ЛАБОРАТОРНАЯ РАБОТА 5

Наследование. Множества

Одно из основных применений наследования – использование ранее разработанных и уже проверенных классов для случаев, когда разрабатываемый класс несколько отличается от имеющегося, но есть желание использовать его в качестве базового. Дополнительные трудности на этапе освоения этой технологии возникают при использовании библиотечных классов из-за того, что библиотечные классы разработаны как шаблоны классов и их использование имеет свои особенности.

Класс множество (set) является одним из типов ассоциативных контейнеров. В ассоциативных контейнерах доступ к данным выполняется не по индексу элемента (как в массиве), а по его значению (ключу). Множества хранят элементы, значения ключей которых должны быть уникальными. Наиболее часто используемой операцией при работе с множествами является поиск элемента. Поэтому для эффективной работы класс set должен обеспечивать быстрый доступ к данным по ключу. Для реализации этого требования множества, как правило, реализуются на основе сбалансированного дерева или отсортированного массива.

Задание.

Разработать класс множество (**MySet**) на базе класса вектор **(MyVector)** для выполнения операций над множествами (+, -, *,+=,-=,*=,==) и функцию **main( )** для его тестирования.

Класс вектор должен быть динамическим массивом, размер которого может автоматически изменяться (увеличиваться или уменьшаться) в процессе выполнения программы. Добавление элементов производится в конец вектора.

Для ускорения выполнения операций над множествами вектор, используемый классом множество, должен быть отсортирован (сортировку достаточно делать только при добавлении элемента в множество). Для поиска элементов множества следует использовать метод половинного деления.

Методы ***add_element()*** и ***delete_element()*** производного класса ***MySet*** перегружают одноименные методы базового класса ***MyVector***, а остальные элементы класса ***MyVector*** наследуются классом ***MySet*** .

Описание программы:

Класс с именем **MySet** включает следующие элементы:

Члены - данные

Все данные наследуются из класса ***MyVector*** (элементы множества хранятся в векторе)**.**

Конструкторы и деструкторы

Так как своих данных в ***MySet*** нет, то можно не перегружать имеющиеся (по умолчанию) конструктор копирования, оператор присваивания и деструктор. При необходимости будут вызываться соответствующие элементы базового класса.

Методы доступа

IsElement, который даёт **true**, если строка-параметр есть в множестве, иначе даёт **false**. Для поиска элементов множества следует использовать метод половинного деления. Для его реализации разработать метод **q_find**, который имеет тип доступа **private**.

Методы изменения

AddElement - добавляет строку в множество, если её там ещё нет. Для ускорения поиска элементов создаваемое множество должно быть отсортировано по возрастанию значения ключа. Для сортировки при добавлении элементов использовать метод **sort()** базового класса.

DeleteElement для удаления строки из множества, если она там есть

Операторы

операторы присваивания -=, +=, *=, где - означает разность, + - объединение и * - пересечение множеств. (См. примеры ниже.)

Функции – не члены класса (друзья класса)

Перегруженная операция потокового вывода;

Операторы + (объединение), **-** (разность), ***** (пересечение) и **==** (сравнение: истина, если элементы двух множеств совпадают).

Примеры операций над множествами (приведены для целых чисел):

{1, 4, 5, 6} + {1, 2, 3, 4} => {1, 2, 3, 4, 5, 6}
{1, 4, 5, 6} * {1, 2, 3, 4} => {1, 4}
{1, 4, 5, 6} - {1, 2, 3, 4} => {5, 6}

Базовый класс **MyVector**, является динамическим ***массивом строк.*** Размер вектора ***maxsize*** должен меняться в процессе выполнения программы следующим образом:

если при добавлении элемента число элементов вектора ***size*** превысит размер вектора, ***maxsize*** увеличивается примерно в 1,5 раза (был 8, станет 12, если ***size***>=8)

если при удалении элемента число элементов вектора ***size*** станет меньше ***maxsize/2,*** ***maxsize*** уменьшается примерно в 1,5 раза, но должен быть не меньше значения по умолчанию (был 12, станет 8, если ***size*** <6). Новый элемент добавляется в конец вектора.

Члены - данные (protected) :

maxsize –размер вектора;
size – количество элементов в векторе;
pdata – указатель, содержащий адрес динамического массива элементов (строк).

Класс MyVector должен реализовывать следующие функции:

add_element – вставка элемента в конец вектора;
delete_element – удаление элемента из произвольного места;
find(el) – возвращает индекс элемента или –1, если элемент не найден;
resize – изменение размера вектора **maxsize** при его переполнении или освобождении места **(private);**

Конструкторы и деструктор

Конструктор с одним параметром (символьная строка) для создания множества размером 1, который имеет значения по умолчанию и поэтому может использоваться для создания пустого множества;
Конструктор копирования;
Деструктор.

Операторы

[ ] - для возврата элемента вектора (доступ по индексу);
= - оператор присваивания.

