CPS235 Object Oriented Programming in C++

LAB 8 BESE-15A *10th May 2010*

Pointers and Dynamic Memory

Objectives

By the end of this lab, you should be able to

- Understand the concept of dynamic memory allocation and the problems of memory leak and dandling pointers
- Use new and delete operators to allocate memory at run-time
- Understand why some functions (copy constructor, overloaded operators) need to be added to classes that have dynamic data and how it can be done

Note: Have a look at the handout **PointersAndMemory.pdf**

Question 1:

(a) The following program contains several errors in the way that the pointers are dereferenced in main. Correct these errors and record the output of the program.

```
class Student
{
public:
   Student();
   Student(string student_name, string major_at_acceptance);
   void set_name(string new_name);
   void set_major(string new_major);
   string get_major() const;
   string get_name() const;
private:
   string name;
   string major;
};

Student::Student()
{
      name = "";
      major = "";
}

Student::Student(string student_name, string major_at_acceptance)
{
      name=student_name;
      major=major_at_acceptance;
}
```

```
void Student::set_name(string new_name)
{
     name = new_name;
}

void Student::set_major(string new_major)
{
     major = new_major;
}

string Student::get_major() const
{
     return major;
}

string Student::get_name() const
{
     return name;
}

int main(void)
{
     Student*  trans_student  =  new  Student("James  Smith","Computer
Science");
     Student* new_student = new Student();
     new_student->set_major("Computer Engineering");
     cout << (*new_student).get_major() <<"\n";
     cout <<"Name: " <<new_student.get_name() <<"\n";
     Student next_student = new Student();
     *(next_student.set_name("Jack Levenworth"));
     next_student.set_major("Computer Engineering");
     if (next_student->get_major() == "Computer Engineering")
          cout <<"Another computer engineer!\n";
     getch();
     return 0;
}
```

(b) Create a class called Student_Club, which will store information on student clubs. The Student_Club class should have as fields President, Vice-President, Secretary, Treasurer, all of which should be pointers to a student object. In this way, a student could hold offices in many different clubs and we would not have to keep separate information about that student in each instance of Student_Club. Your class definition should also include constructors as well as accessor functions. A sample main program is given below.

```
void main()
{
```

```
    Student *s1= new Student("Emily","computer engineering");
    Student *s2= new Student("George","mechanical engineering");
    Student *s3= new Student("Isabella","BBA");
    stu_club stu(s1,s2,s3,s1);
    cout<<"President is:"<<(stu.getPresident()).get_name()<<endl;
    cout<<"VP is:"<<(stu.getVP()).get_name()<<endl;
    cout<<"Secretary is:"<<(stu.getSecy()).get_name()<<endl;
    cout<<"Treasurer is:"<<(stu.getTreasurer()).get_name()<<endl;
    getch();
}
```

Question 2

Open the provided code in the file **Point23.cpp** which implements a point that may be two- or three-dimensional, i.e. some instances of the class have 2 coordinates, and some have 3. The coordinates are stored in an array of size 2 or 3 pointed to by a pointer dataPtr. The array is allocated dynamically (i.e. while the program is running) by a class constructor that creates a point.

There are 3 constructors for the class Point23: a constructor that takes two arguments creates a two-dimensional point, a constructor that takes three arguments creates a three-dimensional point, and the default constructor (with no arguments) creates a two-dimensional point with both coordinates initialized to 0s.

The class Point23 provides a class destructor that deallocates the array of coordinates.

Comments in the given program mark places where you need to add code for each of the exercises.

Exercise 1: Write code for dynamic allocation and initialization of the array of coordinates in all three constructors. The array should be pointed to by the pointer *dataPtr*.

Exercise 2: Write a line of code in the class destructor *~Point23()* that deallocates (i.e. deletes) the array allocated by the constructors.

Exercise 3: Write a line of code in *main()* to delete the point allocated by the command *new*.

This assignment is just an illustration of dynamic memory allocation. In real-life programs this is not be the best way to implement a 2/3-dimensional point, in particular because overhead of allocating memory dynamically is not worth saving memory for one integer!

Question 3

Create a class Vector with private data members :

```
float* list; //to point to an array of floats declared on the heap
int length;  //to store the length of the array
```

In the public section of the class,

- Provide a default constructor which initializes list to NULL and length to 0
- Provide a destructor to deallocate the memory allaocated in the following function
- Write a function **`void ReadList(int n)`** which creates a new array of size n and makes list point to that array. Length is set equal to n. The function runs a loop n times to ask the user for entering values in the list.

- Write a member function **sum** which returns a float value equal to the sum of the numbers in the list
- Write a function **squares()** which returns a pointer to a float array declared within the function. The size of this array should be equal to the size of the **list** array. The function computes the square of each element in the list, and stores these squares in the float array declared in the function. It returns a pointer to the first element in the squares array. The function returns NULL if the list is empty.

A sample main program which should work with your class is given below:

```
void main() {
   vector * v; float * sq;
   v = new vector;
   v->ReadList(5);
   cout << "\n\nSum = " << v->sum();
   sq = v->squares();
   if (sq != NULL) {
      cout << "\n\nSquares =";
      for (int i=0; i<5; i++) cout << " " << sq[i];
      // delete dynamic array created by function squares
      delete[] sq;}
   // delete object created by main
   delete v;

   getch();
}
```

Question 4:

Write a class `CheckoutCalculator` which behaves somewhat like the machine at the checkout counter in supermarkets. This calculator should ask for the number of items for which the total is to be calculated and then allow you to enter the price for every item. After entering all the items it displays all the prices entered and the total amount.

Here you need to use dynamic memory allocation since you do not know how many items will be there. Use an array to store the prices of items. In addition to any other methods, you must provide the following methods for your class

- Constructor
- Copy constructor
- Destructor
- Overloaded assignment operator

The output of the program should be somewhat like this:

```
Enter the number of items to enter:5
enter price for item 1:2
enter price for item 2:3
enter price for item 3:1
enter price for item 4:2
enter price for item 5:2
```

```
price for item 1:2
price for item 2:3
price for item 3:1
price for item 4:2
price for item 5:2
Total:10
```

Question 5:

Write a class `Results` which stores all the results in an array. Assume that we need to store only the results of a single semester which is four results. Each result will be final marks for a course between 0-100 and is stored in an integer array. Use dynamic memory allocation and the four methods mentioned above. Add separate methods to calculate the total marks for all courses and the average marks. Add a field for student ID.