Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Пермский национальный исследовательский политехнический университет» (ПНИПУ)

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

**Лабораторная работа «Классы №6»**

**по дисциплине**

**«Основы алгоритмизации и программирования»**

**(Семестр** 2)

Выполнил студент гр. ИВТ-21-1б

Гребнев Алексей Дмитриевич

Проверил:

Яруллин Денис Владимирович

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

(оценка) (подпись)

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

(дата)

г.Пермь -2022

**Постановка задачи**

1. Определить класс-контейнер.
2. Реализовать конструкторы, деструктор, 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 операций.

**Исходные данные**

Вариант 10:

Класс-контейнер МНОЖЕСТВО с элементами типа int.  
Реализовать операции:  
[] – доступа по индексу;  
() – определение размера множества;

- - разность множеств;  
-- - переход к предыдущему элементу(с помощью класса-итератора)

#include <iostream>

using namespace std;

class Iterator

{

friend class Variety;

private:

int\* elem;

public:

Iterator() //Конструктор без параметров

{

elem = 0;

}

int& operator\*() const

{

return \*elem;

}

Iterator(const Iterator& a)

{

elem = a.elem;

}

void operator++()

{

++elem;

}

void operator--()

{

cout << "Переход к предыдущему элементу: ";

--elem;

}

bool operator!=(const Iterator& a)

{

return elem != a.elem;

}

};

class Variety

{

private:

int size;

int\* data;

Iterator beg;

Iterator end;

public:

~Variety()

{

}

Variety(int size)

{

this->size = size;

data = new int[size];

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

{

data[i] = rand() % 96;

}

beg.elem = &data[0];

end.elem = &data[size];

}

int& operator[](int num)

{

if (num < size && num >= 0)

{

return data[num];

}

else

{

cout << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl << "Ошибка" << endl << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

}

Variety operator-(const Variety& a) //-a

{

Variety temp(size);

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

{

temp[i] = data[i] - a.data[i];

}

delete[] data;

return temp;

}

int& operator() ()

{

return size;

}

Iterator first()

{

return beg;

}

Iterator last()

{

return end;

}

};

int main()

{

setlocale(LC\_ALL, "RUS");

int index;

int size;

Iterator iter;

cout << "Задайте размер(size): ";

cin >> size;

Variety a(size);

cout << "class Variety a: ";

for (iter = a.first(); iter != a.last(); ++iter)

{

cout << \*iter << ", ";

}

cout << endl << "Определение размера множества a: " << a() << endl;

cout << endl << "Задайте индекс(index) для поиска: ";

cin >> index;

if (index >= 0 && index < size)

{

cout << "Доступ по заданному индексу: " << a[index] << endl;

}

else

{

cout << endl << "Доступ по заданному индексу невозможен!" << endl;

}

cout << endl;

--iter;

cout << \*iter << endl;

--iter;

cout << \*iter << endl << endl;

Variety b(size);

cout << "class Variety b: ";

for (iter = b.first(); iter != b.last(); ++iter)

{

cout << \*iter << ", ";

}

cout << endl << "Определение размера множества b: " << b() << endl;

Variety c = a - b;

cout << endl << "class Variety c => " << "\*Разность 'Variety a' и 'Variety b'\*: ";

for (iter = c.first(); iter != c.last(); ++iter)

{

cout << \*iter << ", ";

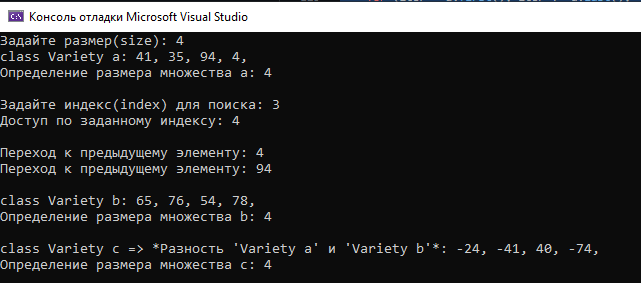
}

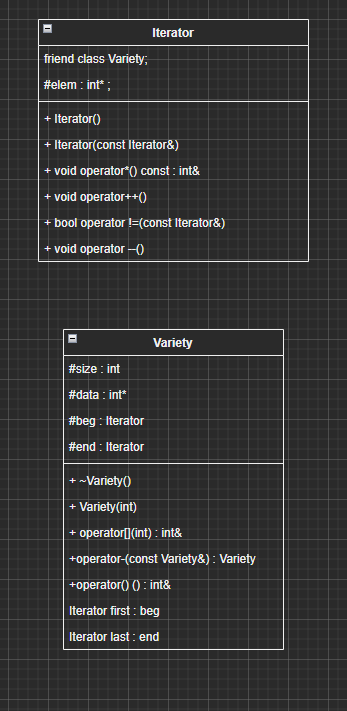
cout << endl << "Определение размера множества с: " << c() << endl;

cout << endl;

