Examination
EDAF50 C++ Programming

2026-06-04, 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 $15 + 6 + 11 + 11 + 7 = 50$ 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, and idiomatic C++ solutions that reimplement standard library facilities may give deductions, even if they are correct.
In solutions, resource management must also be considered when relevant – your solutions must not leak memory.
Free-text anwers should be concise but complete, well motivated and written clearly, to the point, and in complete sentences. Answers may be given in swedish or english.

For all problems, you may choose to answer "I don't know", which will be worth 20% of the maximum score of that problem. If you opt to do so, the sentence "I don't know." or "Jag vet inte." must be clearly given as the only answer to that problem. (I.e., you will not get any credit for an answer "I don't know" to a subproblem.)

An appendix on the last page describes some standard library components appearing in the problems.

Please write on only one side of the paper and hand in your solutions with the papers numbered, sorted and facing the same way. Please make sure to write your anonymous code and personal identifier on each page, and that the papers are not folded or creased, as the solutions may be scanned for the marking.

1. The following program does not work correctly:

```
 1  #include <iostream>
 2  #include <algorithm>
 3  #include <cstring>
 4  #include <vector>
 5  using std::cout;
 6
 7  class person {
 8    public:
 9      person(const char* the_name, int the_id)
10          : name{new char[strlen(the_name) + 1]}, id{the_id} {
11          strcpy(name, the_name);
12      }
13
14      ~person() { delete[] name; }
15
16      person(const person& o) :name{new char[strlen(o.name)+1]}, id{o.id} {
17          strcpy(name, o.name);
18      }
19
20      const char* get_name() const {return name;}
21      int get_id() const {return id;}
22
23    private:
24      char* name;
25      int id;
26  };
27
28  std::ostream& operator<<(std::ostream& os, const person& p){
29      return os << p.get_name() << " [" << p.get_id() << "]";
30  }
31
32  bool operator<(const person& a, const person& b) {
33      return strcmp(a.get_name(), b.get_name()) < 0;
34  }
35
36  std::vector<person> create_persons() {
37      std::vector<person> res;
38
39      auto id = 0;
40      for(const auto n : {"Kalle", "Pelle", "Lisa", "Kim"}){
41         res.emplace_back(n, ++id);
42      }
43      return res;
44  }
45
46  int main() {
47      auto ps = create_persons();
48
49      cout << "created " << ps.size() << " persons: \n";
50      for(const auto& p : ps){
51          cout << p << '\n';
52      }
53
54      cout << "sorted by name: " << '\n';
55
56      std::sort(begin(ps), end(ps));
57
58      for(const auto& p : ps){
59          cout << p << '\n';
60      }
61  }
```