При выполнении лабораторной работы использовать файлы с описанием классов ***MyVector, MySet*** и тестирующей программой ***lab4***, приведенные в Приложении 1. Работа состоит из четырех частей: ЛР4.1, ЛР4.2, ЛР4.3, ЛР4.4. Общее время выполнения работы – 16 часов.

Содержание частей ЛР:

ЛР4.1 – Разработка и тестирование класса ***MyVector*** для хранения символьных строк.

ЛР4.2 – Разработка и тестирование класса ***MySet*** для выполнения операций над множествами символьных строк.

ЛР4.3 - Разработка и тестирование шаблонного класса ***MyVector***.

ЛР4.4 – Разработка и тестирование шаблонного класса ***MySet.***

Приложение 1.

```
//файл MyVector.h - описание класса MyVector
#ifndef MYVECTOR_H
#define MYVECTOR_H
#include <iostream>
using namespace std;

const int MAX_SIZE = 5;
class MyVector
{
public:
      MyVector(char *el = NULL, int maxsz = MAX_SIZE);

      MyVector(MyVector& v);
      ~MyVector();

      void add_element(char* el);
      bool delete_element(int i);
      char*  operator[](int i);

      void sort();

      int Size(){return size;}
      int Maxsize(){return maxsize;}
      int find(char* el);
      MyVector& operator=(MyVector& v);
      friend ostream& operator<<(ostream& out, MyVector& v);
protected:
      int maxsize;
      int size;
      char ** pdata;
private:
      void resize();
};
#endif

//файл MySet.h - описание класса MySet
#ifndef MYSET_H
#define MYSET_H
#include <iostream>
using namespace std;

class MySet:public MyVector
{
public:
      MySet(char* el = NULL):MyVector(el){};
```

```
	friend ostream& operator<<(ostream& out, MySet& s);
	friend MySet operator+(MySet& s1,MySet& s2);
	friend MySet operator-(MySet& s1,MySet& s2);
	friend MySet operator*(MySet& s1,MySet& s2);

	bool operator==(MySet& s);
	MySet& operator+=(MySet& s);
	MySet& operator-=(MySet& s);
	MySet& operator*=(MySet& s);
	void add_element(char* el);
	void delete_element(char* el);
	bool is_element(char* el);
};

#endif

//файл lab4 для тестирования классов MyVector и MySet
#include <iostream>
#include "MyVector.h"
#include "MySet.h"
#include "MyVector.cpp"
#include "MySet.cpp"
using namespace std;

int main ()
{
	setlocale(0,"russian");
	MyVector v("Hello!");
	v.add_element("Привет!");
	v.add_element("Привет!");
	v.add_element("Привет!");
	v.add_element("Привет!");
	v.add_element("Привет!");
	cout<<"Вектор v: "<<v<<endl;
	v.add_element("Привет!");
	v.add_element("Привет!");
	v.add_element("Привет!");
	cout<<"Вектор v: "<<v<<endl;
	MyVector v1=v;
	cout<<"Вектор v1: "<<v1<<endl;
	for(int i=0;i<MAX_SIZE;i++)
		v1.delete_element(0);
	cout<<"Вектор v1: "<<v1<<endl;
	MySet s("Yes"),s1,s2;
	s.add_element("Привет!");
	s.add_element("No");
	char* str="Hello!";
	s.add_element(str);
	cout<<"Множество s: "<<s<<endl;
	s1.add_element("Cat");
	s1.add_element("No");
	s1.add_element("Привет!");
	cout<<"Множество s1: "<<s1<<endl;
	s2=s1-s;
	cout<<"Множество s2=s1-s: "<<s2<<endl;
	cout<<"Множество s1: "<<s1<<endl;
	cout<<"Множество s: "<<s<<endl;
	s2=s-s1;
	cout<<"Множество s2=s-s1: "<<s2<<endl;
	cout<<"Множество s1: "<<s1<<endl;
	cout<<"Множество s: "<<s<<endl;
```

```
	s2=s1+s;
	cout<<"Множество s2=s1+s: "<<s2<<endl;
	cout<<"Множество s1: "<<s1<<endl;
	cout<<"Множество s: "<<s<<endl;
	s2=s1*s;
	cout<<"Множество s2=s1*s: "<<s2<<endl;
	cout<<"Множество s1: "<<s1<<endl;
	cout<<"Множество s: "<<s<<endl;
	MySet s3=s2;
	cout<<"Множество s3=s2: "<<s3<<endl;
	if(s3==s2)
		cout<<"Множество s3=s2\n";
	else
		cout<<"Множество s3!=s2\n";
	if(s3==s1)
		cout<<"Множество s3=s1\n";
	else
		cout<<"Множество s3!=s1\n";
	if(s1==s3)
		cout<<"Множество s1=s3\n";
	else
		cout<<"Множество s1!=s3\n";
	return 0;
}
```