}

**Результат**



**UML – диаграмма**

**Контрольные вопросы:**

* 1. **Что такое абстрактный тип данных? Привести примеры АТД.**

Это тип данных, доступ к которым осуществляется только через некоторый набор действий (операций, команд). Этот набор действий называется интерфейсом абстрактного типа данных.

Основная идея абстракции в программировании как раз и заключается в отделении интерфейса от реализации.

Пример: контейнер

* 1. **Привести примеры абстракции через параметризацию.**

Через параметризацию является концепция «функция», передаваемые параметры которой являются формальными, а фактические связываются с ними в момент использования такой абстракции. Программисты часто используют абстракцию через параметризацию, даже не замечая этого.

* 1. **Привести примеры абстракции через спецификацию.**

Абстракции через спецификацию реализуются, например, в виде библиотечных функций, широко используемых программистами.

* 1. **Что такое контейнер?**

Контейнер это набор некоторого количества обязательно однотипных элементов, упакованных в контейнер определённым образом.   
  
Простейшим прототипом контейнера в классическом языке C++ является массив.

* 1. **Какие группы операций выделяют в контейнерах?**

Вставка, удаление элементов, получение данных, хранящихся в ячейке, перемещение по элементам, поиск элементов, объединение контейнеров, специальные операции, которые зависят от типа контейнера.

* 1. **Какие виды доступа к элементам контейнера существуют?**

Доступ к элементам может осуществляться: последовательно, напрямую и ассоциативно.

Доступ напрямую подразумевает доступ к элементу по его индексу ( Пример: a[10]); доступ ассоциативно выполняется также по индексу, но индексом будет не номер элемента, а его содержимое(Пример:a[“word”]); при последовательном доступе перемещение осуществляется от элемента к элементу контейнера и содержит набор операций последовательного доступа (Пример: v.first, v.last, v.next, v.prevи т.д)

* 1. **Что такое итератор?**

Это класс, который позволяет перемещаться по элементам контейнера из функции main.

* 1. **Каким образом может быть реализован итератор?**

Итератор может быть реализован как часть класса-контейнера в виде набора методов:  
 v.first, v.last, v.next, v.prev, v.skip и т.д.

* 1. **Каким образом можно организовать объединение контейнеров?**
  + Простое сцепление двух контейнеров: в новый контейнер попадают сначала элементы первого контейнера, потом второго, операция не коммутативна.
  + Объединение упорядоченных контейнеров, новый контейнер тоже будет упорядочен, операция коммутативна.
  + Объединение контейнеров как объединение множеств, в новый контейнер попадают только те элементы, которые есть хотя бы в одном контейнере, операция коммутативна.
  + Объединение контейнеров как пересечение множеств, в новый контейнер попадают только те элементы, которые есть в обоих контейнерах, операция коммутативна
  + Для контейнеров-множеств может быть еще реализована операция вычитания, в контейнер попадают только те элементы первого контейнера, которых нет во втором, операция некоммутативна.
  + Извлечение части элементов из контейнера и создание нового контейнера. Эта операция может быть выполнена с помощью конструктора, а часть контейнера задается двумя итераторами.
  1. **Какой доступ к элементам предоставляет контейнер, состоящий из элементов «ключ-значение»?**

Ассоциативный доступ.

* 1. **Как называется контейнер, в котором вставка и удаление элементов выполняется на одном конце контейнера?**

Стек

* 1. **Какой из объектов (a,b,c,d) является контейнером?**
     1. Int mas=10;
     2. int mas;
     3. struct{char name[30]; int age;} mas;
     4. int mas[100];
  2. **Какой из объектов (a,b,c,d)не является** **контейнером?**
     1. int a[]={1,2,3,4,5};
     2. Int mas[30];
     3. struct{charname[30]; intage;} mas[30];
     4. int mas;

1. **Контейнер реализован как динамический массив, в нем** **определена операция доступ по индексу. Каким будет доступ к элементам контейнера?**

Прямой доступ

1. **Контейнер реализован как линейный список. Каким будет доступ к элементам контейнера?**

Последовательный доступ