Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №18.6**

Дисциплина: «Основы теории алгоритмов и структуры данных»  
Тема: Абстрактные типы данных. Контейнеры

Вариант 14

Выполнил:

студент группы РИС-20-2б

Вичугов Алексей Дмитриевич

Проверила:

доцент кафедры ИТАС

Полякова Ольга Андреевна

Пермь, 2021

**Цель работы**

1. Создание консольного приложения, состоящего из нескольких файлов.

2. Реализация класса-контейнера.

**Постановка задачи**

1. Определить класс-контейнер

2. Реализовать конструкторы, деструктор, операции ввода-вывода, операцию присваивания.

3. Перегрузить необходимые операции.

4. Реализовать класс-итератор с помощью операции последовательного доступа.

5. Написать тестирующую программу, иллюстрирующую выполнение операций.

6. Задание варианта: Класс-контейнер список с ключевыми значениями типа int. Реализовать операции доступа по индексу, определения размера списка, умножения всех элементов списка на число и операцию перехода влево к элементу с номером n.

**Анализ задачи**

1. Описание класса:

struct S{

int val;

S \*sled, \*pred;

};

class Lister{

public:

S \*first, \*last;

int len;

Lister();

Lister(S \*f, S \*l);

Lister(Lister &l);

~Lister();

void Create(int r);

int operator [](int index);

int operator ()();

void operator \*(int n);

friend ostream& operator <<(ostream& ost, Lister &l);

};

class Iterator{

public:

Lister \*list;

S \*cur;

Iterator();

Iterator(Lister \*l);

Iterator(Iterator &i);

~Iterator();

S\* operator -(int n);

S\* operator --();

S\* operator ++();

};

2. Определение компонентных функций:

Lister::Lister(){len=0;first=NULL;last=NULL;}

Lister::Lister(S \*f, S \*l){first=f;last=l;}

Lister::Lister(Lister &l){first=l.first;last=l.last;len=l.len;}

Lister::~Lister(){}

void Lister::Create(int r){

if (r<1) cout << "List must contain 1 or more elements! " << endl;

else{

len=1;

first = new S;

first->pred=NULL;

first->sled=NULL;

S\* cur=first;

S\* nov=NULL;

cout << "1: ";

cin >> cur->val;

for (int i=1;i<r;i++){

len++;

nov = new S;

nov->pred=cur;

nov->sled=NULL;

cur->sled=nov;

cur=nov;

cout << i+1 << ": ";

cin >> cur->val;

}

last=cur;

}

}

int Lister::operator()(){

return len;

}

int Lister::operator [](int index){

bool b=0;

S \*cur=first;

if(len>0){

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

if (cur->sled!=NULL){

cur=cur->sled;

}

else b=1;

}

}

else b=1;

if(b) return 0;

else return cur->val;

}

void Lister::operator \*(int n){

if(len>0){

S \*cur=first;

while (cur->sled!=NULL){

cur->val\*=n;

cur=cur->sled;

}

cur->val\*=n;

}

}

Iterator::Iterator(){list=NULL;cur=NULL;}

Iterator::Iterator(Lister \*l){list=l;cur=l->first;}

Iterator::Iterator(Iterator &i){list=i.list;cur=i.cur;}

Iterator::~Iterator(){}

S\* Iterator::operator -(int n){

if (cur!=NULL)

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

if(cur->pred!=NULL) cur=cur->pred;

}

return cur;

}

S\* Iterator::operator --(){

if (cur!=NULL&&cur->pred!=NULL) cur=cur->pred;

return cur;

}

S\* Iterator::operator ++() {

if (cur!=NULL&&cur->sled!=NULL) cur=cur->sled;

return cur;

}

3. Определение глобальных функций:

ostream& operator << (ostream& ost, Lister &l){

S\* cur=l.first;

while(cur->sled!=NULL){

ost << cur->val << " ";

cur=cur->sled;

}

ost << cur->val << endl;

return ost;

}

4. Функция main():

main(){

Lister cont;

cont.Create(-7);

cont.Create(5);

cout << cont;

cout << cont[3] << endl;

cout << "Length: " << cont() << endl;

Iterator ter(&cont);

++ter;

cout << ter.cur->val << endl;

--ter;

cout << ter.cur->val << endl;

++ter;

