

7. Sequence Data I

11. Sequences

Sequences

```
#include <iostream>
int main() {
    double n1, n2, n3, n4, n5;
    std::cout << "Please enter five numbers: ";
    std::cin >> n1 >> n2 >> n3 >> n4 >> n5;
    std::cout << "The average of " << n1 << ", " << n2 << ", "
        << n3 << ", " << n4 << ", " << n5 << " is "
        << (n1 + n2 + n3 + n4 + n5)/5 << '\n';
}

-----
#include <iostream>
int main() {
    double sum = 0.0, num;    const int NUMBER_OF_ENTRIES = 5;
    std::cout << "Please enter " << NUMBER_OF_ENTRIES << " numbers: ";
    for (int i = 0; i < NUMBER_OF_ENTRIES; i++) {
        std::cin >> num;
        sum += num;
    }
    std::cout << "The average of " << NUMBER_OF_ENTRIES << " values is "
        << sum/NUMBER_OF_ENTRIES << '\n';
}
```

Vectors (1)

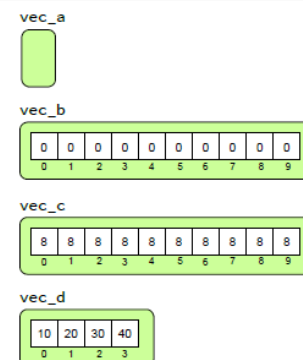
- Sequences: vectors and arrays
 - `std::vector`, `std::array`
- Nonempty sequence
 - Every nonempty sequence has a unique first element.
 - Every nonempty sequence has a unique last element.
 - Every element in a nonempty sequence except for the first element has a unique predecessor element.
 - Every element in a nonempty sequence except for the last element has a unique successor element.
- Vector is a template class in STL (Standard Template Library) of C++ programming language. C++ vectors are sequence containers that store elements.

Vectors (2)

```
#include <vector>

using std::vector;

std::vector<int> vec_a;
std::vector<int> vec_b(10); // All elements are zero by default.
std::vector<int> vec_c(10, 8);
std::vector<int> vec_d{10, 20, 30, 40};
```



Vectors (3)

```
std::vector<int> list(3);
list[0] = 5;
list[1] = -3;
list[2] = 12;

std::cout << list[1] << '\n'; // list[1] : int variable

std::vector<int> list;
std::vector<double> collection{ 1.0, 3.5, 0.5, 7.2 };
std::vector<char> letters{ 'a', 'b', 'c' };

std::vector<double> nums(10);
int i = 3;
nums[1] = 2.4;
nums[i] = 2.1;
nums[i*2-1] = 2.2; // nums[a[i]], a[max(x,y)]
std::cout << nums << '\n';
nums[10] = 5.1; // memory access error, nums[-1]
nums[1.5] = 2.2; // 1.5 --> 1
```

Vectors (4)

```
#include <iostream>
#include <vector>
int main() {
    double sum = 0.0;
    const int NUMBER_OF_ENTRIES = 5;
    std::vector<double> numbers(NUMBER_OF_ENTRIES);
    std::cout << "Please enter " << NUMBER_OF_ENTRIES << " numbers: ";
    for (int i = 0; i < NUMBER_OF_ENTRIES; i++) {
        std::cin >> numbers[i];
        sum += numbers[i];
    }
    std::cout << "The average of ";
    for (int i = 0; i < NUMBER_OF_ENTRIES; i++)
        std::cout << numbers[i] << ", ";
    std::cout << "The average is " << sum/NUMBER_OF_ENTRIES << '\n';
}
```

Vector Methods (1)

- `push_back`
 - —inserts a new element onto the back of a vector
- `pop_back`
 - —removes the last element from a vector
- `operator[]`
 - —provides access to the value stored at a given index within the vector
- `at`
 - —provides bounds-checking access to the value stored at a given position within the vector
- `size`
 - —returns the number of values currently stored in the vector
- `empty`
 - —returns true if the vector contains no elements; returns false if the vector contains one or more elements
- `clear`
 - —makes the vector empty.

A method in object-oriented programming is a procedure associated with a class.

- <https://www.cplusplus.com/reference/vector/vector/>

Vector Methods (2)

```
std::vector<int> list;           // Declare list to be a vector
list.push_back(5);              // Add 5 to the end of list
list.push_back(-3);             // Add -3 to the end of the list
list.push_back(12);             // Add 12 to the end of list
list.pop_back();                // Removes 12 from the list
list.pop_back();                // Removes -3 from the list
std::cout << list.size() << std::endl;    // 1

// reference operator[](size_type position);
// const_reference operator[](size_type position) const;
std::vector<int> vec = {10, 20, 30};
vec.operator[](2) = 3;
std::cout << vec[2] << std::endl;    // 3
vec[2] = 4;
std::cout << vec[2] << std::endl;    // 4
vec.at(2) = 5;
std::cout << vec[2] << std::endl;    // 5
```

Vector Methods (3)

```
for(type elementVariable : vectorVariable)    // c++11
    statement;
```

```
#include <iostream>
#include <vector>
int main() {
    std::vector<double> vec(10);
    std::cout << "Please enter 10 numbers: ";

    for (double& elem : vec)
        std::cin >> elem;

    for (double elem : vec)
        std::cout << elem << '\n';
}
// for (unsigned i = 0; i < vec.size(); i++)
//     std::cout << vec[i] << '\n';
```

Vectors and Functions (1)

```
returnType functionName(std::vector<type> variableName)
```

```
#include <iostream>
#include <vector>
void print(std::vector<int> v) {
    for (int elem : v) std::cout << elem << " ";
    std::cout << '\n';
}
void square(std::vector<int>& v) {
    for (int& elem : v) elem *= elem;
}
int sum(std::vector<int> v) {
    int result = 0;
    for (int elem : v) result += elem;
    return result;
}
int main() {
    std::vector<int> list{ 2, 4, 6, 8, };
    print(list);          std::cout << sum(list) << '\n';
    square(list);         print(list);    std::cout << sum(list) << '\n';
}
```

Vectors and Functions (2)

```
std::vector<type> functionName(parameterList)
-----
...
std::vector<int> primes(int begin, int end) {
    std::vector<int> result;
    for (int i = begin; i <= end; i++)
        if (is_prime(i))    result.push_back(i);
    return result;
}
int main() {
    int low, high;
    std::cout << "Please enter lowest and highest values in "
        << "the range: ";
    std::cin >> low >> high;
    std::vector<int> prime_list = primes(low, high);
    print(prime_list);
}
```

Multidimensional Vectors (1)

```
std::vector<std::vector<type>> a(size2, std::vector<type>(size1));
-----
std::vector<std::vector<int>> a(2, std::vector<int>(3));
a[0][0] = 5;
a[0][1] = 19;
a[0][2] = 3;
a[1][0] = 22;
a[1][1] = -8;
a[1][2] = 10;

std::vector<std::vector<int>> a({ 5, 19, 3}, {22, -8, 10});
```

Multidimensional Vectors (2)

```
void print(const std::vector<std::vector<double>>& m) {
    for (unsigned row = 0; row < m.size(); row++) {
        for (unsigned col = 0; col < m[row].size(); col++)
            std::cout << std::setw(5) << m[row][col];
        std::cout << '\n';
    }
}

-----

void print(const std::vector<std::vector<double>>& m) {
    for (const std::vector<double> row : m) { // auto row
        for (double elem : row)             // auto elem
            std::cout << std::setw(5) << elem;
        std::cout << '\n';
    }
}
```