Лабораторная работа №2. Использование std::vector

Что будет сегодня в лабораторной:

- работа с std::vector;
- практика по передаче vector в функцию и возврату из неё;
- вспомним решето Эратосфена;
- исследование зависимости размера вектора в памяти от количества элементов в нём.

Язык C++ имеет свои, более удобные массивы, чем те, что использовались в языке C. Это контейнерный тип std::vector, который может использоваться в качестве замены обычным массивам. Пример использования:

```
#include <iostream>
#include <vector>

using namespace std;

int main()
{
    int n;
    cin>>n;
    vector<int> a(n);
    for (int i=0; i<n; i++)
    {
        a[i] = i;
    }
    cout<<a[4]+a[5];
    return 0;
}
```

Элементы физически хранятся непрерывным блоком в памяти. Хранилище вектора обрабатывается автоматически, расширяясь и сужаясь по мере необходимости. Векторы обычно занимают больше места, чем статические массивы, поскольку некоторое количество памяти выделяется про запас на обработку будущего роста. Таким образом, память для вектора

требуется выделять не при каждой вставке элемента, а только после исчерпания резервов. Общий объём выделенной памяти можно получить с помощью функции `capacity()`. Резервная память может быть возвращена системе через вызов `shrink_to_fit()`.

Перераспределения обычно являются дорогостоящими операциями в плане производительности. Функция `reserve()` может использоваться для предварительного выделения памяти и устранения перераспределений, если заранее известно количество элементов.

Сложность (эффективность) обычных операций над векторами следующая:

- Произвольный доступ — постоянная O(1)
- Вставка и удаление элементов в конце — амортизированная постоянная O(1)
- Вставка и удаление элементов — линейная по расстоянию до конца вектора O(n)

Замечание:

vector<bool> хранит вектор в сжатом ввиде. Это значит, что каждый элемент такого вектора занимает не 8 бит в памяти, а только 1.

Задача 1. Написать функцию поиска наибольшего элемента в массиве. Массив должен быть представлен как vector<int>.

Пример передачи vector в функцию:

```
#include <iostream>
#include <vector>

using namespace std;

//Заполнение массива
void fill(vector<int> &v)
{
    for (int i=0; i<v.size(); i++)
    {
        v[i] = i;
    }
}

//Вывод массива
void print(const vector<int> &v)
{
    for (int i=0; i<v.size(); i++)
    {
        cout<<v[i]<<endl;
    }
}

int main()
{
    int n;
    cin>>n;
    vector<int> a(n);
    fill(a);
    print(a);
    return 0;
}
```

Задача 2. Написать программу, которой на вход подаётся N чисел от 1 до 20. Программа должна подсчитывать количество введённых чисел для каждого значения. В конце работы программы нужно вывести в отдельных строках количество введённых единиц, двоек, троек и так далее, до 20.

Формат ввода	Формат вывода
10 1 1 2 3 3 3 3 3 5 5	1: 2 2: 1 3: 5 5: 2

Задача 3. Написать решето эратосфена в виде функции. Использовать vector<bool>. Количество искомых простых чисел N должно передаваться в функцию в качестве аргумента. Для избежания лишних выделений и освобождений памяти, а так же переносов всего массива, резервирование вектором памяти под N элементов должно осуществляться до того, как в вектор начнут добавляться элементы.

а) Функция eratosthene должна возвращать вектор, содержащий N простых чисел, начиная с числа 2.

б) Функция eratosthene должна принимать на вход ссылку на вектор с возможностью его дальнейшего изменения.

Результат работы функции вывести в текстовый файл, используя std::ofstream.

Пример добавления числа в конец вектора:

```
a.push_back(i);
```

Пример работы с std::ofstream:

```
#include <fstream>
using namespace std;

int main()
{
    ofstream fout("output.txt");
    fout<<"Text";
    return 0;
}
```

Задача 4. Известно, что vector может неограниченно (в пределах оперативной памяти) менять свои размеры при добавлении в него элементов с помощью метода push_back. Нужно исследовать рост выделенной под vector памяти с добавлением в конец него элементов. Для этого добавляйте элементы в вектор по одному и выявите зависимость выделенной памяти от количества элементов в векторе. Текущий размер вектора можно определить при помощи функции capacity(). Эта функция возвращает количество элементов, для которых выделена память в vector.

Для создания пустого вектора можно использовать конструктор по умолчанию:

```
vector<int> a;
```

Вопросы:

1. Какова сложность добавления элемента в середину вектора?
2. Какова сложность вставки в конец?
3. Как выполнить обращение к элементу вектора по его номеру?
4. Какова сложность операции обращения к произвольному элементу?
5. Сколько байт в памяти будет занимать vector<bool> на 1024 элемента?