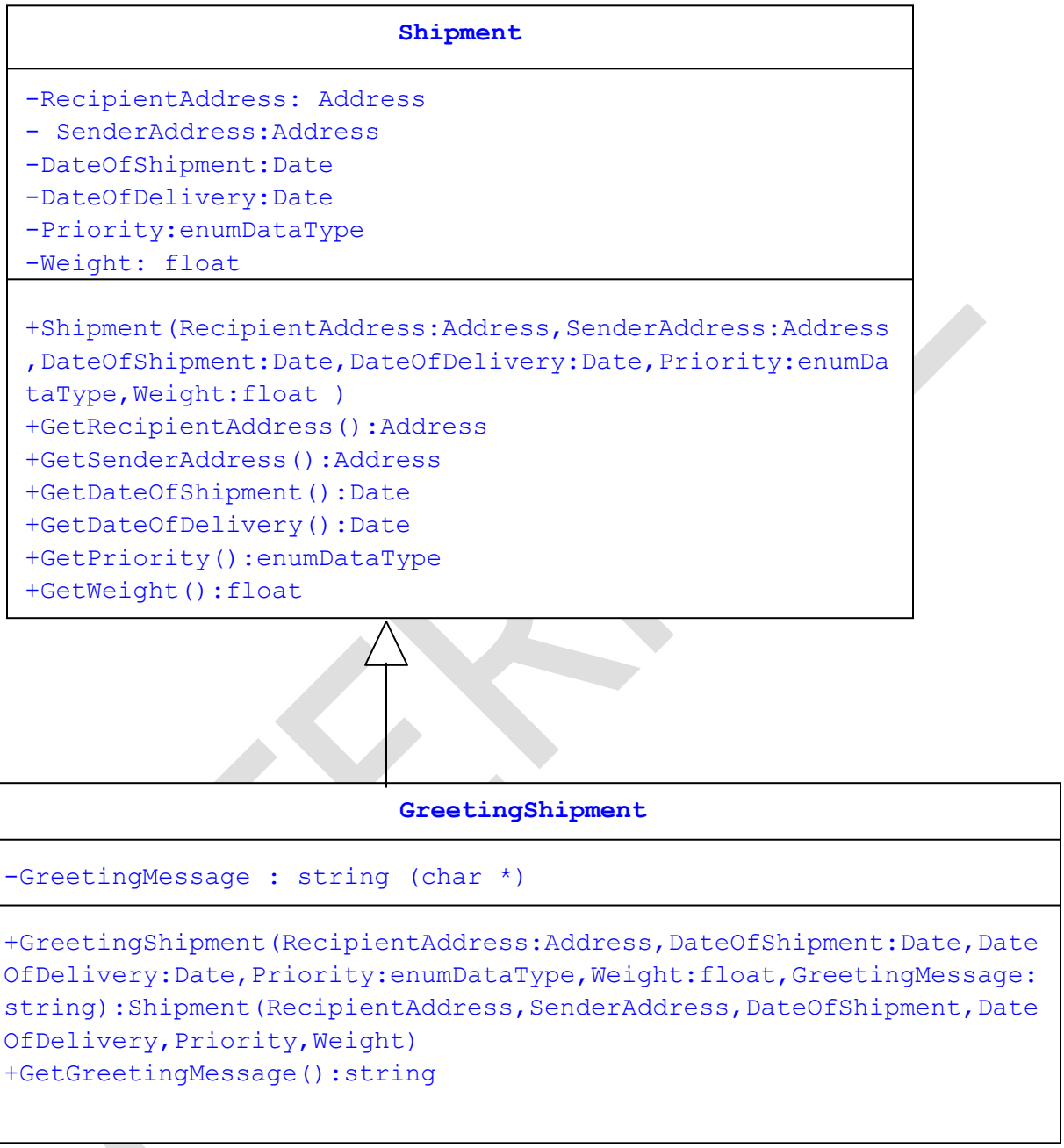
Day 5 Assignments

Assignment 1: Inheritance

Objective: To understand the concept of Inheritance

Problem Description:

Enhance Assignment 2 of Day 3 by implementing the inheritance



Estimated time: 30 mins

Step 1: Create a class "**Greeting Shipment**" by reusing "**Shipment**" class. (**GreetingShipment.h**)
Step 2: Implement all the methods specified in the "**Greeting Shipment**" and store it in **GreetingShipment.cpp**
Step 3: In main function,

a. Create 2 objects of type Date class to store delivery and shipment dates
b. Create 2 objects of type Address class to store address of sender and receiver.

c. Create an object of type Greeting Shipment by invoking the parameterized constructor with the required parameters.
d. Display all the details of Greeting Shipment by invoking appropriate getter methods.