The following debugger output shows the failure: (The problem continues on the next page.)

```
(lldb) target create "./sort-person"
Current executable set to '/exam-problems/src/sort-person' (arm64).
(lldb) run
Process 23217 launched: '/exam-problems/src/sort-person' (arm64)
created 4 persons:
Kalle [1]
Pelle [2]
Lisa [3]
Kim [4]
sorted by name:
=================================================================
==23217==ERROR: AddressSanitizer: heap-use-after-free on address 0x6020000001f0 at pc 0x0001005537f8 bp 0
    x00016fdfdef0 sp 0x00016fdfd688
READ of size 1 at 0x6020000001f0 thread T0
    #0 0x0001005537f4 in strcmp+0x4d0 (libclang_rt.asan_osx_dynamic.dylib:arm64+0x177f4)
    #1 0x000100000a44 in operator<(person const&, person const&) sort-person.cc:33
    #2 0x000100008ee0 in bool std::__1::__less<void, void>::operator()[abi:nqe220106]<person, person>(
        person const&, person const&) const comp.h:42
    #3 0x000100009474 in void std::__1::__sort4[abi:nqe220106]<std::__1::_ClassicAlgPolicy, std::__1::
        __less<void, void>&, person*, 0>(person*, person*, person*, person*, std::__1::__less<void, void
        >&) sort.h:168
    #4 0x0001000089dc in void std::__1::__introsort<std::__1::_ClassicAlgPolicy, std::__1::__less<void,
        void>&, person*, false>(person*, person*, std::__1::__less<void, void>&, std::__1::
        iterator_traits<person*>::difference_type, bool) sort.h:743
    #5 0x0001000086a0 in void std::__1::__sort_dispatch[abi:nqe220106]<std::__1::_ClassicAlgPolicy, person
        *, std::__1::__less<void, void>>(person*, person*, std::__1::__less<void, void>&) sort.h:872
    #6 0x0001000085c8 in void std::__1::__sort_impl[abi:nqe220106]<std::__1::_ClassicAlgPolicy, std::__1::
        __wrap_iter<person*>, std::__1::__less<void, void>>(std::__1::__wrap_iter<person*>, std::__1::
        __wrap_iter<person*>, std::__1::__less<void, void>&) sort.h:937
    #7 0x0001000084ac in void std::__1::sort[abi:nqe220106]<std::__1::__wrap_iter<person*>, std::__1::
        __less<void, void>>(std::__1::__wrap_iter<person*>, std::__1::__wrap_iter<person*>, std::__1::
        __less<void, void>) sort.h:945
    #8 0x000100001d8c in void std::__1::sort[abi:nqe220106]<std::__1::__wrap_iter<person*>>(std::__1::
        __wrap_iter<person*>, std::__1::__wrap_iter<person*>) sort.h:951
    #9 0x0001000016b4 in main sort-person.cc:56
    #10 0x00018bcabdfc in start+0x1b4c (dyld:arm64e+0x1fdfc)

0x6020000001f0 is located 0 bytes inside of 6-byte region [0x6020000001f0,0x6020000001f6)
freed by thread T0 here:
    #0 0x0001005a15b4 in _ZdaPv+0x6c (libclang_rt.asan_osx_dynamic.dylib:arm64+0x655b4)
    #1 0x000100004144 in person::~person() sort-person.cc:14
    #2 0x0001000040cc in person::~person() sort-person.cc:14
    #3 0x00010000be44 in std::__1::enable_if<is_move_constructible<person>::value && is_move_assignable<
        person>::value, void>::type std::__1::swap[abi:nqe220106]<person>(person&, person&) swap.h:46
    #4 0x00010000bcf0 in void std::__1::iter_swap[abi:nqe220106]<person*, person*>(person*, person*)
        iter_swap.h:26
    #5 0x000100008f80 in void std::__1::_IterOps<std::__1::_ClassicAlgPolicy>::iter_swap[abi:nqe220106]<
        person*&, person*&>(person*&, person*&) iterator_operations.h:138
    #6 0x000100009148 in bool std::__1::__sort3[abi:nqe220106]<std::__1::_ClassicAlgPolicy, std::__1::
        __less<void, void>&, person*, 0>(person*, person*, person*, std::__1::__less<void, void>&) sort.h
        :120
    #7 0x000100009460 in void std::__1::__sort4[abi:nqe220106]<std::__1::_ClassicAlgPolicy, std::__1::
        __less<void, void>&, person*, 0>(person*, person*, person*, person*, std::__1::__less<void, void
        >&) sort.h:167
    #8 0x0001000089dc in void std::__1::__introsort<std::__1::_ClassicAlgPolicy, std::__1::__less<void,
        void>&, person*, false>(person*, person*, std::__1::__less<void, void>&, std::__1::
        iterator_traits<person*>::difference_type, bool) sort.h:743
