Iterators and Lists

In C++, the data structure that implements an array list is called std::vector. When it comes to linked lists, C++ provides two options: std::list and std::forward_list.

The C++ Standard Template Library (STL) makes heavy use of iterators. Iterators are typically used in pairs: a begin iterator, and an end iterator. An operation is then conducted on all elements, starting at the begin iterator, and ending at the end iterator. The end iterator is typically placed one past the final element in a collection.

<algorithm> header

The <algorithm> header in the STL contains most of the function that we will use in this lab.

C++20

In C++20, the concept of ranges was introduced. A range is simply a pair of begin and end iterators to improve the usability of iterators. Because C++20 compiler support is incomplete at the time of writing, we will not be dealing with ranges in this lab. Be sure to check out the documentation though, since compiler support may be complete by the time you are taking this lab.

Code

1. Write two functions that fill respectively a std::vector<std::string> and a std::list<string> with random names. Make sure that the user of your function can specify a seed for the random number generator and the number of names!
2. Write a couple of unit tests that check that the random names that are generated are always generated the same when using the same seed. Also check that the correct number of names is generated, and that your generation procedure works.
3. Write the following functions, using functionality present in the <algorithm> header. Each of these functions should take a pair of iterators as an argument. The best way to implement this, is to make each function a templated function on the iterator types you take as arguments. std::copy is a function in the standard library that's been built this way.
   - A function that prints all the given names to the console.
   - A function that returns all names that start with a given prefix. The prefix is given as an argument to the function and contains at least 1 character. The function should also take an output iterator as argument, similar to std::copy.
   - A function that prints a set of names to the console in reverse order. For this function, you can temporarily store the names in another data structure.

Report

1. What are the differences between array-backed lists (std::vector) and linked lists (std::forward_list and std::list)? When would you use an array-backed list and when would you use a linked list?
2. Why do we need to be able to seed our random number generator?
3. The documentation for std::vector::push_back mentions that under certain conditions, all iterators and references are invalidated. What does this mean?
4. We notice that almost every algorithm in the <algorithm> header takes a begin and end iterator as an argument. Why do you think the writers of the C++ standard chose to do this?

Iterators en Lists

In C++ heet de data structuur die een array list implementeert std::vector. Als het op linked lists aankomt biedt C++ twee opties aan: std::list en std::forward_list.

De C++ Standard Template Library (STL) maakt erg veel gebruik van iterators. Vaak worden iterators gebruikt in paren: een begin iterator, en een end iterator. Een operatie wordt dan uitgevoerd op alle elementen, beginnend bij de begin iterator, en eindigend bij de eind iterator. De eind iterator staat meestal op de positie achter het laatste element, zodat de combinatie van begin en eind ervoor zorgt dat een operatie op alle elementen wordt uitgevoerd.

<algorithm> header

De <algorithm> header in de STL bevat de meeste functies die we zullen gebruiken in dit practicum.

C++20

In C++20 werd het concept van ranges toegevoegd. Een range is gewoon een paar van begin en eind iterators, en maakt het gebruik van iterators iets eenvoudiger. Omdat C++20 compiler support nog niet volledig is op het moment van schrijven, zullen we voorlopig nog geen gebruik maken van ranges. Bekijk wel zeker de documentatie, omdat de kans reëel is dat op het moment dat dit practicum gegeven wordt, compiler support wel al deels aanwezig is.

Code

1. Schrijf twee functies die respectievelijk een std::vector<std::string> en een std::list<std::string> vullen met willekeurige namen. Zorg ervoor dat de gebruiker van je functie een seed kan meegeven voor de random number generator en het aantal namen!
2. Schrijf enkele unit tests die controleren dat je altijd dezelfde namen genereert wanneer je dezelfde seed meegeeft. Controleer ook dat je het juiste aantal namen genereerd, en dat de procedure om ze te genereren correct werkt.
3. Schrijf volgende set functies, gebruik makend van de algoritmen in de <algorithm> header. Elk van deze functie moet een paar iterators nemen als argument. De beste manier om dit te implementeren is door de functies te templaten op de iterator types die je als input neemt. std::copy is een functie in de standard library die op deze manier gebouwd is.
    - Schrijf een functie die alle namen in de datastructuur wegschrijft naar de console
    - Schrijf een functie die alle namen die beginnen met een gegeven prefix returnt. De prefix wordt meegegeven als argument aan de functie, kan 1 of meer karakters lang zijn. De functie moet ook een output iterator nemen als argument, net als std::copy.
    - Schrijf een functie die een reeks namen in omgekeerde volgorde afdrukt. Voor deze functie mag je tijdelijke de namen in een andere data structuur opslaan.

Verslag

1. Wat zijn de verschillen tussen array-backed lists (std::vector) en linked lists (std::forward_list en std::list)? Wanneer zou je een array-backed list en wanneer een linked list gebruiken?
2. Waarom moeten we onze random number generators kunnen seeden?
3. De documentatie voor std::vector::push_back vermeld dat onder bepaalde omstandigheden, alle iterators en references “invalidated” worden. Wat wil dat zeggen?
4. We merken op dat bijna alle functies in de <algorithm> header een begin en eind iterator als argument nemen. Waarom denk je dat de schrijvers van de C++ standard hiervoor gekozen hebben? Waarom kunnen we niet gewoon rechtstreeks een container zoals std::vector of std::map als argument nemen?