Summary of this assignment:

In this assignment, you have learnt

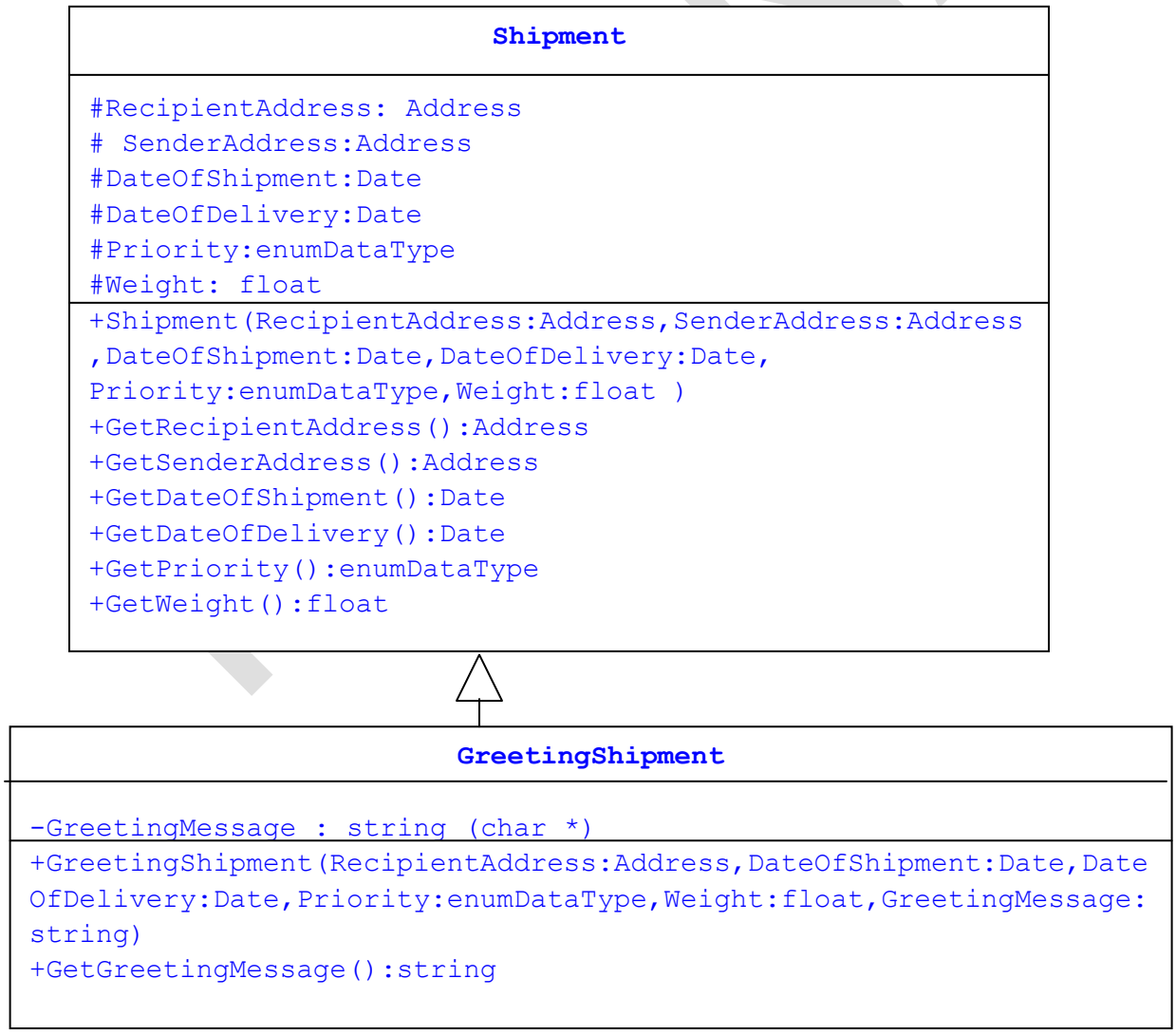
- How to implement simple inheritance.

Assignment 2: 'protected' Access Specifier

Objective: To understand the significance of protected access specifier in Inheritance.

Problem Description:

Modify the Assignment 1 of Day 5 with the following class diagram and steps



Steps mentioned in Assignment 1 of Day5 have to be repeated with the following modification.

