Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «АТД. Контейнеры»

Семестр 2

Выполнил работу:

Студент группы ИВТ-22-2б

Голубцов Никита Валерьевич

Проверил:

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Г. 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.

**Описание класса**

Класс- контейнер ВЕКТОР с элементами типа int.

Реализовать операции:

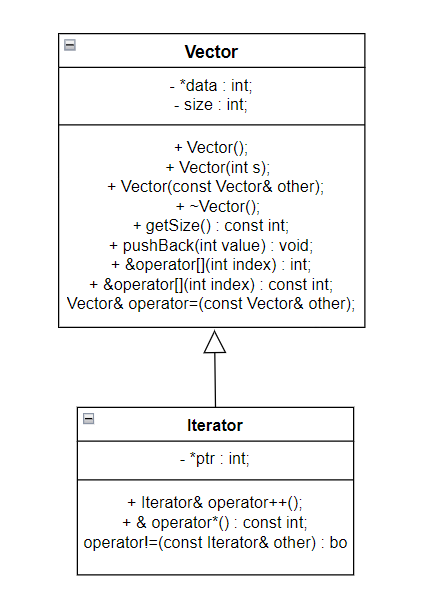
[]– доступа по индексу;

int() – определение размера вектора;

+ вектор – сложение элементов векторов a[i]+b[i];

+n - переход вправо к элементу c номером n ( с помощью класса-итератора).

**UML - Диаграмма**



**Рис. 1 – UML-диаграмма классов Vector и Iterator**

**Контрольные 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.