    #9 0x0001000086a0 in void std::__1::__sort_dispatch[abi:nqe220106]<std::__1::_ClassicAlgPolicy, person
        *, std::__1::__less<void, void>>(person*, person*, std::__1::__less<void, void>&) sort.h:872
    #10 0x0001000085c8 in void std::__1::__sort_impl[abi:nqe220106]<std::__1::_ClassicAlgPolicy, std::__1
        ::__wrap_iter<person*>, std::__1::__less<void, void>>(std::__1::__wrap_iter<person*>, std::__1::
        __wrap_iter<person*>, std::__1::__less<void, void>&) sort.h:937
    #11 0x0001000084ac in void std::__1::sort[abi:nqe220106]<std::__1::__wrap_iter<person*>, std::__1::
        __less<void, void>>(std::__1::__wrap_iter<person*>, std::__1::__wrap_iter<person*>, std::__1::
        __less<void, void>) sort.h:945
    #12 0x000100001d8c in void std::__1::sort[abi:nqe220106]<std::__1::__wrap_iter<person*>>(std::__1::
        __wrap_iter<person*>, std::__1::__wrap_iter<person*>) sort.h:951
    #13 0x0001000016b4 in main sort-person.cc:56
    #14 0x00018bcabdfc in start+0x1b4c (dyld:arm64e+0x1fdfc)

previously allocated by thread T0 here:
    #0 0x0001005a11f8 in _Znam+0x6c (libclang_rt.asan_osx_dynamic.dylib:arm64+0x651f8)
    #1 0x000100007470 in person::person(person const&) sort-person.cc:16
    #2 0x0001000073f8 in person::person(person const&) sort-person.cc:16
    #3 0x00010000be18 in std::__1::enable_if<is_move_constructible<person>::value && is_move_assignable<
        person>::value, void>::type std::__1::swap[abi:nqe220106]<person>(person&, person&) swap.h:43
    #4 0x00010000bcf0 in void std::__1::iter_swap[abi:nqe220106]<person*, person*>(person*, person*)
        iter_swap.h:26
```

```
    #5 0x000100008f80 in void std::__1::_IterOps<std::__1::_ClassicAlgPolicy>::iter_swap[abi:nqe220106]<
        person*&, person*&>(person*&, person*&) iterator_operations.h:138
    #6 0x000100009148 in bool std::__1::__sort3[abi:nqe220106]<std::__1::_ClassicAlgPolicy, std::__1::
        __less<void, void>&, person*, 0>(person*, person*, person*, std::__1::__less<void, void>&) sort.h
        :120
    #7 0x000100009460 in void std::__1::__sort4[abi:nqe220106]<std::__1::_ClassicAlgPolicy, std::__1::
        __less<void, void>&, person*, 0>(person*, person*, person*, person*, std::__1::__less<void, void
        >&) sort.h:167
    #8 0x0001000089dc in void std::__1::__introsort<std::__1::_ClassicAlgPolicy, std::__1::__less<void,
        void>&, person*, false>(person*, person*, std::__1::__less<void, void>&, std::__1::
        iterator_traits<person*>::difference_type, bool) sort.h:743
    #9 0x0001000086a0 in void std::__1::__sort_dispatch[abi:nqe220106]<std::__1::_ClassicAlgPolicy, person
        *, std::__1::__less<void, void>>(person*, person*, std::__1::__less<void, void>&) sort.h:872
    #10 0x0001000085c8 in void std::__1::__sort_impl[abi:nqe220106]<std::__1::_ClassicAlgPolicy, std::__1
        ::__wrap_iter<person*>, std::__1::__less<void, void>>(std::__1::__wrap_iter<person*>, std::__1::
        __wrap_iter<person*>, std::__1::__less<void, void>&) sort.h:937
    #11 0x0001000084ac in void std::__1::sort[abi:nqe220106]<std::__1::__wrap_iter<person*>, std::__1::
        __less<void, void>>(std::__1::__wrap_iter<person*>, std::__1::__wrap_iter<person*>, std::__1::
        __less<void, void>) sort.h:945
    #12 0x000100001d8c in void std::__1::sort[abi:nqe220106]<std::__1::__wrap_iter<person*>>(std::__1::
        __wrap_iter<person*>, std::__1::__wrap_iter<person*>) sort.h:951
    #13 0x0001000016b4 in main sort-person.cc:56
    #14 0x00018bcabdfc in start+0x1b4c (dyld:arm64e+0x1fdfc)

SUMMARY: AddressSanitizer: heap-use-after-free sort-person.cc:33 in operator<(person const&, person const
    &)
==23217==ABORTING
Process 23217 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = Use of deallocated memory
frame #5: 0x0000000100000a48 sort-person`operator<(a=0x00006060000006b0, b=0x00006060000006a0)
at sort-person.cc:33:12
   32   bool operator<(const person& a, const person& b) {
-> 33       return strcmp(a.get_name(), b.get_name()) < 0;
   34   }
```

