Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

Семестр: 2

На тему: «Классы и объекты. Инкапсуляция»

Выполнил студент ИВТ-22-2б:

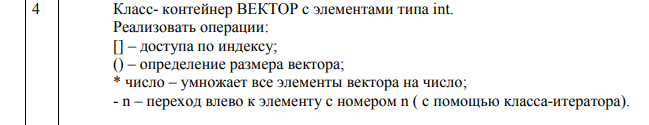
Захаров Дмитрий Сергеевич

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**1 Постановка задачи**

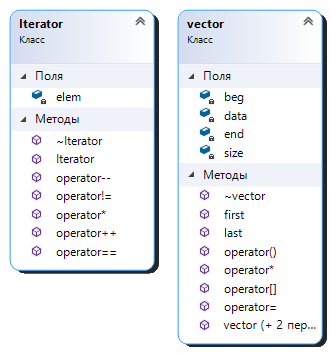
****

**2 Анализ задачи**

АТД – тип данных, определяемый только через операции, которые могут выполняться над соответствующими объектами безотносительно к способному представления этих объектов.

Контейнер – набор однотипных элементов. Встроенные в массив С++ - частный случай контейнера.

**3 UML-диаграмма**

****

**4 Код программы**

#include <iostream>

using namespace std;

class vector;

class Iterator {

friend class vector;

private:

int\* elem;

public:

Iterator() {

elem = 0;

}

~Iterator() {

}

bool operator==(const Iterator& i) {

return elem == i.elem;

}

bool operator!=(const Iterator& i) {

return elem != i.elem;

}

void operator++() {

elem++;

}

void operator--() {

--elem;

}

int& operator\*() const {

return \*elem;

}

};

class vector {

private:

int size;

int\* data;

Iterator beg;

Iterator end;

public:

vector(int s, int k = 0);

vector(const vector& a);

~vector();

vector();

vector& operator=(const vector& a);

int& operator[](int index);

vector operator\*(const int k);

int operator()();

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

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

Iterator first() {

return beg;

}

Iterator last() {

return end;

}

friend Iterator operator-(vector& a, int n);

};

Iterator operator-(vector& a, int n) {

for (int i = 0; i <= a.size - n; i++) {

--a.end;

}

return a.end;

}

vector::vector() {

size = 0;

data = 0;

}

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];

};

vector::vector(const vector& a) {

size = a.size;

data = new int[size];

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

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

beg = a.beg;

end = a.end;

};

vector::~vector() {

delete[] data;

data = 0;

};

vector& vector:: operator=(const vector& a) {

if (this == &a)

return \*this;

size = a.size;

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

data = new int[size];

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

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

beg = a.beg;

end = a.end;

return \*this;

};

int& vector:: operator[](int index) {

if (index < size || index>0) return data[index];

else cout << "Такого индекса нет!";

};

vector vector:: operator\*(const int k) {

vector tmp(size);

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

tmp.data[i] += data[i] \* k;

return tmp;

};

int vector:: operator()() {

return size;

};

ostream& operator<<(ostream& out, const vector& a) {

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

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

return out;

};

istream& operator>>(istream& in, vector& a) {

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

in >> a.data[i];

return in;

};

int main()

{

setlocale(LC\_ALL, "ru");

vector a(5);

cout << a << endl;

cin >> a;

cout << "Контейнер a: " << a << endl;

vector b(10);

b = a;

cout <<"Контейнер b: " << b << endl;

vector c(10);

c = b \* 4;

cout << "Контейнер с(b\*4):" << c << endl;

cout <<"Первый элемент: " << \*(a.first()) << endl;

Iterator it = a.first();

++it;

cout << "Второй элемент: " << \*it << endl;

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

cout << \*it << " ";

int n;

cout << endl << "Введите номер элемента, который нужно вывести: ";

cin >> n;

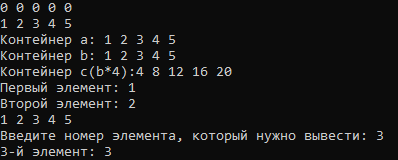
it = a - n;

cout << n << "-й " << "элемент: " << \*it;

return 0;

}

**5 Результаты работы программы**

****

**6 Ответы на вопросы**

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

Абстрактный тип данных (АТД) - совокупность данных и выполняемых над ними операций.

template <typename T>

T& Stack<T>::push()

{

return head->data;

}

1. Привести примеры абстракции через параметризацию.

Абстракция через параметризацию :

- Параметр определяется некоторым изменяемым типом данных при создании класса. Такой класс может быть использован для работы с разными типами данных. Класс переписываться для каждого нового типа данных не будет.

template <typename T>

class node

{

T data;

node\* next, \* prev;

}

1. Привести примеры абстракции через спецификацию.

