UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Faculty of Engineering, Built Environment & IT
University of Pretoria

COS110 - Program Design: Introduction

Practical 7 Specifications

Release Date: 06-10-2024 at 06:00

Due Date: 10-10-2024 at 23:59

Late Deadline: 10-10-2024 at 00:59

Total Marks: 122

Read the entire specification before starting with the practical.

Contents

1 General Instructions

- ***Read the entire assignment thoroughly before you begin coding.***
- This assignment should be completed individually.
- **Every submission will be inspected with the help of dedicated plagiarism detection software.**
- Be ready to upload your assignment well before the deadline. There is a late deadline which is 1 hour after the initial deadline which has a penalty of 20% of your achieved mark. **No extensions will be granted.**
- If your code does not compile, you will be awarded a mark of 0. The output of your program will be primarily considered for marks, although internal structure may also be tested (eg. the presence/absence of certain functions or structure).
- Failure of your program to successfully exit will result in a mark of 0.
- Note that plagiarism is considered a very serious offence. Plagiarism will not be tolerated, and disciplinary action will be taken against offending students. Please refer to the University of Pretoria's plagiarism page at `https://portal.cs.up.ac.za/files/departmental-guide/`.
- Unless otherwise stated, the usage of C++11 or additional libraries outside of those indicated in the assignment, will **not** be allowed. Some of the appropriate files that you have submit will be overwritten during marking to ensure compliance to these requirements. **Please ensure you use C++98**
- All functions should be implemented in the corresponding `cpp` file. No inline implementation in the header file apart from the provided functions.
- The usage of ChatGPT and other AI-Related software to generate submitted code is strictly forbidden and will be considered as plagiarism.

2 Overview

Template classes allow programmers to define class blueprints which work with different data types. This makes more flexible and reusable as the class does not need to be rewritten to work with different data types. This can be particularly powerful when used in conjunction with class hierarchies, sine template classes can serve as base or derived classes, enabling polymorphism while still taking advantage of the template features.

3 Template Compiling

There are many different approaches to compile and use templates in c++. For this practical you have to follow the following approach, as the header files will be overwritten on Fitchfork and will only work with this approach. The approach used is the approach on Slide 7 of Week 9 Lecture 3. This approach is:

- At the bottom of the header file (but before the #endif) include the implementation file.
- When you want to use a class in a different file, only include the header file.
- When compiling your code, don't compile the template classes.
- *Optional: In the implementation file you are allowed to include the header file at the top to allow auto complete to work for most IDE's.*

4 Your Task:

You are required to implement the following class diagram illustrated in Figure 1. Pay close attention to the function signatures as the `h` files will be overwritten, thus failure to comply with the UML, will result in a mark of 0.

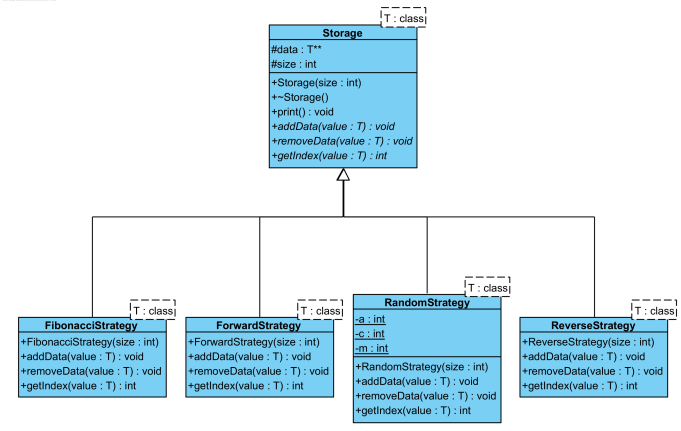

Figure 1: Class diagrams

Note, the importance of the arrows between the classes are not important for COS 110 and will be expanded on in COS 214.

4.1 Storage

This is an abstract class that defines the functions needed by all derived classes. It is also a template class. It will be used to simulate a basic storage device.

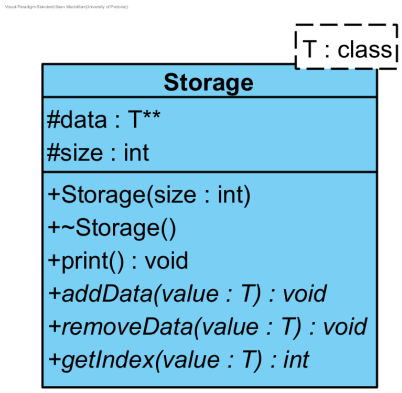

Figure 2: Storage UML

4.1.1 Members

- Member 1
  - This is a 1D dynamic array of dynamic objects.
  - It will store the data for this storage.
  - There will be gaps in the array, and this is represented using NULL.
- Member 2
  - This is the size of the 1D dynamic array.