(Memory map omitted)

Moving up to `main()` and looking at the state of the program when it has crashed indicates corruption in the `person` vector:

```
(lldb) bt
* thread #1, queue = 'com.apple.main-thread', stop reason = Use of deallocated memory
    frame #4: 0x0000000100553820 libclang_rt.asan_osx_dynamic.dylib`wrap_strcmp + 1276
    * frame #5: 0x0000000100000a48 sort-person`operator<(a=0x00006060000006b0, b=0x00006060000006a0)
    at sort-person.cc:33:12
    frame #11: 0x00000001000084b0 sort-person`void std::__1::sort[abi:nqe220106]<std::__1::__wrap_iter<
        person*>, std::__1::__less<void, void>>(__first=__wrap_iter<person *> @ 0x000000016fdfe458,
        __last=__wrap_iter<person *> @ 0x000000016fdfe450, __comp=__less<void, void> @ 0x000000016fdfe3a0
        ) at sort.h:945:3
    frame #12: 0x0000000100001d90 sort-person`void std::__1::sort[abi:nqe220106]<std::__1::__wrap_iter<
        person*>>(__first=__wrap_iter<person *> @ 0x000000016fdfe4e8, __last=__wrap_iter<person *> @ 0
        x000000016fdfe4e0) at sort.h:951:3
    frame #13: 0x00000001000016b8 sort-person`main at sort-person.cc:56:5
    frame #14: 0x000000018bcabe00 dyld`start + 6992
(lldb) frame select 13
frame #13: 0x00000001000016b8 sort-person`main at sort-person.cc:56:5
   55
-> 56       std::sort(begin(ps), end(ps));
   57
(lldb) p ps
(std::vector<person>) size=4 {
  [0] = (name = "Kalle", id = 1)
  [1] = (name = "Lisa", id = 3)
  [2] = (name = "\x9a", id = 2)
  [3] = (name = "Kim", id = 4)
}
```

a) Explain why the program crashes, and how to fix it. For full marks you should describe the root cause (i.e., what in the code that causes the problem) and how to address that. You must also explain why the fundamental problem leads to the observed behaviour.

b) Implement the fix you described in a). Answer with the necessary changes or additions.

c) In `operator<` the function `strcmp` is used to compare the strings. Would the result of sorting be different if the expression was instead

```
return a.get_name() < b.get_name();
```

in the implementation of `operator<` ? If so, how would they be sorted?

2. A programmer is experimenting with the performance of the standard algorithms for searching and has written the following functions that use `std::find` which does linear search, and `std::lower_bound` which does binary search.

```
std::vector<int>::iterator do_find(std::vector<int> vec, int x)
{
    auto v = std::find(begin(vec), end(vec), x);
    if(v != end(vec) && *v != x){
        std::cout << "WRONG!\n";
    }
    return v;
}

std::vector<int>::iterator do_binary(std::vector<int> vec, int x)
{
    auto v = std::lower_bound(begin(vec), end(vec), x);
    if(v != end(vec) && *v < x){
        std::cout << "WRONG!\n";
    }
    return v;
}
```

