

(This handout might be used by Dr. Vigmond in his guest lecture.)

Electronic copies of handouts for L02 can be found at

<http://www.enel.ucalgary.ca/People/Norman/engg233-L02-fall2004/>

Programming problem #1: Let's write a program that will accept a list of positive numbers from a user, then print them out in reverse order. The program won't be very practically useful but it will demonstrate an important technique for situations where we don't know in advance how big a vector needs to be.

Here are two example dialogues with the program, with a ***bold slanted font*** used for user input:

Please enter a list of positive numbers, separated by spaces.

Use a negative number or zero to mark the end of the list.

1.3 8.8 47.615 2.2 -99.9

In reverse order the numbers are:

2.2 47.615 8.8 1.3

Please enter a list of positive numbers, separated by spaces.

Use a negative number or zero to mark the end of the list.

11.2 22.4 0

In reverse order the numbers are:

22.4 11.2

Since we don't know in advance how many numbers the user will enter, the plan will be to start with an empty vector, and grow the vector each time we need to copy an new value from the user. A program that does this is **vector-input-1.cc**:

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    vector<double> v;           // Make an empty vector.
    cout << "Please enter a list of positive numbers, "
         << "separated by spaces." << endl;
    cout << "Use a negative number or zero to mark the end of the list." << endl;

    // Put user's positive numbers into v one at a time.
    // WARNING: This relies on the user following instructions properly!
    double user_input;
    while (true) {
        cin >> user_input;
        if (user_input <= 0.0)
            break;
        v.resize(v.size() + 1); // Make room for one more element.
        v.at(v.size() - 1) = user_input; // New element = user_input.
    }

    cout << "In reverse order the numbers are:" << endl;
    for (int k = v.size() - 1; k >= 0; k--)
        cout << v.at(k) << " ";
    cout << endl;
    return 0;
}
```

(If you have finished Lab 6 Exercise 1, you should realize that there is a danger of an infinite loop if the user makes a mistake and enters letters or punctuation instead of numbers and spaces. To be safe, a check of `cin.fail()` should be put somewhere inside the input loop. That check has been left out to keep the code short and simple.)

In looking at **vector-input-1.cc**, you might notice that the lines

```
v.resize(v.size() + 1); // Make room for one more element.
v.at(v.size() - 1) = user_input; // New element = user_input.
```

do the required job, but look somewhat awkward. Making a vector one element bigger and copying a value into the new element is such a common operation that vectors provide an operation called `push_back` to do just that. The `while` loop in **vector-input-1.cc** could be replaced with this:

```

// Put user's positive numbers into v one at a time.
// WARNING: This relies on the user following instructions properly!
double user_input;
while (true) {
    cin >> user_input;
    if (user_input <= 0.0)
        break;
    v.push_back(user_input);
}

```

Selection sort. Here is a function that will sort a vector of ints from smallest to largest:

```

void selection_sort(vector<int>& v)
{
    int vsize = v.size();

    // One pass through the outer loop body swaps v[i] with the smallest
    // element among v[i], ... , v[vsize-1].
    for (int i = 0; i < vsize - 1; i++) {
        int i_of_min = i;

        // Completion of the inner loop sets i_of_min equal to the index
        // of the smallest element among v[i], ... , v[vsize-1].
        for (int j = i + 1; j < vsize; j++)
            if (v.at(j) < v.at(i_of_min))
                i_of_min = j;

        int temp = v.at(i);
        v.at(i) = v.at(i_of_min);
        v.at(i_of_min) = temp;
    }
}

```

Self-test: Would you be able to modify **vector-input-1.cc** on page 1 so that instead of reversing the order of the input, the program sorts it from smallest to largest before displaying it? (See the Web address at the top of page 1 to look up a solution.)

A little note about index range checking with `at`. In the lecture Oct. 19, Dr. Norman said that in the ENGG 233 lab, a program that tries to use `v.at(i)`, where `v` is a vector and `i` is an out-of-range index, will stop with a one-word error message: `Aborted`.

That's true, but it's worth mentioning two other points. First, a program might stop with `Aborted` as an error message for some other reason, depending on what library features the program is using. However, in the lab exercises in this course, it's very likely that `Aborted` means that a vector index was out-of-range.

Second, using a technique called *exception handling*, which involves the reserved words `try` and `catch`, it's possible to let a program continue after an out-of-range index is detected by `at`. C++ exception handling is too complex to cover in an introductory course such as ENGG 233.