Examination EDAF50 (EDA031) C++ Programming

2018–03–14, 14:00–19:00

Aid at the exam: one C++ book. *Not allowed:* printed copies of the lecture slides or other papers.

Assessment (preliminary): the questions give $8 + 10 + 12 + 12 + 8 = 50$ points. You need 25 points for a passing grade (3/25, 4/33, 5/42).

You must show that you know C++ and that you can use the C++ standard library. "C solutions" don't give any points, even if they are correct.

Free-text anwers should be concise but complete, well motivated and written clearly, to the point, and in complete sentences.

Please write on only one side of the paper and hand in your solutions with the papers sorted and facing the same way, as the solutions may be scanned for the marking.

---

1. We want a function template `apply_all` that operates on a sequence of binary function objects (given as an iterator range), and calls each of the functions with the two given arguments. The result of each call is written to an `OutputIterator`.
   The use of `apply_all` is illustrated by the following code example:

```
#include <iostream>
#include <vector>
#include <functional>
#include "apply_all.h"

void use()
{
    std::vector<std::function<int(int,int)>> fs;  // the functions

    fs.push_back(std::plus<int>{});
    fs.push_back([](int x, int y) {return 2*x-y;});
    fs.push_back(std::divides<int>{});

    std::vector<int> rs(fs.size());  // where the results will be stored

    apply_all(begin(fs), end(fs), begin(rs), 10, 2);

    for(auto r : rs) std::cout << r << " ";
    std::cout << endl;
}
```

   When executed, this code should print `12 18 5`.

   Implement the function template `apply_all`. In the example, the functions are $(int, int) \to int$, but for full marks your implementation should handle the general case with functions $(A, B) \to C$, for some types A, B, and C. (I.e., the sequence can be a `std::vector<std::function<C(A,B)>>`.) You may assume that `apply_all` is called correctly, that the provided iterators are valid, and that all the functions in the sequence can be called with the provided arguments.

2. The following program does not work as expected.

```
 1 #include <cstring>
 2 #include <vector>
 3 #include <iostream>
 4 #include <algorithm>
 5 using std::cout;
 6 using std::endl;
 7
 8 class Foo{
 9 public:
10     Foo() :Foo(-1, "Foo") {}
11     Foo(int x, const char* cs) :v{x},s{new char[std::strlen(cs)+1]} {std::strcpy(s,cs);}
12     ~Foo() {delete[](s);}
13     std::string get_s() const {return  std::string(s);}
14     int get_v() const {return v;}
15 private:
16     int v;
17     char* s;
18 };
19
20
21 int main()
22 {
23     std::vector<Foo> v;
24     v.reserve(4);
25     v.emplace_back(1, "Hello");
26     v.emplace_back(2, "World");
27     v.emplace_back(3, "Testing");
28     v.emplace_back(5, "Testing");
29
30     auto res = std::find_if(begin(v), end(v), [](Foo f){return f.get_v() == 2;});
31
32     if(res != v.end()){
33         cout << "found " << res->get_s() << endl;
34     } else {
35         cout << "not found" << endl;
36     }
37 }
```

When executed, the output was `Found P0!` instead of the expected `Found World`.
To investigate, the program is executed under valgrind, which complains about *invalid reads* and *invalid* `free() / delete / delete[] / realloc()` as seen in the valgrind report on page 3 .

a) Explain why the program fails. Describe the failure, what the problem with the code is, and how this problem causes the failure.

b) Is it possible to fix the problem, so that `main()` executes as expected, by changing *only the function* `main()`?
If yes, show how. Otherwise, motivate.

c) Is it possible to fix the problem, so that `main()` executes as expected, by changing *only the class* `Foo`?
If yes, show how. Otherwise, motivate.