4.1.2 Functions

- Function 1
  - This should intiliase both member variables using the passed-in size. The array should be filled with NULL.
- Function 2
  - This is a virtual function.
  - This should deallocate all dynamic memory used by the class.

- Function 3
  - This should print a string representation of the array.
  - The string should start with an open square brace, and end with a closing square brace and an endline.
  - The data should be printed out inside the square braces.
  - There should be a comma between the elements of the array. There should **NOT** be a comma after the last element.
  - If the data at an index is NULL, then print '-'.
- Function 4
  - This is a pure virtual function.
  - This function will add the passed-in data to the next empty spot in the array, by making a deep copy of the passed-in value.
  - Each derived class will have a different strategy to find the next empty spot.
- Function 5
  - This is a pure virtual function.
  - This function will remove the passed-in data from the array.
  - Each derived class will have a different strategy to search for the element.
- Function 6
  - This is a pure virtual function.
  - This function will return the index of the passed-in element.
  - Each derived class will have a different strategy to search for the element.

4.2 ForwardStrategy

This is a template class inheriting publically from the Storage class. It will traverse the array linearly starting from the front of the array.

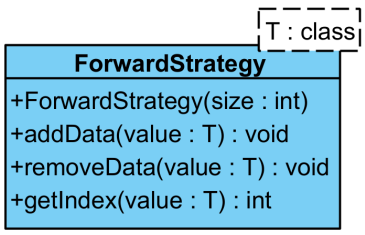

Figure 3: ForwardStrategy UML

4.2.1 Functions

- Function 1
  - Call the parent constructor using the passed-in parameter.
- Function 2
  - This is a virtual function.
  - This function will add the passed-in value at the next empty spot in the array.
  - To find the next empty spot, start at the front of the array, and traverse to the back.
  - If an empty spot cannot be found, then don't add the passed-in value.
- Function 3
  - This is a virtual function.
  - This function will remove the passed-in value from the array. Only remove the first occurrence of the data.
  - To search for the passed-in value, start at the front of the array, and traverse to the back.
  - If the passed-in value is found, then delete that index in the array, and set it equal to NULL.
- Function 4
  - This is a virtual function.
  - This function will return the index of the first occurrence of the passed-in value in the array.
  - To search for the passed-in value, start at the front of the array, and traverse to the back.
  - If the passed-in value is found, return that index of the first matching value.
  - If the passed-in value was not found, return -1.

4.3 ReverseStrategy

This is a template class inheriting publically from the Storage class. It will traverse the array linearly starting from the back of the array.

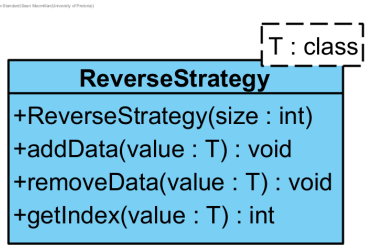

Figure 4: ReverseStrategy UML

4.3.1 Functions

- Function 1
  - Call the parent constructor using the passed-in parameter.
- Function 2
  - This is a virtual function.
  - This function will add the passed-in value at the next empty spot in the array.
  - To find the next empty spot, start at the back of the array, and traverse to the front.
  - If an empty spot cannot be found, then don't add the passed-in value.
- Function 3
  - This is a virtual function.
  - This function will remove the passed-in value from the array. Only remove the first occurrence of the data.
  - To search for the passed-in value, start at the back of the array, and traverse to the front.
  - If the passed-in value is found, then delete that index in the array, and set it equal to NULL.
- Function 4
  - This is a virtual function.
  - This function will return the index of the first occurrence of the passed-in value in the array.
  - To search for the passed-in value, start at the back of the array, and traverse to the front.
  - If the passed-in value is found, return that index of the first matching value.
  - If the passed-in value was not found, return -1.

4.4 RandomStrategy

This is a template class inheriting publically from the Storage class. It will use a random number generator to traverse the array. This will be implemented using a Lienar Congruential Generator which mimics the random number generator found in `gcc`.

