Livella gratillo
Doct rexytiner
lespelitus otaratio eterniarea

CHHC e@ma.il

2023-12-06

Contents

1. Homework

1.1. h7

1.1.1. Ex3. File I/O in C++

- File, string, standard input/output can all be converted into stream
- Similar to `FILE*` in C
- DO NOT forget to check whether the files are correctly opened
- DO NOT forget to close the files

Think: How to read a line that contains numbers separated by spaces?

Think: How to read a line that contains numbers separated by spaces?

You can still manually read a line and convert the numbers. Or?

One cpp-style-way is to take the advantages of `std::getline` and `std::istringstream`.

```
// Read a line from input. input can be a file or standard input.

std::string line;
std::getline(input, line);
// Now line is one line
std::istringstream iss(line);

// Read numbers from iss
while(iss >> num) {
    // Do something with num
}
```

Think: How to detect input errors?

For example, if we use `cin>>a;` where `a` is an int, but the real input is not an integer.

Maybe you can try `cin.fail()`, `cin.clear()` and `cin.ignore()`?

1.1.2. Ex4. Basic Programming

$$\begin{cases} u_0 = a \\ u_{i+1} = \begin{cases} \frac{1}{2}u_i & \text{if } n \text{ is even} \\ 3u_i+1 & \text{if } n \text{ is odd} \end{cases} \end{cases}$$

- Basic logic questions
- Recall what is recursion
- Think about Fibonacci sequence. How to find the n-th Fibonacci number?

1.1.3. Ex5. From C to C++

Here are some questions you should think of:

- What is `class` in C++?
- What is OOP?
- What is polymorphism in C++?
- What is abstract class in C++?
- What is `virtual` function in C++? What is virtual destructor and when to use it?
  - This answer might be of help.
- What are the common containers in STL? How to use them?

1.2. h8

1.2.1. Ex1. Containers

Basic C++ Containers library & Data structure. Make full use of cppreference.

Parameter `T` inside `<>` is called **template parameter**, deduced when compilation.

- `std::array<T, N>`: static contiguous array
- `std::vector<T>`: dynamic contiguous array. **Random access** is supported.
- `std::stack<T>`: a container providing stack (LIFO data structure)
- `std::queue<T>`: a container providing queue (FIFO data structure)

```cpp
#include <array>
#include <iostream>
#include <queue>
#include <stack>
#include <string>
#include <vector>

using std::array;
using std::cin;
using std::cout;
using std::queue;
using std::stack;
using std::string;
using std::vector;
// Instead of "using namespace std;"

void ex1_reverse_array()
{
    array<string, 10000> re;
    string word;
    size_t count = 0;
    while (cin >> word)
        re[count++] = word;
    for (auto iter = re.cbegin() + count - 1; iter != re.cbegin() - 1;
--iter) {
        // do something with *iter
    }
}
void ex1_reverse_vector()
{
    vector<string> re;
    string word;
    while (cin >> word)
        re.push_back(word);
    for (auto iter = re.crbegin(); iter != re.crend(); ++iter) {
        // do something with *iter
    }
}
void ex1_reverse_stack()
{
    stack<string> re;
    string word;
    while (cin >> word)
        re.push(word);
    for (; !re.empty(); re.pop()) {
        // do something with re.top()
    };
}
void ex1_ordered_queue()
{
    queue<string> re;
    string word;
    while (cin >> word)
        re.push(word);
    for (; !re.empty(); re.pop()) {
        // do something with re.front()
    };
}
```

1.2.2. Ex2. Class Implementation

- Basic inheritance & polymorphism
- Basic drawing in OpenGL
- Draw hierarchy diagram

1.2.3. Ex3. Classes and OpenGL

- Basic animation in OpenGL
- Draw different figures in OpenGL
  - Line
  - Rectangle
  - Triangle
  - Polynomial
  - Circle
- Combination of classes

2. Lab

Refer to lab materials! Basically the most important things are those that appear repeatedly in every lab section ;D

2.1. lab9

- What is stack?
- What is Postfix Expression?
- How to use stack and implement the *Shunting Yard algorithm* ?
- Basic usage of `std::stack`

2.2. lab10

Basic class design:

- Methods
- Attributes
- Hierarchy (public, protected, private)
- Abstract class (virtual functions)
- Polymorphism (optional? depends on what Prof. said in lecture)

Avoid diamond structure in class design!

Some design patterns (optional!):

- Singleton
- Factory Method
- Observer
- Adapter

Reference

- Cralia sarytie
- *Uaractive contactina maleficio*

Break?