Министерство науки и высшего образования РФ

Федеральное бюджетное государственное образовательное учреждение высшего профессионального образования

Пермский национальный исследовательский политехнический университет (ПНИПУ)

Факультет: Электротехнический (ЭТФ)

Направление: 09.03.01 «Информатика и вычислительная техника (Автоматизированные системы обработки информации и управления)»

Кафедра: «Информационные технологии и автоматизированных систем» (ИТАС)

Информатика

Лабораторная работа №6

АТД. Контейнеры

Студент: Балтаев Э. А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись)

Группа: АСУ-20-1бзу

Работу проверил:

доцент кафедры ИТАС

Полякова О. А.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Подпись:\_\_\_\_\_\_\_\_\_\_\_

Пермь 2021г.

Постановка задачи

1. Определить класс-контейнер

2. Реализовать конструкторы, деструкторы, операции ввода-вывода, операцию присваивания

3. Перегрузить операции, указанные в варианте

4. Реализовать класс-итератор. Реализовать с его помощью операции последовательного доступа

5. Написать тестирующую программу, иллюстрирующую выполнение операции

Задание варианта

Класс-контейнер МНОЖЕСТВО с элементами типа int

Реализовать операции:

[] – доступа по индексу

Int() – определение размера вектора;

\* - пересечение множеств

-- - переход к предыдущему элементу (с помощью класса-итератора)

Описание класса

#pragma once

#include <iostream>

using namespace std;

class Iterator

{

friend class Set;

private:

int\* elem;

public:

Iterator() { elem = 0; }

Iterator(const Iterator& it) { elem = it.elem; }

bool operator== (const Iterator& it) { return elem == it.elem; }

bool operator!= (const Iterator& it) { return elem != it.elem; }

void operator++ () { ++elem; }

void operator-- () { --elem; }

int& operator\*() const { return \*elem; }

};

class Set

{

private:

int size;

int\* data;

Iterator beg;

Iterator end;

public:

Set(int s);

Set(const Set& s);

~Set();

Set& operator= (const Set& v);

friend ostream& operator<< (ostream& out, const Set& v);

friend istream& operator>> (istream& in, Set& v);

int& operator[] (int index); // доступ по индексу

int operator() (); // определение размера

Set operator\* (const Set& s); // пересечение множеств

Iterator first() { return beg; }

Iterator last() { return end; }

private:

void reset(); // убрать одинаковые

};

Определение компонентных функций

#include "Set.h"

Set::Set(int s)

{

size = s;

data = new int[size];

for (int i = 0; i < size; i++)

data[i] = i + 1;

beg.elem = &data[0];

end.elem = &data[size];

}

Set::Set(const Set& s)

{

size = s.size;

data = new int[size];

for (int i = 0; i < size; i++)

data[i] = s.data[i];

beg.elem = &data[0];

end.elem = &data[size];

}

Set::~Set()

{

delete[] data;

data = 0;

}

Set& Set::operator=(const Set& v)

{

if (this == &v)

return \*this;

if (size != 0)

delete[] data;

size = v.size;

data = new int[size];

for (int i = 0; i < size; i++)

data[i] = v.data[i];

beg.elem = &data[0];

end.elem = &data[size];

return \*this;

}

void Set::reset()

{

int ok, k = 0;

int\* arr = new int[size];

// идем по элементами

for (int i = 0; i < size; i++) {

ok = 0;

// идем по доп. массиву

for (int j = 0; j < k; j++)

// если он там есть

if (data[i] == arr[j]) {

ok = 1; // отмечаем, что копировать не надо

break;

}

// если его там нет

if (ok == 0) {

arr[k] = data[i];

k++;

}

}

// копируем

if (size != 0)

delete[] data;

size = k;

data = new int[size];

for (int i = 0; i < size; i++)

data[i] = arr[i];

delete[] arr;

beg.elem = &data[0];

end.elem = &data[size];

}

int& Set::operator[](int index)

{

if (index < 0 || index >= size) {

cout << "Неверный индекс";

