CS 008
Lecture notes
3/5/24

1 Announcements

More office hours from 4-5 pm (Monday, Wednesday, Friday) on Zoom. Curriculum is being rewritten from scratch, feedback appreciated. Starting Thursday, there will be assigned readings before lecture instead of slides.

Note:

Please get the assigned textbook for free, you can talk to other classmates to figure out how to get it for free:

There is also a summer internship opportunity with Snapchat for community college students (snapacademies.org)

2 Bag class review

There is a section where you were instructed to make a keyed bag class. Use the built in C++ "pair" class *(See Main and Savitch, Section 2.6, page 85)*

Announcement

There are some modifications in the bag.h and bag.cpp files and some notes in Canvas: Look for "Lab 2: Bag Class w/ extra comments " in the week 2 module on Canvas. The original bag files could also be found in the textbook.

The bag class is basically a way to store unrelated items in one 'container'. The goal of the class is to act as a container class for objects. The bag class is considered a zero-dimensional container class because each piece of data is completely independent of each other. There is no structure among these items inside the bag. Also, the order in the bag does not matter but the state of the bag does.

2.1 Typedef

Question:

Why is typedef used in the bag class?

Answer: We use typedef because we may want to change this type later to accommidate a different datatype. Part of it is for ease of programmer development, but in addition, built in container classes often use value_type and size_type.

Warning:

Do not include (`using namespace std;`) in a header file.

Only use it in implementation files where it makes sense, putting it in the header file would force people to use it when they include your header file.

In the bag class, you may notice the data type size_t. This data type can only hold non-negative numbers and also guarantees that the values of the size_t type can hold the size of any variable that can be declared on your machine. Basically, size_t is a positive integer that changes it's size depending on the machine it is on.

2.2 Static

You might notice `static const size_type CAPACITY = 30;` in the bag class. **Static** allows a value to be 'static', it is the same in every instance of the class. Static variables is the one copy of a variable that is shared between all instances of a class. Basically, all instances (objects) of a class would reference the same value for a static variable, when changed it will be the new value for all instances.

The keyword **const** is added to make it a constant value that cannot be changed. In combination with static, the variable is shared between all instances of the class and is also constant.

Definition:

Static variables are variables that are shared between all instances of a class.

2.3 Inline definitions

Notice how in the bag class, the size method is written in only one line of code. For really small methods, you can write your method in one line. This is known as an **inline definition**.

Warning:

Do not use inline definitions for large methods.

2.4 Include

One of the methods that is used that is included in the program is **assert**. The purpose of assert is to check if a condition is true or false. If the condition is false, assert will create an error in the program.

Unlike an if-statement, assert will create an error in the program when the condition inside it is false. It kills the program with an error.

2.5 Pass by reference

In the bag.cpp file, the symbol (&) is used to make the parameter in the method pass-by-reference. With pass-by-reference, the parameter given in a method would be a reference of the parameter and not a copy of a parameter. With the addition of const, you would be passing in a reference to a variable that cannot be modified to a method.

2.6 std::copy

When you pass an array to a function, it becomes a pointer. It becomes a pointer to the function that points to the first element of an array. In the bag class, the overloading of the += operator uses std::copy. It's used to copy the array from the first element and then finds the element **used** amount away from the first element and then copies it over to after the *used* element in the array.

3 Additional stuff

3.1 Overloading an operator

There are 2 (maybe 2.5) different methods for overloading an operator.

- Overloading with a class member function
- Overloading with a nonmember function
- Overloading with a friend nonmember function

3.2 Sequence class

Sequence is another container class: Extra assignment since there's no time for the lecture