++ter;

++ter;

++ter;

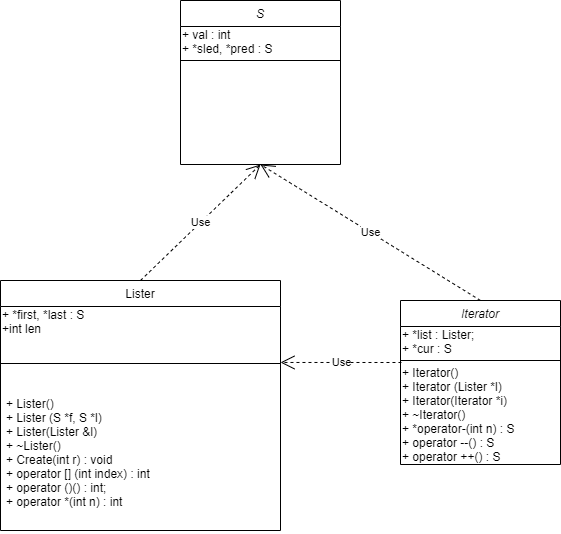
ter-2;

cout << ter.cur->val << endl;

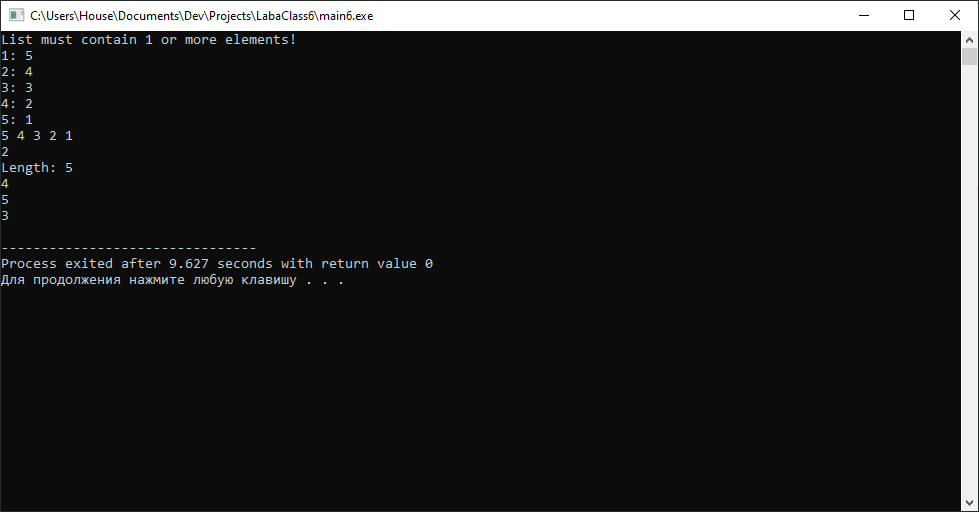
return 0;

}

**Диаграмма классов**



**Результаты работы программы**

****

**Ответы на вопросы**

1. Это тип данных, определяемый только через операции, которые могут выполняться над объектами этого типа данных, без указания способа представления этих объектов. Абстрактным типом данных является класс.
2. Class A{…}; void Show(&A);
3. Class B{public: void Show(){…}}; B b; b.Show();
4. Набор однотипных элементов, например, связанный список или стек.
5. Операции: доступ к элементам; добавление и удаление элементов; поиск элементов; объединение контейнеров; особые операции для определённых видов контейнеров.
6. Последовательный – перемещение от элемента к элементу (например при поиске элемента в связанном списке); прямой – доступ по индексу (доступ к элементу массива); ассоциативный – доступ по ключу элемента (например, доступ к элементам контейнера multimap).
7. Это объект, который обеспечивает последовательный доступ к элементам контейнера.
8. Либо в виде части класса-контейнера, либо как отдельный класс.
9. Простое объединение двух контейнеров (заносятся элементы каждого контейнера), объединение множеств (заносятся все элементы, элементы в новом контейнере не повторяются), пересечение множеств (заносятся только элементы, имеющиеся в обоих контейнерах) и извлечение части элементов из существующего контейнера с созданием нового.
10. Ассоциативный.
11. Односторонний (стек или односторонняя очередь).
12. D.
13. D.
14. Прямым.
15. Последовательным.