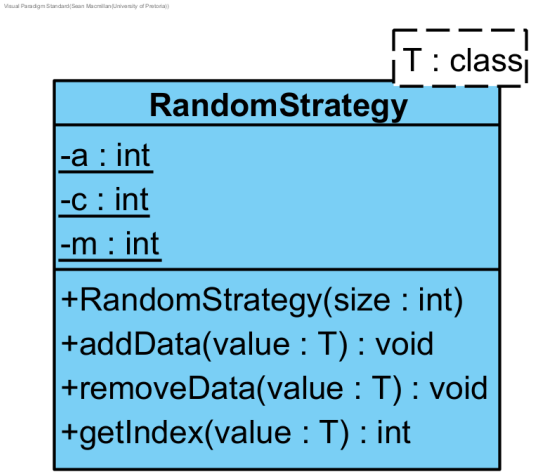

Figure 5: RandomStrategy UML

4.4.1 Members

- Member 1
  - This is a variable used by the random number generator.
  - Initialise this variable to 1103515245.
- Member 2
  - This is a variable used by the random number generator.
  - Initialise this variable to 12345.
- Member 3
  - This is a variable used by the random number generator.
  - Initialise this variable to 2147483648.

4.4.2 Functions

- Function 1
  - Call the parent constructor using the passed-in parameter.
- Function 2
  - This is a virtual function.
  - This function will add the passed-in value at the next empty spot in the array.
  - To find the next empty spot, use the random number algorithm explained in Section 4.4.3.
  - If an empty spot cannot be found, then don't add the passed-in value.
- Function 3
  - This is a virtual function.
  - This function will remove the passed-in value from the array. Only remove the first occurrence of the data.
  - To search for the passed-in value, use the random number algorithm explained in Section 4.4.3.
  - If the passed-in value is found, then delete that index in the array, and set it equal to NULL.
- Function 4
  - This is a virtual function.
  - This function will return the index of the first occurrence of the passed-in value in the array.
  - To search for the passed-in value, use the random number algorithm explained in Section 4.4.3.
  - If the passed-in value is found, return that index of the first matching value.
  - If the passed-in value was not found, return -1.

4.4.3 Generating random numbers

- This random number generator works by using a sequence starting from an initial value, called a seed. The seed for our algorithm will be the size of the array.
- Every time a function needs a random number the sequence will reset, to ensure that we can find the elements again.
- The random value is stored in an int variable called r. This value is initialised to the size of the array. When using the random value to look up values in the array, take the variable r and use the modulus operator with the size of the array to cap the value to a valid index.

- Every time you need a new random value, update r using the formula

$$r = |(a \times r + c)\%m|$$

- To prevent infinite loops, the random number generator should only generate $3 *$ array size numbers.
- Example:
  - Create a RandomStrategy object with a size of 2. Call the getIndex function with any input parameter. Since there are no elements in the array, it will generate the maximum number of random numbers and then return -1. The generated random numbers are:

Iteration	r	index
1	2087924461	1
2	1495329502	0
3	4707519	1
4	203552652	0
5	2043986219	1
6	78875016	0

4.5 FibonacciStrategy

This is a template class inheriting publically from the Storage class. It will use the Fibonacci sequence to traverse the array.

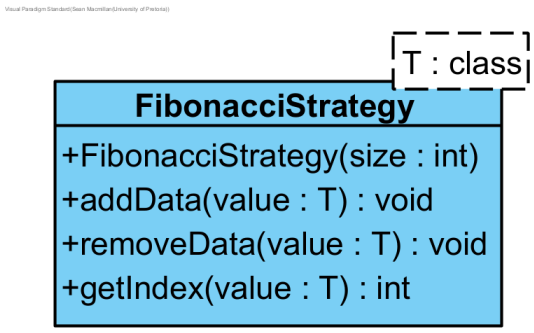

Figure 6: FibonacciStrategy UML

4.5.1 Functions

- Function 1
  - Call the parent constructor using the passed-in parameter.
- Function 2
  - This is a virtual function.
  - This function will add the passed-in value at the next empty spot in the array.
  - To find the next empty spot, use the algorithm explained in Section 4.5.2.
  - If an empty spot cannot be found, then don't add the passed-in value.
- Function 3
  - This is a virtual function.
  - This function will remove the passed-in value from the array. Only remove the first occurrence of the data.
  - To search for the passed-in value, use the algorithm explained in Section 4.5.2.
  - If the passed-in value is found, then delete that index in the array, and set it equal to NULL.
- Function 4
  - This is a virtual function.
  - This function will return the index of the first occurrence of the passed-in value in the array.
  - To search for the passed-in value, use the algorithm explained in Section 4.5.2.
  - If the passed-in value is found, return that index of the first matching value.
  - If the passed-in value was not found, return -1.

