***Класс vector***

Наиболее универсальным контейнерным классом является класс **vector** ─ динамический массив. В C++ в процессе компиляции размеры массива фиксируются. Использование класса vector позволяет менять размер массива динамически.

Примеры объявления вектора:

vector <int> iv; // создание вектора нулевой длины для целых

vector<char> cv(5); // создание пятиэлементного вектора для символов

Для класса vector определяются операторы сравнения:

== < > <= != >= и оператор индекса []

Наиболее используемые функции-члены класса vector

|  |  |
| --- | --- |
| **Функция-член** | **Назначение** |
| **size()** | возвращает текущий размер вектора |
| **begin()** | возвращает итератор начала вектора |
| **end()** | возвращает итератор конца вектора |
| **push\_back()** | помещает значение в конец вектора |
| **insert()** | помещает значение в середину вектора |
| **erase()** | удаление элементов из вектора |

**Пример**:// *Основные операции вектора*

#include <iostream>

#include <vector>

using namespace std;

int main()

{

setlocale(LC\_ALL, "RUS");

vector<int> v; // создание вектора нулевой длины

int i;

// вывод на экран размера исходного вектора v

cout << "Размер = " << v.size() << endl;

// помещение значений в конец вектора,

// по мере необходимости вектор будет расти

for (i = 0; i < 10; i++) v.push\_back(i);

// вывод на экран текущего размера вектора v

cout << "Новый размер = " << v.size() << endl;

// вывод на экран содержимого вектора v

cout << "Текущее содержимое:\n";

for (i = 0; i < v.size(); i++) cout << v[i] << " ";

cout << endl;

// помещение новых значений в конец вектора,

// и опять по мере необходимости вектор будет расти

for (i = 0; i < 10; i++) v.push\_back(i + 10);

// вывод на экран текущего размера вектора

cout << "Новый размер = " << v.size() << endl;

// вывод на экран содержимого вектора

cout << "Текущее содержимое:\n";

for (i = 0; i < v.size(); i++) cout << v[i] << " ";

cout << endl;

// изменение содержимого вектора

for (i = 0; i < v.size(); i++) v[i] = v[i] + v[i];

// вывод на экран содержимого вектора

cout << "Удвоенное содержимое:\n";

for (i = 0; i < v.size(); i++) cout << v[i] << " ";

cout << endl;

system("pause");

return 0;

}

**Пример**:// *Организация доступа к вектору с помощью итератора*

#include <iostream>

#include <vector>

using namespace std;

int main()

{

vector<int> v; // создание вектора нулевой длины

int i;

// помещение значений в вектор

for (i = 0; i < 10; i++) v.push\_back(i);

// доступ к содержимому вектора

// с использованием оператора индекса

for (i = 0; i < 10; i++) cout << v[i] << " ";

cout << endl;

// доступ к вектору через итератор

vector<int>::iterator p = v.begin();

while (p != v.end()) {

cout << \*p << " ";

p++;

}

system("pause");

return 0;

}

**Пример**:// Работа функций insert() и erase()

#include <iostream>

#include <vector>

#include<iterator>

using namespace std;

int main()

{

setlocale(LC\_ALL, "RUS");

vector<int> v(5, 1); // создание пятиэлементного вектора

// из единиц

int i;

// вывод на экран исходных размера и содержимого вектора

cout << "Размер = " << v.size() << endl;

cout << "Исходное содержимое:\n";

for (i = 0; i < v.size(); i++) cout << v[i] << " ";

cout << endl;

vector<int>::iterator p = v.begin();

p += 2; // указывает на третий элемент

// вставка в вектор на то место,

// куда указывает итератор p десяти новых элементов,

// каждый из которых равен 9

v.insert(p, 10, 9);

// вывод на экран размера

// и содержимого вектора после вставки

cout << "Размер после вставки = " << v.size() << endl;

cout << "Содержимое после вставки:\n";

for (i = 0; i < v.size(); i++) cout << v[i] << " ";

cout << endl;

// удаление вставленных элементов

p = v.begin();

p += 2; // указывает на третий элемент

v.erase(p, p + 10); // удаление следующих десяти элементов

// за элементом, на который указывает

// итератор p

// вывод на экран размера

// и содержимого вектора после удаления

cout << "Размер после удаления = " << v.size() << endl;

cout << "Содержимое после удаления:\n";

for (i = 0; i < v.size(); i++) cout << v[i] << " ";

cout << endl;

// вывод на экран размера

p = v.begin();

cout << "Вывод с использованием итератора вывода:\n";

copy(p+2, v.end(), ostream\_iterator<int>(cout, " "));

cout << endl;

system("pause");

return 0;

}

***Стандартный строковый класс string***

Для включения в C++ стандартного класса string (к дополнению к уже имеющемуся массиву символов в стиле языка Си) имеется три довода:

совместимость (строка является типом данных);

стандартность (можно использовать стандартные операторы C++);

безопасность (границы массивов контролируются).

Класс string является одним из определенных в C++ классов контейнеров, что позволяет поддерживать использование алгоритмов.

Чтобы получить доступ к классу string, в программу следует включить заголовок <string>.

Основные операторы, допустимые при работе с объектами string:

|  |  |
| --- | --- |
| Оператор | Значение |
| = | присваивание |
| + | конкатенация |
| += | присваивание с конкатенацией |
| == | Равенство (сравнение) |
| != | неравенство |
| < | меньше |
| <= | меньше или равно |
| > | больше |
| >= | больше или равно |
| [] | индекс |
| << | вывод |
| >> | ввод |

**Пример**: // использования строкового класса

// Использование функций insert(), erase() и replace()

#include <iostream>

#include <string>

using namespace std;

int main()

{

setlocale(LC\_ALL,"RUS");

string str1("Это проверка");

string str2("АБВГДЕЖ");

cout << "Исходные строки:\n";

cout << "str1: " << str1 << endl;

cout << "str2: " << str2 << "\n\n";

// работа функции insert()

cout << "Вставка строки str2 в строку str1:\n";

str1.insert(4, str2);

cout << str1 << "\n\n";

// работа функции erase()

cout << "Удаление семи символов из строки str1:\n";

str1.erase(4, 7);

cout << str1 << "\n\n";

// работа функции replace()

cout << "Замена восьми символов из str1 символами из str2:\n";

str1.replace(4, 8, str2);

cout << str1 << "\n\n";

system("pause");

return 0;

}

// Особенности работы оператора cin

#include <iostream>

#include <string>

using namespace std;

int main()

{

setlocale(LC\_ALL, "RUS");

string Name;

cout << "Введите Ваше полное имя: ";

cin>> Name;

cout << Name<<endl;

system("pause");

return 0;

}

// Использование функции getline()и SetConsoleCP(1251), SetConsoleOutputCP(1251)

#include <iostream>

#include <string>

#include <windows.h>

using namespace std;

int main()

{

//setlocale(LC\_ALL, "RUS");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string Name;

cout << "Введите Ваше полное имя (на английском языке): ";

getline(cin, Name);

cout << Name<<endl;

system("pause");

return 0;

}

***Задание***: Написать программу «Статистический анализ» с использованием STL.