The scope of the question is limited to what is called from (or affecting the execution of) `main()`. *You do not need to consider other aspects or possible problems with the code, but the changes you make to the code should be motivated by your answer to a).*

Valgrind error report. Some template parameters and lines referring to files in the standard library have been omitted.)

```
==29979== Memcheck, a memory error detector
==29979== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==29979== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==29979== Command: ./program
==29979==
==29979== Invalid read of size 1
==29979==    at 0x4C2C1A2: strlen (vg_replace_strmem.c:412)
==29979==    by 0x4EF5C30: std::basic_string<char, ... >::basic_string(char const*, ...> const&) (in /usr/lib/libstdc++.so.6.0.20)
==29979==    by 0x40143D: Foo::get_s() const (program.cc:13)
==29979==    by 0x400FD6: main (program.cc:33)
==29979==  Address 0x5a02110 is 0 bytes inside a block of size 6 free'd
==29979==    at 0x4C2A8E0: operator delete[](void*) (vg_replace_malloc.c:542)
==29979==    by 0x401402: Foo::~Foo() (in program)
==29979==    by 0x4012F8: bool __gnu_cxx::__ops::_Iter_pred<main::{lambda(Foo)#1}>::operator()<...>(...) (predefined_ops.h:231)
==29979==    by 0x401190: __gnu_cxx::__normal_iterator<Foo*, ...> std::__find_if<...>(...) (stl_algo.h:124)
             (---)
==29979==    by 0x400F89: main (program.cc:30)
==29979==
==29979== Invalid read of size 1
==29979==    at 0x4C2C1B4: strlen (vg_replace_strmem.c:412)
==29979==    by 0x4EF5C30: std::basic_string<char, ... >::basic_string(char const*, ...> const&) (in /usr/lib/libstdc++.so.6.0.20)
==29979==    by 0x40143D: Foo::get_s() const (program.cc:13)
==29979==    by 0x400FD6: main (program.cc:33)
==29979==  Address 0x5a02111 is 1 bytes inside a block of size 6 free'd
==29979==    at 0x4C2A8E0: operator delete[](void*) (vg_replace_malloc.c:542)
==29979==    by 0x401402: Foo::~Foo() (in program)
==29979==    by 0x4012F8: bool __gnu_cxx::__ops::_Iter_pred<main::{lambda(Foo)#1}>::operator()<...>(...) (predefined_ops.h:231)
==29979==    by 0x401190: __gnu_cxx::__normal_iterator<...> std::__find_if<...>(...) (stl_algo.h:124)
             (---)
==29979==    by 0x400F89: main (program.cc:30)
==29979==
==29979== Invalid read of size 2
==29979==    at 0x4C2D988: memcpy@@GLIBC_2.14 (vg_replace_strmem.c:915)
==29979==    by 0x4EF582F: char* std::string::_S_construct<char const*>(char const*, char const*, ...) (in /usr/lib/libstdc++.so.6.0.20)
==29979==    by 0x4EF5C45: std::basic_string<char, ... >::basic_string(char const*, ...> const&) (in /usr/lib/libstdc++.so.6.0.20)
==29979==    by 0x40143D: Foo::get_s() const (program.cc:13)
==29979==    by 0x400FD6: main (program.cc:33)
==29979==  Address 0x5a02110 is 0 bytes inside a block of size 6 free'd
==29979==    at 0x4C2A8E0: operator delete[](void*) (vg_replace_malloc.c:542)
