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 ПО ЛАБОРАТОРНОЙ РАБОТЕ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема : "АТД. Контейнеры"

Выполнила работу

Студентка группы РИС-22-1Б

Верхоланцева Е. С.

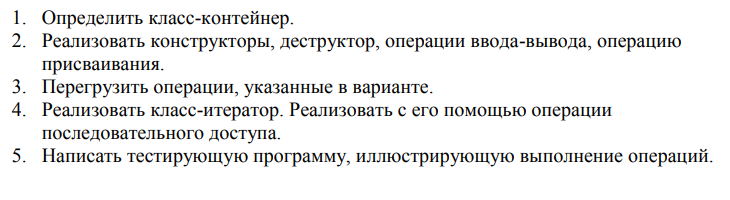
Проверила

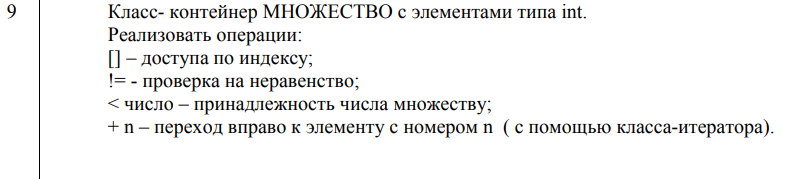
Доцент кафедры ИТАС

Полякова О.А.

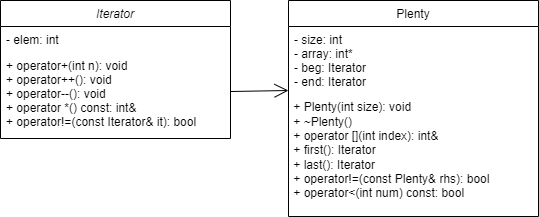
Г. Пермь-2023

**Постановка задачи**





**Диаграммы классов**



**Описание класса-контейнера**

class Plenty

{

private:

int size;

int\* array;

Iterator beg;

Iterator end;

public:

Plenty(int size);

~Plenty();

int& operator [](int index);

Iterator first();

Iterator last();

bool operator!=(const Plenty& rhs);

bool operator<(int num) const;

};

**Определение компонентных функций**

// конструктор

Plenty::Plenty(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];

}

// деструктор

Plenty::~Plenty() {}

// операция проверки на неравенство

bool Plenty::operator!=(const Plenty& rhs) {

bool f = false;

int i = 0;

while (!f) {

if (array[i] != rhs.array[i]) f = true;

i++;

}

return f;

}

// операция доступа по индексу

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

{

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

{

return array[index];

}

else

{

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

}

}

bool Plenty::operator<(int num) const

{

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

if (array[i] == num) {

return true;

}

}

return false;

}

Iterator Plenty::first() { return beg; }

Iterator Plenty::last() { return end; }

**Описание класса-итератора и его компонентных функций**

class Iterator

{

private:

friend class Plenty;

int\* elem;

public:

Iterator();

// перегрузка операции +n - переход вправо к элементу с номером n

void operator+(int n);

void operator++();

void operator--();

int& operator \*() const;

bool operator!=(const Iterator& it);

};

Iterator::Iterator() { elem = 0; }

void Iterator::operator+(int n) { elem = elem + n; }

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

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

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

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

**Функция main**

int main()

{

setlocale(LC\_ALL, "rus");

srand(time(0));

int size = 10, k;

Plenty a(size);

Iterator iter;

cout << "Множество а: ";

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

{

cout << \*iter << " ";

}

cout << endl;

cout << "Введите индекс элемента: "; cin >> k;

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

cout << endl;

int c;

cout << "Введите число, которое хотите проверить на принадлежность множеству а: "; cin >> c;

if (a < c) cout << "Число " << c << " принадлежит множеству а." << endl;

else cout << "Число " << c << " не принадлежит множеству а." << endl;

Plenty b(size);

cout << "Множество b: ";

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

{

cout << \*iter << " ";

}

cout << endl;

cout << "Операция проверки на неравенство:" << endl;

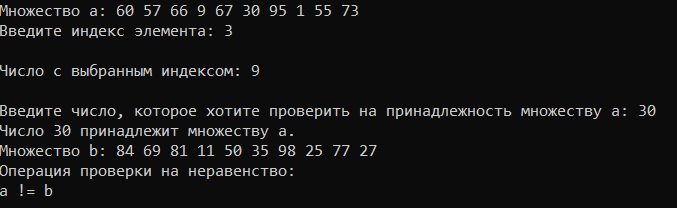
if (a != b) cout << "a != b" << endl;

else cout << "a = b" << endl;

return 0;

}

**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.

Примеры: стек, очередь, список

2. Привести примеры абстракции через параметризацию.

функция, передаваемые параметры которой являются формальными, а фактические связываются с ними в момент использования такой абстракции.

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

3. Привести примеры абстракции через спецификацию.

позволяет абстрагироваться от процесса вычислений, описанных в теле процедуры, до уровня знаний лишь того, что данная процедура должна в итоге реализовать.

4. Что такое контейнер? Привести примеры.

Контейнер - набор однотипных элементов. Самый простой пример контейнера - массив.

5. 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.

6. 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.   
При последовательном доступе осуществляется перемещение от элемента к элементу контейнера.

7. Что такое итератор?

Итератор - это объект, который обеспечивает доступ к его элементам, используя указатели.

8. Каким образом может быть реализован итератор?

Можно реализовать как класс или как часть класса.

9. 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 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.

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

Ассоциативный доступ.

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

Такой контейнер называется стеком.

12. 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

13. Какой из объектов (a,b,c,d) не является контейнером?  
a. int a[]={1,2,3,4,5};

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

Ответ: d

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

Последовательный доступ.

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

Прямой доступ.