Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

Лабораторная работа №6

АТД. Контейнеры

Выполнил

Студент группы РИС-22-2б

Вековшинин Д. А.

Проверила

Доцент кафедры ИТАС

Полякова О. А.

Пермь 2023

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

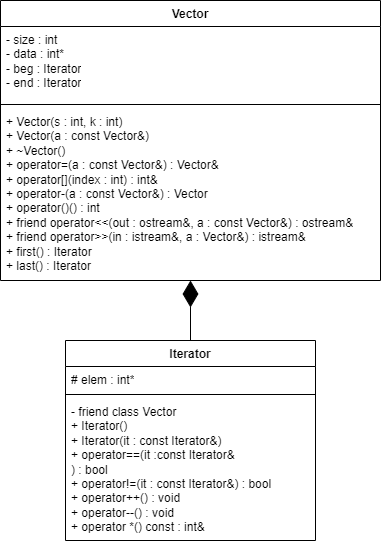
Вариант 10: Класс- контейнер Вектор с элементами типа int.

Реализовать операции:

[] – доступа по индексу;

() – определение размера вектора; - – разность множеств; -- – переход к предыдущему элементу (с помощью класса-итератора).

UML-диаграмма



Определение компонентных функций класса Vector

//конструктор с параметрами

Vector::Vector(int s, int k)

{

size = s;

data = new int[size];

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

data[i] = k;

beg.elem = &data[0];

end.elem = &data[size];

}

//конструктор копирования

Vector::Vector(const Vector& m)

{

size = m.size;

data = new int[size];

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

data[i] = m.data[i];

beg = m.beg;

end = m.end;

}

//деструктор

Vector::~Vector()

{

delete[]data;

data = 0;

}

//операция присваивания

Vector& Vector::operator=(const Vector& m)

{

if (this == &m)return \*this;

size = m.size;

if (data != 0) delete[]data;

data = new int[size];

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

data[i] = m.data[i];

beg = m.beg;

end = m.end;

return \*this;

}

//операция доступа по индексу

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

{

if (index < size) return data[index];

else cout << "\nError! Index>size";

}

//операция разности множеств

Vector Vector::operator-(const Vector& a)

{

Vector temp(size);

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

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

return temp;

}

//операция для получения длины вектора

int Vector::operator()()

{

return size;

}

//операции для ввода-выода

ostream& operator<<(ostream& out, const Vector& m)

{

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

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

return out;

}

istream& operator>>(istream& in, Vector& m)

{

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

in >> m.data[i];

return in;

}

Определение компонентных функций класса Iterator

//перегруженные операции сравнения

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

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

//перегруженная операция инкремент

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

//перегруженная операция декремент

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

//перегруженная операция разыменования

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

private:

int\* elem;//указатель на элемент типа int

Функция main()

void main()

{

setlocale(LC\_ALL, "ru");

Vector a(5);

cout << a << "\n";

cout << "Введите значения вектора A: " << endl;

cin >> a;

cout << a << "\n";

a[2] = 100;

cout << a << "\n";

cout << "Вектор B: " << endl;

Vector b(10);

cout << b << "\n";

b = a;

cout << b << "\n";

Vector c(5);

cout << "Введите значения вектора C: " << endl;

cin >> c;

Vector d(10);

d = b - c;

cout << "Разница векторов B и C:" << d << "\n";

cout << "Размер вектора A = " << a() << endl;

cout << \*(a.first()) << endl;

Iterator i = a.first();

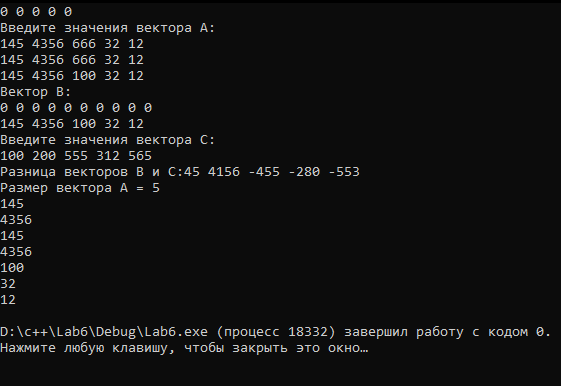
++i;

cout << \*i << endl;

for (i = a.first(); i != a.last(); ++i)cout << \*i << endl;

}

Результаты работы программы



Ответы на контрольные вопросы

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

АТД - тип данных, определяемый только через операции, которые могут

выполняться над соответствующими объектами безотносительно к способу представления этих объектов.

Примером абстрактного типа данных является класс в языке С++.

class Mnojestvo{};

1. Привести примеры абстракции через параметризацию.