==29979==    by 0x401402: Foo::~Foo() (in program)
==29979==    by 0x4012F8: bool __gnu_cxx::__ops::_Iter_pred<main::{lambda(Foo)#1}>::operator()<...>(...) (predefined_ops.h:231)
==29979==    by 0x401190: __gnu_cxx::__normal_iterator<...> std::__find_if<...>(...) (stl_algo.h:124)
             (---)
==29979==    by 0x400F89: main (program.cc:30)
==29979==
==29979== Invalid read of size 1
==29979==    at 0x4C2D9C0: memcpy@@GLIBC_2.14 (vg_replace_strmem.c:915)
==29979==    by 0x4EF582F: char* std::string::_S_construct<char const*>(char const*, char const*, ...) (in /usr/lib/libstdc++.so.6.0.20)
==29979==    by 0x4EF5C45: std::basic_string<char, ... >::basic_string(char const*, ...> const&) (in /usr/lib/libstdc++.so.6.0.20)
==29979==    by 0x40143D: Foo::get_s() const (program.cc:13)
==29979==    by 0x400FD6: main (program.cc:33)
==29979==  Address 0x5a02114 is 4 bytes inside a block of size 6 free'd
==29979==    at 0x4C2A8E0: operator delete[](void*) (vg_replace_malloc.c:542)
==29979==    by 0x401402: Foo::~Foo() (in program)
==29979==    by 0x4012F8: bool __gnu_cxx::__ops::_Iter_pred<main::{lambda(Foo)#1}>::operator()<...>(...) (predefined_ops.h:231)
==29979==    by 0x401190: __gnu_cxx::__normal_iterator<...> std::__find_if<...>(...) (stl_algo.h:124)
             (---)
==29979==    by 0x400F89: main (program.cc:30)
==29979==
==29979== Invalid free() / delete / delete[] / realloc()
==29979==    at 0x4C2A8E0: operator delete[](void*) (vg_replace_malloc.c:542)
==29979==    by 0x401402: Foo::~Foo() (in program)
==29979==    by 0x402591: void std::_Destroy<Foo>(Foo*) (stl_construct.h:93)
==29979==    by 0x4022E5: void std::_Destroy_aux<false>::__destroy<Foo*>(Foo*, Foo*) (stl_construct.h:103)
==29979==    by 0x401FD2: void std::_Destroy<Foo*>(Foo*, Foo*) (stl_construct.h:126)
==29979==    by 0x401904: void std::_Destroy<Foo*, Foo>(Foo*, Foo*, std::allocator<Foo>&) (stl_construct.h:151)
==29979==    by 0x4014D0: std::vector<Foo, std::allocator<Foo> >::~vector() (stl_vector.h:424)
==29979==    by 0x40103A: main (program.cc:36)
==29979==  Address 0x5a020c0 is 0 bytes inside a block of size 6 free'd
==29979==    at 0x4C2A8E0: operator delete[](void*) (vg_replace_malloc.c:542)
==29979==    by 0x401402: Foo::~Foo() (in program)
==29979==    by 0x4012F8: bool __gnu_cxx::__ops::_Iter_pred<main::{lambda(Foo)#1}>::operator()<...>(...) (predefined_ops.h:231)
==29979==    by 0x401167: __gnu_cxx::__normal_iterator<...> std::__find_if<...>(...) (stl_algo.h:120)
             (---)
==29979==    by 0x400F89: main (program.cc:30)
==29979==
==29979==
==29979== HEAP SUMMARY:
==29979==     in use at exit: 0 bytes in 0 blocks
==29979==   total heap usage: 6 allocs, 8 frees, 122 bytes allocated
==29979==
==29979== All heap blocks were freed -- no leaks are possible
==29979==
==29979== For counts of detected and suppressed errors, rerun with: -v
==29979== ERROR SUMMARY: 11 errors from 5 contexts (suppressed: 0 from 0)
```