}

else

return data[index];

}

int Set::operator()()

{

return size;

}

Set Set::operator\*(const Set& s)

{

int count, k = 0;

if (size >= s.size)

count = size;

else

count = s.size;

int\* arr = new int[count];

// идем по элементами

for(int i=0; i<size; i++)

// идем по элементами другого множества

for(int j=0; j<s.size; j++)

// если они равны

if (data[i] == s.data[j]) {

arr[k] = data[i]; // копируем

k++;

break;

}

// копируем

Set temp(k);

for (int i = 0; i < k; i++)

temp.data[i] = arr[i];

delete[] arr;

return temp;

}

ostream& operator<<(ostream& out, const Set& v)

{

if (v.size == 0)

out << "Пусто";

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

out << v.data[i] << " ";

return out;

}

istream& operator>>(istream& in, Set& v)

{

for (int i = 0; i < v.size; i++) {

cout << "val? ";

in >> v.data[i];

}

v.reset();

return in;

}

Определение главного файла

#include <iostream>

#include "Set.h"

int main()

{

system("chcp 1251");

cout << "\nСоздание, ввод и вывод множеств\n";

Set s1(5);

cin >> s1;

cout << "s1: " << s1 << endl;

Set s2(s1);

cout << "s2: " << s2 << endl;

cin >> s1;

cout << "s1 - " << s1 << endl;

cout << "\nУнарные операции\n";

cout << "s1 size = " << s1() << endl;

cout << "s2[2] = " << s2[2] << endl;

cout << "\nБинарные операции\n";

Set s3 = s1 \* s2;

cout << "s1 \* s2 = " << s3 << endl;

cout << "\nИтератор\n";

Iterator it = s1.first();

cout << "s1 first = " << \*it << endl;

++it; ++it;

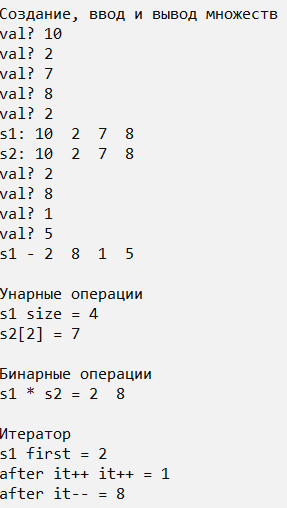
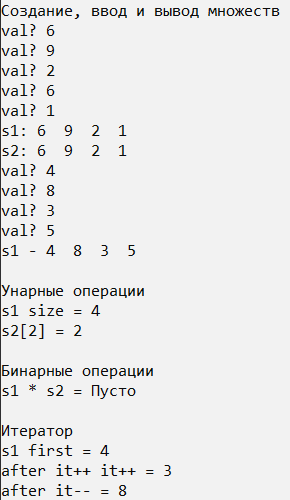
cout << "after it++ it++ = " << \*it << endl;

--it;

cout << "after it-- = " << \*it << endl;

}

Результаты работы программы

Ответы на контрольные вопросы

1) АТД – тип, определяемый по уникальным свойствам и своим операциям. Пример: класс, структура…

2) Стороннике функции, использующие в качестве параметра – абстрактный тип данных

3) Методы самого класса

4) Контейнер – набор однотипных данных, в частности массивы, множества, списки …

5) Доступа к элементами с изменением; Добавление/удаление; Поиск; Объединения; Специальные операции;

6) Последовательный (перебирать каждый элемент пока не найдем - очередь); Прямой (доступ по индексу - массив); Ассоциативный (с помощью ключа - словарь);

7) Итератор – объект, который обеспечивает последовательный доступ к элементам контейнера.

8) Как класс, представляющий набор операций.

9) Создается новый контейнер, в который объединяются данные двух контейнеров разными способами: сначала первого, потом второго; с сортировкой; объединения без повторения; пересечение множеств; как операция вычитания; выбрана часть контейнера;

10) Ассоциативный доступ

11) Стек

12) d

13) d

14) mas[ind]

15) Последовательный с помощью итератора, (\*iter)