Mnojestvo::Mnojestvo(int s, int k)

{

size = s;

data = new int[size];

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

data[i] = k;

beg.elem = &data[0];

end.elem = &data[size];

}

1. Привести примеры абстракции через спецификацию.

class Object

{

public:

virtual void Show() = 0;

virtual void Input() = 0;

};

class Print :

public Object

{

public:

Print(void);

public:

virtual ~Print(void);

void Show();

protected:

string name;

string author;

};

void Print::Show()

{

cout << "\nNAME : " << name;

cout << "\nAUTHOR : " << author;

cout << "\n";

}

void Print::Input()

{

cout << "\nName: "; cin >> name;

cout << "\nAuthor: "; cin >> author;

}

Класс Object является абстрактным классом, так как содержит чисто

виртуальные методы Show и Input, которые не имеют конкретной

реализации в этом классе. Эти методы переопределены в производном классе

Print.

1. Что такое контейнер? Привести примеры.

Контейнер – набор однотипных элементов.

Класс-контейнер МНОЖЕСТВО с элементами типа int:

class Mnojestvo

{

public:

//конструктор с параметрами: выделяет память под s элементов и заполняет их

//значением k

Mnojestvo(int s, int k = 0);

//конструктор с параметрами

Mnojestvo(const Mnojestvo& a);

//деструктор

~Mnojestvo();

//оператор присваивания

Mnojestvo& operator=(const Mnojestvo& a);

//операция доступа по индексу

int& operator[](int index);

//операция для добавление константы

Mnojestvo operator-(const Mnojestvo& a);

//операция, возвращающая длину вектора

int operator()();

//перегруженные операции ввода-вывода

friend ostream& operator<<(ostream& out, const Mnojestvo& a);

friend istream& operator>>(istream& in, Mnojestvo& a);

Iterator first() { return beg; }//возвращает указатель на первый элемент

Iterator last() { return end; }//возвращает указатель на элемент следующий за

//последним

private:

int size;

int\* data;

Iterator beg;//указатель на первый элемент вектора

Iterator end;//указатель на элемент следующий за последним

};

1. Какие группы операций выделяют в контейнерах?

Среди всех операций контейнера можно выделить несколько типовых групп:

* Операции доступа к элементам, которые обеспечивают и операцию замены значений элементов;
* Операции добавления и удаления элементов или групп элементов;
* Операции поиска элементов и групп элементов;
* Операции объединения контейнеров;
* Специальные операции, которые зависят от вида контейнера

1. Какие виды доступа к элементам контейнера существуют? Привести примеры.

Доступ к элементам контейнера бывает: последовательный, прямой и

ассоциативный.

Прямой доступ – это доступ по индексу. Например, a[10] – требуется найти

элемент контейнера с номером 10. В С++ нумерацию элементов контейнера принято начинать с нуля.

Ассоциативный доступ также выполняется по индексу, но индексом будет являться не номер элемента, а его содержимое. Пусть имеется контейнер –словарь, в котором хранится информация, состоящая, как минимум из двух полей: слово и его перевод. Индексом может служить слово, например, a[“word”]. С этим словом будет связано слово- перевод. Поле, с содержимым которого ассоциируется элемент контейнера, называется ключом или полем доступа. Элемент, с которым ассоциируется ключ, называется значением. Контейнер, который представляет ассоциативный доступ, состоит из пар «ключ-значение».

1. Что такое итератор?

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

1. Каким образом может быть реализован итератор?

Итератор может быть реализован как часть класса-контейнера в виде набора методов:

v.first() перейти к первому элементу

v.last() перейти к последнему элементу

v.next() перейти к следующему элементу

v.prev() перейти к предыдущему элементу

v.skip(n) перейти на n элементов вперед

v.skip(-n) перейти на n элементов назад

v.current() получить текущий элемент

Итератор можно реализовать как класс, представляющий такой же набор операций.

В С++ итератор реализуется как класс, который имеет такой же интерфейс, как и указатель для совместимости с массивами.

Если объект-итератор имеет имя iterv, то операции могут быть представлены

следующим образом:

iterv=v.first() перейти к первому элементу

iterv=v.last() перейти к последнему элементу

iterv++ перейти к следующему элементу

iterv-- перейти к предыдущему элементу

iterv+=n перейти на n элементов вперед

v.skip-=n перейти на n элементов назад

\*iterv получить текущий элемент

1. Каким образом можно организовать объединение контейнеров?

Наиболее часто используется операция объединения двух контейнеров с получением нового контейнера. Она может быть реализована в разных вариантах:

* 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 не коммутативна.
* 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 (по ключу)

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?

a. int mas=10;

b. int mas;

c. struct {char name[30]; int age;} mas;

d. int mas[100];

Ответ: d

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

Ответ: d

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