The programmer has written a benchmarking program that searches for a randomly generated number in a `std::vector<int>`, with 100000 elements, that is sorted in ascending order. Benchmarking gives the following times:

```
  ----------------------------------------------------------------------
Benchmark                       Time             CPU   Iterations
----------------------------------------------------------------------
ns_bench/LinearSearch     451404 ns       451399 ns         1550
ns_bench/BinarySearch     418733 ns       418732 ns         1674
```

where `LinearSearch` is a benchmark that calls `do_find` and `BinarySearch` calls `do_binary`.
The programmer expected binary search to be significantly faster, but measurements show almost no difference in time between the two.
Explain why `do_binary()` is no faster than `do_find()`. For full credit you must both identify what – in the code – makes the two benchmarks take roughly the same amount of time, and explain why.

3. We want functionality that lets us iterate over all elements in an iterator range `[first,last)` that are equal to a value `val`. This is to be implemented by a function template `find_all` and a helper class template `result_iter`:

```
template <typename Iter, typename T>
result_iter<Iter,T>
find_all(Iter first, Iter last, const T& val);
```

Example use: the program

```
#include <vector>
#include <iostream>
void example()
{
    std::vector<int> v{3,1,2,4,3,5,4,6,5,7,3};

    auto it = find_all(begin(v), end(v), 3);

    while(it != end(v)){
        *it = 42;
        ++it;
    }

    for(auto x : v) std::cout << x << " ";
    std::cout << "\n";
}
```

should output `42 1 2 4 42 5 4 6 5 7 42`

Implement the function template `find_all` and the helper class template `result_iter`. You should only implement the functionality required by the above example[1].

As defined by the use, `result_iter` should have a public interface like:

```
template <typename Iter, typename T>
class result_iter {
public:
    // add suitable constructor
    result_iter& operator++();
    T& operator*();
    bool operator!=(Iter) const;
private:
    // add members as needed
};
```

`operator++` should return an iterator to the next element in the range `[first,last)` that is equal to `val`, or – when no such element is found – an iterator that is equal to `last`.

`operator*` should return a reference to an element in the original sequence.

Note that you only need an `operator!=` that compares a `result_iter` to an instance of its underlying interator type, as it is only used in this way: `while (it != end(v))`.

Answer with implementations of the function template `find_all` and the class template `result_iter`.

Hint: Most of the work is done in `result_iter`.

[1] Note that — to simplify the problem — the `result_iter` class does not need to have all the functionality of a standard library iterator. It does not need to have an iterator tag and the type members required by the standard algorithms.

4. In C++ a user-defined type can be written to behave just like a built-in type. Your task is to write a class `counter` that behaves just like an int, but also works as an accumulating function. The latter means that with a `counter` variable intialised to 100:

   ```
   counter c{100};
   ```

   calling it as a function

   ```
   c(10);
   ```

   increments its value by the argument, so in this case, `c` has the value 110 after the call. Calling it without an argument increments the counter by one.

   Implement the class `counter`*, so that the following example works as intended. Only implement what is needed by the example.*

   ```
   void example()
   {
       counter c;
       cout << "1: c = " << c << '\n';

       c = 17;
       cout << "2: c = " << c << '\n';

       int x = c;
       cout << "3: x = " << x << '\n';

       c();
       cout << "4: c = " << c << '\n';

       c(24);
       cout << "5: c = " << c << '\n';

       c = c + 100;
       cout << "6: c = " << c << '\n';

       if(c > 100){
           cout << "correct: " << c << " > 100\n";
       }
   }
   ```

   The expected output is

   ```
   1: c = 0
   2: c = 17
   3: x = 17
   4: c = 18
   5: c = 42
   6: c = 142
   correct: 142 > 100
   ```