3. You are in the process of implementing a `Vector` data structure that has the following fundamental representation:

```
#include <memory>
template <typename T>
class Vector{
public:
    Vector() =default;
    Vector(int size) :sz(size),elem(new T[size]) {}
private:
    int sz{0};
    std::unique_ptr<T[]> elem{nullptr};
};
```

   You are doing test-driven development and the expected behaviour is defined by unit tests:

```
#include <cassert>
template <typename T>
void test_assign(Vector<T>& v, const T& val1, const T& val2)
{
    assert(v.size() >= 2);
    assert(val1 != val2); // test sanity checks

    v[0] = val1;
    assert(v[0] == val1);
    v[1] = val2;
    assert(v[1] == val2);
    assert(v[0] != v[1]);
}

template <typename T>
void test_read(const Vector<T>& v, const T& val1, const T& val2)
{
    assert(v.size() >= 2);
    assert(v[0] == val1);
    assert(v[1] == val2);
}

int main()
{
    Vector<int> v(2);
    auto val1 = 17;
    auto val2 = 42;
    test_assign(v, val1, val2);
    test_read(v, val1, val2);
    std::cout << "Tests passed."<<std::endl;
}
```

   a) Add the required functionality to `Vector` so that the above tests pass.
   b) Add the required functionality to `Vector` to make the following `test_create()` pass.

```
void test_create()
{
    Vector<int> v1(5);
    assert(v1.size() == 5);
    for(int i=0; i< v1.size(); ++i){
        v1[i]=i;
    }
    Vector<int> v2{0,1,2,3,4};
    assert(v2.size() == v1.size());
    assert(v1 == v2);
}
```

   Note: If you need a raw pointer to the array held by the `unique_ptr`, you can use `elem.get()`.

4. A reverse dictionary is a dictionary where the words are sorted after their last letters. Words that end in `a` appear before words that end in `b`, words that end in `ma` appear before words that end in `na`, and so on. A reverse dictionary can be useful for crossword solvers and for poets who look for words that rhyme.

   The file *words.txt* contains a number of English words separated by whitespace. Write a program that reads the file, creates a reverse dictionary and writes it (one word per line) to the file *backwords.txt*. Duplicate words should be eliminated and uppercase letters should be converted to lowercase. You *must* use the standard library where possible — avoid `for` and `while` statements.

   Example input and output:

   words.txt:

```
africa
america
angelica
antarctica
basilica
britannica
corsica
dominica
erica
formica
harmonica
jamaica
jessica
metallica
mica
monica
patrica
pica
replica
replica
sciatica
seneca
silica
spica
veronica
yucca
```

   backwords.txt:

```
yucca
seneca
jamaica
angelica
silica
basilica
metallica
replica
mica
formica
dominica
britannica
monica
harmonica
veronica
pica
spica
erica
america
africa
patrica
corsica
jessica
sciatica
antarctica
```

   Hint: The algorithm `std::lexicographical_compare` has the following declaration

```
template< class InputIterator1, class InputIterator2 >
bool lexicographical_compare( InputIterator1 first1, InputIterator1 last1,
                              InputIterator2 first2, InputIterator2 last2 );

template< class InputIterator1, class InputIterator2, class Compare >
bool lexicographical_compare( InputIterator1 first1, InputIterator1 last1,
                              InputIterator2 first2, InputIterator2 last2,
                              Compare comp );
```

   where the first version compares the ranges using `operator<`, and the second uses the supplied comparator.

   The C function `int tolower(int ch)` converts an uppercase letter to lowercase; if `ch` isn't uppercase it returns the character unchanged. Like all other functions from <cctype>, the behavior of std::tolower is undefined if the argument's value is neither representable as unsigned char nor equal to EOF. To use these functions safely with plain chars (or signed chars), the argument should first be converted to unsigned char: Similarly, they should not be directly used with standard algorithms when the iterator's value type is char or signed char. Instead, convert the value to unsigned char first.

5. The following code fragment does not compile.

```
 1  class Foo{
 2  public:
 3      Foo(int i) {x = i;}
 4  private:
 5      int x;
 6  };
 7
 8  class Bar{
 9  public:
10      Bar(int i) {a = i;}
11  private:
12      Foo a;
13  };
```

The compiler error is

```
foobar.cc: In constructor ’Bar::Bar(int)’:
foobar.cc:10:16: error: no matching function for call to ’Foo::Foo()’
     Bar(int i) {a = i;}
                ^
foobar.cc:10:16: note: candidates are:
foobar.cc:3:5: note: Foo::Foo(int)
     Foo(int i) {x = i;}
     ^
foobar.cc:3:5: note:   candidate expects 1 argument, 0 provided
foobar.cc:1:7: note: Foo::Foo(const Foo&)
 class Foo{
       ^
foobar.cc:1:7: note:   candidate expects 1 argument, 0 provided
```

a) Explain why the compiler gives this error.

b) Change the code fragment to make it correct.