VG101: Introduction to Computer and Programming

Final Review

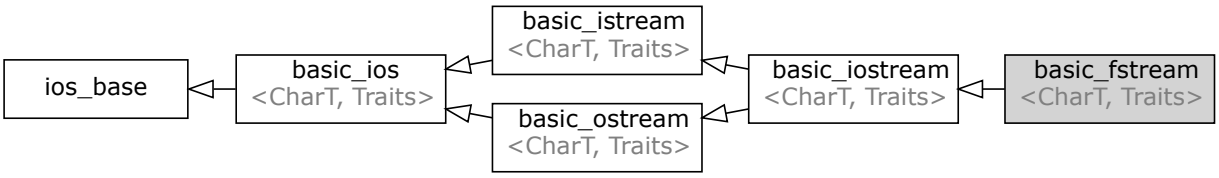
About Final Exam

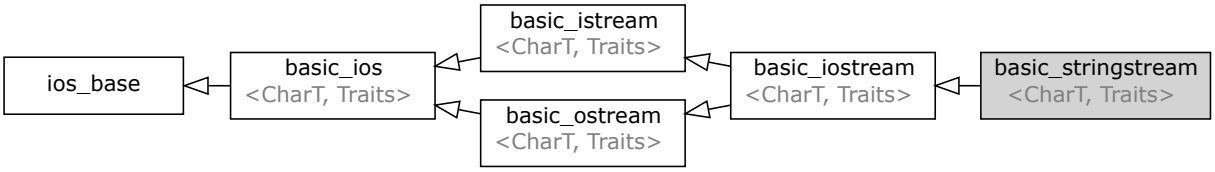
- 100-minute open-book computer-based exam
- On `C++`, though compatible `C` code is allowed
- Weight a lot in final grade
- Read the instructions first
- Put this into your web browser bookmark: https://en.cppreference.com/

`class`

- `pulic`, `private`, and `protected`
- `namespace`: `void MyClass::func()`
- ctor and dtor
- function/operator overloading

I/O





- `operator>>` for `stream` will ignore the white space
- memberfunction `get()` to extract a single char: `cin.get()`
- `iostream`: `cin`, `cout`

```
char c;
int i;
string str;
cin >> c >> i >> str;
cout << str << i << c;
```

- `fstream`: read binary file or text file

```cpp
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

int main()
{
    fstream file;
    unsigned char i;
    file.open("test.dat", ios::out | ios::binary);

    for (i=0; i<20; i++)
        file.write( reinterpret_cast<char*>(&i), 1);

    file.close();
    return 0;
}
```

```cpp
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    std::ifstream file1("test1.txt");
    if (file1.is_open()) // use it same as `cin`
    {
        std::string line;
        while (getline(file1, line))
            cout << line << endl;
        file1.close();
    }

    std::ifstream file2("test2.txt");
    if (file2.is_open()) // use it same as `cin`
    {
        std::string first_line;
        file2 >> first_line;
        cout << first_line;
        file2.close();
    }
    // you may handle the case that file open fail by `else`
    // file open may fail if couldn't find the file, etc
    return 0;
}
```

- `stringstream`: separate words

```cpp
#include <iostream>
```

```
#include <string>
#include <sstream>  // library required for stringstream
using namespace std;
int main()
{
    string word1, word2, word3;
    string line;
    // input: "VG101 hello world"
    getline(cin, line); // now `line`: "VG101      hello world\n"
    stringstream ss(line); // ctor of stringstream: take a string as the
parameter
    ss >> word1 >> word2 >> word3;
        // word1: "VG101"
        // word2: "hello"
        // word3: "world"
}
```

Reference and `const`

- Reference: `int i=0; int& r = i; r++;`
- `const int* px` vs. `int* const px`

```
#include <iostream>
using namespace std;
const int* bar()
{
    static int winky = 5;
    cout << "winky= " << winky << endl;
    return(&winky);
}

int main()
{
    const int *pinky = bar();
    // *pinky = 6; // will cause an error
    bar();
    return 0;
}
```

`new` and `delete`

- `new` : `int* px = new int`
- `new` array: `int* parray = new int[10]`
- `delete` : `delete px`
- `delete` array: `delete[] parray`
- Always `delete` all the element you `new`, memory leak will result in deduction

STL `vector` and `string`

- ctor
- `push_back`
- `empty`
- `size`
- iterator
- What's the problem of the following code?

```
#include <iostream>
#include<vector>
using namespace std;
int main(void)
{
    vector<int>array;
    array.push_back(100);
    array.push_back(300);
    array.push_back(300);
    array.push_back(300);
    array.push_back(300);
    array.push_back(500);
    vector<int>::iterator itor;
    for(itor=array.begin();itor!=array.end();itor++)
        if(*itor==300)
            itor=array.erase(itor);

    for(itor=array.begin();itor!=array.end();itor++)
        cout << *itor << " ";
    cout << endl;
    // output: 100 300 300 500
  return 0;
}
```

STL `algorithm`

You can definitely finish exam without the functions discussed here, but they could make your life easier

- `find` : https://en.cppreference.com/w/cpp/algorithm/find
- `count` : https://en.cppreference.com/w/cpp/algorithm/count
- `max_element/min_element` : https://en.cppreference.com/w/cpp/algorithm/max_element / https://en.cppreference.com/w/cpp/algorithm/min_element
- `swap` : https://en.cppreference.com/w/cpp/algorithm/swap
- `reverse` : https://en.cppreference.com/w/cpp/algorithm/reverse
- `sort` : https://en.cppreference.com/w/cpp/algorithm/sort