4.5.2 Generating Fibonacci numbers

- This sequence generator will use the Fibonacci sequence to pick the traversal order. If you need a reminder about the Fibonacci sequence please read up about it here.
- Every time a function needs a number the sequence will reset, to ensure that we can find the elements again. When using the Fibonacci number to look up values in the array, take the variable and use the modulus operator with the size of the array to cap the value to a valid index.
- To prevent infinite loops, the number generator should only generate $3 *$ array size numbers.
- Example:
  - Create a FibonacciStrategy object with a size of 2. Call the getIndex function with any input parameter. Since there are no elements in the array, it will generate the maximum number of numbers and then return -1. The generated numbers are:

Iteration	v	index
1	0	0
2	1	1
3	1	1
4	2	0
5	3	1
6	5	1

5 Memory Management

As memory management is a core part of COS110 and C++, each task on FitchFork will allocate approximately 10% of the marks to memory management. The following command is used:

```
valgrind --leak-check=full ./main
```

Please ensure, at all times, that your code *correctly* de-allocates *all* the memory that was allocated.

6 Testing

As testing is a vital skill that all software developers need to know and be able to perform daily, 10% of the assignment marks will be allocated to your testing skills. To do this, you will need to submit a testing main (inside the `main.cpp` file) that will be used to test an Instructor Provided solution. You may add any helper functions to the main.cpp file to aid your testing. In order to determine the coverage of your testing the gcov [1] tool, specifically the following version *gcov (Debian 8.3.0-6) 8.3.0*, will be used. The following set of commands will be used to run gcov:

```
g++ --coverage *.cpp -o main
./main
gcov -f -m -r -j ${files}
```

This will generate output which we will use to determine your testing coverage. The following coverage ratio will be used:

$$\frac{\text{number of lines executed}}{\text{number of source code lines}}$$

We will scale this ration based on class size.

The mark you will receive for the testing coverage task is determined using Table 1:

Coverage ratio range	% of testing mark
0%-5%	0%
5%-20%	20%
20%-40%	40%
40%-60%	60%
60%-80%	80%
80%-100%	100%

Table 1: Mark assignment for testing

Note the top boundary for the Coverage ratio range is not inclusive except for 100%. Also, note that only the functions stipulated in this specification will be considered to determine your testing

[1] For more information on gcov please see `https://gcc.gnu.org/onlinedocs/gcc/Gcov.html`

mark. Remember that your main will be testing the Instructor Provided code and as such, it can only be assumed that the functions outlined in this specification are defined and implemented.

As you will be receiving marks for your testing main, we will also be doing plagiarism checks on your testing main.

7 Implementation Details

- You must implement the functions in the header files exactly as stipulated in this specification. Failure to do so will result in compilation errors on FitchFork.
- You may only use **c++98**.
- You may only utilize the specified libraries. Failure to do so will result in compilation errors on FitchFork.
- You may only use the following libraries:
  - cmath
  - iostream
  - sstream
- You are supplied with a **trivial** main demonstrating the basic functionality of this assessment.

8 Upload Checklist

The following c++ files should be in a zip archive named uXXXXXXXX.zip where XXXXXXXX is your student number:

- Storage.cpp
- ForwardStrategy.cpp
- ReverseStrategy.cpp
- RandomStrategy.cpp
- FibonacciStrategy.cpp
- `main.cpp`
- Any textfiles used by your `main.cpp`
- `testingFramework.h` and `testingFramework.cpp` if you used these files.

The files should be in the root directory of your zip file. In other words, when you open your zip file you should immediately see your files. They should not be inside another folder.

9 Submission

You need to submit your source files on the FitchFork website (`https://ff.cs.up.ac.za/`). All methods need to be implemented (or at least stubbed) before submission. Your code should be able to be compiled with the following command:

```
g++ -Werror -Wall *.cpp -o main
```

and run with the following command:

```
./main
```

Remember your `h` file will be overwritten, so ensure you do not alter the provided `h` files.
You have 10 submissions and your best mark will be your final mark. Upload your archive to the Practical 7 slot on the FitchFork website. If you submit after the deadline but before the late deadline, a 20% mark deduction will be applied. **No submissions after the late deadline will be accepted!**