Step 1: Shipment class data members have to be made protected.

Step 2: GreetingShipment class is derived from Shipment class with the "public" visibility mode.

Step 3: In GreetingShipment () parameterized constructor, all the member variables (including Shipment class members) have to be initialized with the corresponding parameters passed.

Summary of this assignment:

In this assignment, you have learnt

- How to use the protected access specifier

Assignment 3: Complete Class Diagram Implementation

Objective: To understand the complete Class Diagram implementation in C++.

Problem Description:

A Customer wants to send a courier to the intended recipient through the dispatcher.The Dispatcher accomplishes this task through Shipment.

Estimated time: 120 mins

Step 1: Create a class "Dispatcher" as below:

Dispatcher
-Name: string (char *)
-DispatcherID :int -oCustomer:Customer -oShipment:GreetingShipment
+Dispatcher () +SetName(Name:string):void +GetName():string +SetDispatcherID(DispatcherID:int):void +GetDispatcherID():int +SetCustomer(oCustomer:Customer):void +GetCustomer():Customer +SetShipment(oShipment:GreetingShipment):void +GetShipment(oShipment:GreetingShipement):void

Step 2: In main function,

e. Create an object of type Dispatcher.

f. Get the details of the Customer and set the oCustomer member variable of the Dispatcher with the corresponding values.
g. Get the details of the Shipment including the type of shipment (Normal / Greeting). The type of shipment can be stored in a temporary variable.
    i. If it is normal Shipment, set GreetingMessage member variable of GreetingShipment to NULL otherwise, set it to the read value.
h. Display all the details in a neat format including the calculated amount of the shipment.

Summary of this assignment:

In this assignment, you have learnt

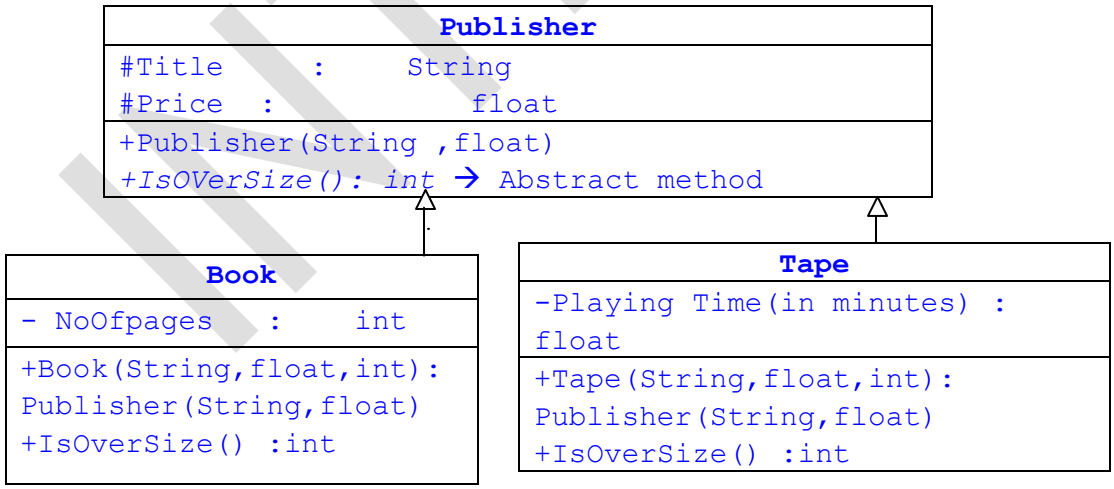
- The complete implementation of Class diagram in C++

Assignment 4: Abstract Class, Overriding and Dynamic Binding

Objective: To understand the concept of Abstract Class, Overriding and Dynamic bindling

Problem Description:
Write a C++ program to implement the given Class Diagram.

In the main program use pointers of publisher type, create objects of Book and Tape. Invoke IsOverSize() method of both objects using the pointer"



Implementation of IsOverSize method in Book Class:

- If the NoOfpages is greater than 800, return 1 to indicate oversize.
- In main(), display the string "Book exceeds maximum size"

Implementation of IsOverSize method in Tape Class:

- If the Playing Time is greater than 90 minutes, return 1 to indicate oversize.
- In main(), display the string "Playing Time is Longer"

Estimated time: 60 mins

Summary of this assignment:

In this assignment, you have learnt

- How to create Abstract Class
- How to implement method overriding
- How to implement Dynamic Binding