   Answer with a complete definition of the class `counter`*.*

5. When trying to compile the following program

```
 1  #include <iostream>
 2  using std::cout;
 3
 4  struct A {
 5      A(int x) {val = x;}
 6      void print() {cout << "A("<<val<<")";}
 7      int val;
 8  };
 9
10
11  struct B {
12      B(int x) {a=A(x);}
13      void print() {cout << "B("; a.print(); cout << ")";}
14      A a;
15  };
16
17  int main()
18  {
19      B b(10);
20      b.print();
21      cout << '\n';
22  }
```

the compiler gives the error

```
testprogram.cpp|12 col 14 error| no matching function for call to 'A::A()'
|| B(int x) {a=A(x);}
||           ^
testprogram.cpp|5 col 5 info| candidate: A::A(int)
|| A(int x) {val = x;}
|| ^
testprogram.cpp|5 col 5 info| candidate expects 1 argument, 0 provided
testprogram.cpp|4 col 8 info| candidate: constexpr A::A(const A&)
|| struct A {
||        ^
testprogram.cpp|4 col 8 info| candidate expects 1 argument, 0 provided
testprogram.cpp|4 col 8 info| candidate: constexpr A::A(A&&)
testprogram.cpp|4 col 8 info| candidate expects 1 argument, 0 provided
```

a) Explain why the compiler gives this error

b) Show how the program can be fixed, by changing *the class* `A`.

c) Show how the program can be fixed, by changing *the class* `B`.

The program is expected to print

```
B(A(10))
```

Appendix

`std::find`:

```
   template< class InputIt, class T >
   InputIt
   find( InputIt first, InputIt last,
         OutputIt d_first, const T& value );
```

`std::find` searches for an element equal to value (using `operator==`).
`std::find` returns the first iterator it in the range `[first, last)` for which `*it == value` or `last` if there is no such iterator.

`std::find_if`:

```
   template< class InputIt, class UnaryPred >
   InputIt
   find_if( InputIt first, InputIt last, UnaryPred pred );
```

`std::find_if` searches for an element for which predicate `p` returns `true`.
`std::find_if` returns the first iterator it in the range `[first, last)` for which `p(*it)` returns true or `last` if there is no such iterator.
`std::lower_bound`:

```
   template< class ForwardIt, class T >
   ForwardIt lower_bound( ForwardIt first, ForwardIt last, const T& value );
```

`lower_bound` returns an iterator pointing to the first `element` in the range [first, last) such that `element < value` is false, (i.e. that is greater than or equal to value), or `last` if no such element is found. The range [first, last) must be partitioned with respect to the expression `element < value`, i.e., all elements for which the expression is true must precede all elements for which the expression is false. A fully-sorted range meets this criterion.

`std::swap`:

```
     template< class T >
     void swap( T& a, T& b );
```

Swaps the values `a` and `b`.
T must meet the requirements of `MoveConstructible` and `MoveAssignable`, where `MoveConstructible` specifies that an instance of the type can be constructed from an rvalue argument and `MoveAssignable` specifies that an instance of the type can be assigned from an rvalue argument.
`std::iter_swap`:

```
   template< class ForwardIt1, class ForwardIt2 >
   void iter_swap( ForwardIt1 a, ForwardIt2 b );
```

Swaps the values of the elements the given iterators are pointing to.

C library string functions, in `<cstring>`:

```
   char *strcpy(char *dst, const char * src);
```

`strcpy` copies the string pointed to by `src`, into a string at the buffer pointed to by `dst`. The programmer is responsible for allocating a destination buffer large enough, that is, `strlen(src) + 1`. The function returns `dst`.

```
   int strcmp(const char *s1, const char *s2);
```

The `strcmp`() function compares the two strings `s1` and `s2`. It returns an integer less than, equal to, or greater than zero if `s1` is found, respectively, to be less than, to match, or be greater than `s2`.

```
   size_t strlen(const char *s);
```

The `strlen()` function calculates the length of the string pointed to by `s`, excluding the terminating null byte (`'\0'`).