Абстракция через спецификацию:

- позволяет абстрагироваться от процесса вычислений, описанных в теле процедуры, до уровня знания того, что данная процедура должна в итоге реализовать. Это достигается путём задания для каждой процедуры спецификации, описывающей эффект этой работы. При этом смысл обращения к процедуре становится ясным через

анализ её спецификации, а не тела процедуры.

class bynar

{

double first, second;

friend bynar operator+(bynar, bynar);

public:

bynar() { first = second = 0; }

bynar(double r) { first = r; second = 0; }

bynar(double r, double i) { first = r; second = i; }

~bynar() {};

};

bynar operator+(bynar a1, bynar a2)

{

return bynar(a1.first + a2.first, a1.second + a2.second);

}

1. Что такое контейнер? Привести примеры.

Контейнер - набор однотипных элементов, встроенные массивы в С++ частный случай контейнера. Контейнер – это объект. Имя контейнера – это имя переменной. Контейнер, так же как и другие объекты, обладает временем жизни. Время жизни контейнера в общем случае не зависит от времени жизни его элементов. Элементами контейнера могут любые объекты, в том числе, и другие контейнеры.

class stack

{

public:

int size;

double inf;

stack\* head, \* tail;

stack();

~stack() {};

stack(int, double, stack\*, stack\*);

};

void main()

{

stack\* q; //контейнер стэк

double a = 3.14;

q = q->head;

q->inf = a; //головой очереди q является переменная типа double

stack\* c;

c->tail = q; //хвостом очереди c является контейнер stack

}

1. Какие группы операций выделяют в контейнерах?
2. Операции доступа к элементам, которые обеспечивают и операцию замены значений элементов.

class stack

{

int size;

T\* head, \* tail;

public:

T& getHead() { return head->data; }

void stack::setHead(T head) { this->head = head; }

...

};

1. Операции добавления и удаления элементов или групп элементов.

T stack::pop()

{

T data = tail->data;

size--;

tail = tail->prev;

tail->next = nullptr;

return data;

}

1. Поиск элементов

int stack::search(T data)

{

int counter = 0;

T\* curr = this->head;

while (counter < size)

{

if (curr->data = data) return counter;

curr = curr->next;

counter++;

}

}

1. Объединение контейнеров

5)Специальные операции, зависящие от состава контейнера.

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

Доступ к элементам контейнера бывает: последовательный, прямой и ассоциативный.

- Последовательный доступ - это доступ, при котором осуществляется перемещение от одного элемента контейнера к другому.

- Прямой доступ – это доступ по индексу. Например, a[10] – требуется найти элемент контейнера с номером 10.

- 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».

int stack::find\_index(T data) // прямой доступ

{

int counter = 0;

node\* curr = this->head;

while (counter < size)

{

if (curr->data = data) return counter;

curr = curr->next; // переходит от одного к другому

counter++;

}

}

1. Что такое итератор?

Итератор – это объект, который обеспечивает последовательный доступ к элементам контейнера. Итератор может быть реализован как часть класса-контейнера в виде набора методов. 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.

1. Каким образом может быть реализован итератор?

class Iterator

{

int\* elem;

friend class Vector;//дружественный класс

public:

Iterator() { elem = 0; }

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

//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; }

};

1. Каким образом можно организовать объединение контейнеров?

stack stack::merge(stack q1, stack q2) {

q1->tail->next = q2->head;

q1->tail = q2->tail;

delete q2;

return q1;

}

1. Какой доступ к элементам предоставляет контейнер, состоящий из элементов «ключ-значение»?

Ассоциативный.

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

Такой контейнер называется стэк.

1. Какой из объектов (a,b,c,d) является контейнером?

a.int mas=10; // Переменная, равная 10

b.2. int mas; // Объявленная переменная

c.3. struct {char name[30]; int age;} mas; // Структура

d.4. int mas[100]; // Массив, то есть контейнер

Контейнером является объект d.

1. Какой из объектов (a,b,c,d) не является контейнером?

a. int a[]={1,2,3,4,5}; // Инициализированный массив

b.2. int mas[30]; // Массив на 30 элементов

c.3. struct {char name[30]; int age;} mas[30]; //массив структур

d.4. int mas; // Переменная

Контейнером не будет являться объект d.

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

Если индекс - это обычный индекс в массиве, то доступ будет прямым. Если индекс - это ключ, то доступ будет ассоциативным.

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

При линейном списке, где у элементов нет индексов, доступ будет последовательным.