Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное учреждение высшего образования  
«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет  
Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т  
по лабораторной работе**

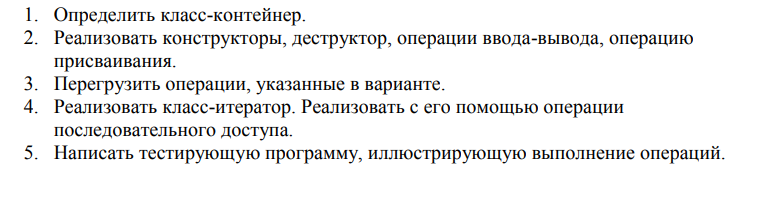
Дисциплина: «Основы алгоритмизации и программирования»

Выполнил  
студент группы ИВТ-21-1б  
Мифтахов И.Г

Проверил  
ст. п. кафедры ИТАС  
Яруллин Д.В.

Пермь 2022

Постановка задачи:

 Анализ задачи:

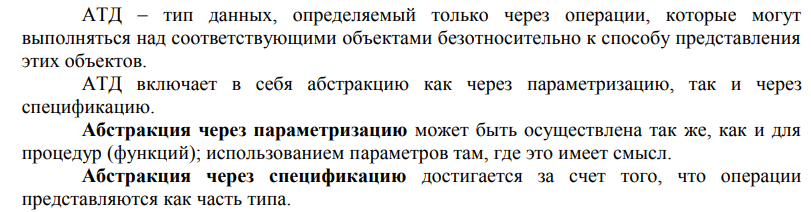
1. int size – количество элементов в множестве. Int k – индекс выбранного элемента. Int\*array – указатель для создания на его основе динамических массивов. Int \*elem – указатель на элемент множества.

2. Класс host является дружественным для класса Iterator. Все поля находятся в private секции, а методы в public. В классе Iterator реализованы перегруженные операторы.

3. В классе host реализован конструктор с параметрами, также реализованы перегруженные операторы.

4. В main() создаются объекты, вызываются их методы. Объект «с» будет являться результатом вычитания одного множества из другого. Вывод реализован при помощи объекта iter класса Iterator.

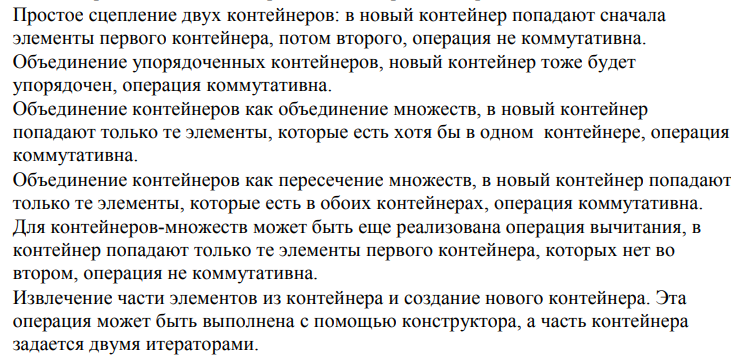
Ответы на вопросы:



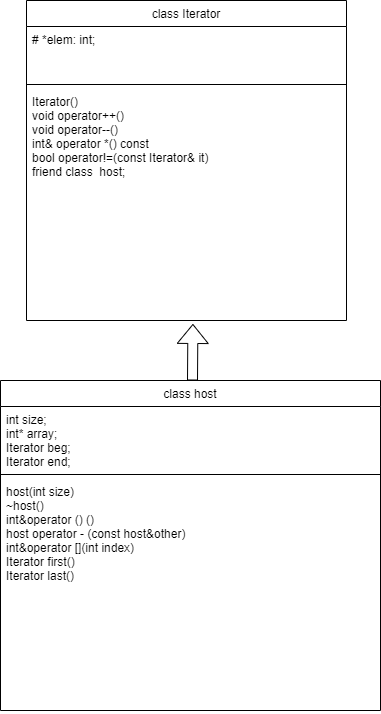
* Например у нас существует процедура сортировки массива:

Procedure Sort(Var A: TArray; N: Word);

+Если эта процедура получает сортируемый массив в качестве параметра, то может быть вызвана любое количество раз, причем для различных массивов. Код процедуры абстрагируется от конкретики данных, которые она обрабатывает.

* На примере той же процедуры сортировки: если спецификация написана следующим образом {Сортирует первые N элементов массива A по возрастанию. Предполагается, что массив индексируется с 0 (последний отсортированный элемент - N-1)}
* Контейнер – шаблон класса. Например, STL.
* Перегрузки операторов.
* Прямой – по индексу. Ассоциативный – кейс содержит в себе ключ и содержимое. Последовательный – доступ к элементам поступательно.
* Итератор – объект, обеспечивающий последовательный доступ к элементам.
* Можно реализовать как класс.
* 
* Ассоциативный.
* Очередь
* D
* D
* Прямым.
* Последовательный.

Блок-схема:



Код программы:

#include<iostream>

#include<ctime>

using namespace std;

class Iterator

{

private:

friend class host;

int\* elem;

public:

Iterator() { elem = 0; }

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

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

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

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

};

class host

{

private:

int size;

int\* array;

Iterator beg;

Iterator end;

public:

host(int size)

{

this->size = size;

array = new int[size];

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

{

array[i] = rand() % 100+1;

}

beg.elem = &array[0];

end.elem = &array[size];

}

~host()

{

}

int&operator () () { return size; }

host operator - (const host&other)

{

host tmp(size);

int\* temp = new int[size];

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

{

temp[i] = array[i] - other.array[i];

}

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

{

tmp.array[i] = temp[i];

}

tmp.beg.elem = &(tmp.array[0]);

tmp.end.elem = &(tmp.array[size]);

delete [] temp;

delete[]array;

return tmp;

}

int&operator [](int index)

{

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

{

return array[index];

}

else

{

cout <<endl<< "Error" << endl;

}

}

Iterator first() { return beg; }

Iterator last() { return end; }

};

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

int k = 1;

int size = 5;

host a(size);

Iterator iter;

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

{

cout << \*iter << " ";

}

cout << endl <<"Число с выбранным индексом: "<< a[k] << endl;

--iter;

cout <<"Предыдущий элемент: "<< \*iter << endl;

--iter;

cout << "Предыдущий элемент: " << \*iter << endl;

cout << "Длина вектора a: " << a() << endl;

host b(size);

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

{

cout << \*iter << " ";

}

cout <<endl<< "Длина вектора b: " << b() << endl;

host c = a - b;

cout << "Разность множеств равна - ";

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

{

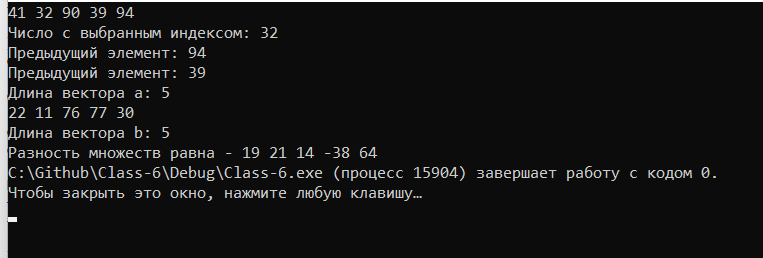
cout << \*iter << " ";

}

return 0;

}

Скриншоты результатов:



Анализ результатов:

Программа работает корректно. Выводятся множества, их длины, работает переход к предыдущему элементу. Разность также находится и выводится корректно.