Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет»**

Электротехнический факультет Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 - «Программная инженерия»

**Лабораторная работа №11**

**"Последовательные контейнеры библиотеки STL"**

Выполнил студент гр. РИС-24-3б Носков Егор Михайлович

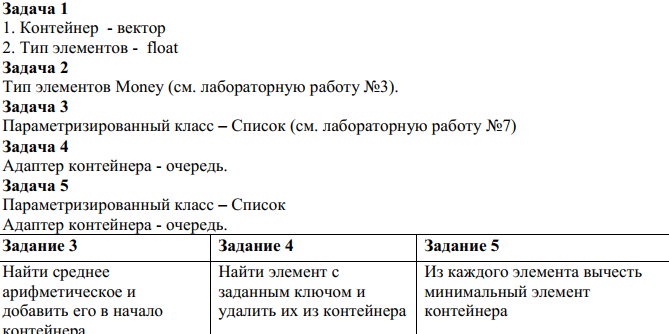
Проверил: Доцент кафедры ИТАС Ольга Андреевна Полякова

|  |  |
| --- | --- |
|  | |
| (оценка) (подпись) | |
|  |
| (дата) |

Г. Пермь, 2024

**Постановка задачи**

**Вариант 1:**

****

**Программа на языке C++**

Проект lab11OOP\_task1:

Lab11\_task1.cpp:

#include <iostream>

#include <deque>

using namespace std;

typedef deque<int>Deq;

Deq make\_deque(int n)

{

Deq d;

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

{

int a = rand() % 100 - 50;

d.push\_back(a);

}

return d;

}

void print\_deque(Deq d)

{

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

{

cout << d[i] << " ";

}

cout << endl;

}

Deq push\_back\_max(Deq d)

{

int max = NULL;

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

{

if (d[i] > max) max = d[i];

}

d.push\_back(max);

return d;

}

Deq remove\_elem(Deq d, int value)

{

for (deque<int>::iterator di = d.begin(); di != d.end();)

{

if (\*di == value)

di = d.erase(di);

else

di++;

}

return(d);

}

Deq add\_mean\_all(Deq d)

{

double sum = NULL;

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

{

sum += d[i];

}

double mean = sum / d.size();

cout << mean << "\n";

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

{

d[i] += (int)mean;

}

return d;

}

int main()

{

try

{

deque<int> d;

int n;

cout << "Enter n: "; cin >> n;

d = make\_deque(n);

cout << "deque: "; print\_deque(d);

d = push\_back\_max(d);

cout << "push\_back\_max: "; print\_deque(d);

int value;

cout << "Enter the value to remove: "; cin >> value;

d = remove\_elem(d, value);

cout << "remove\_elem: "; print\_deque(d);

d = add\_mean\_all(d);

cout << "add\_mean\_all: "; print\_deque(d);

}

catch (int)

{

cout << "Error!";

}

return 0;

}

Проект lab11OOP\_task2:

Money.h:

#include <iostream>

#include <string>

using namespace std;

class Money

{

long rub;

int kop;

private:

void recount();

public:

Money() { rub = 0; kop = 0; };

Money(long r, int k) { rub = r; kop = k; recount(); };

Money(const Money& m) { rub = m.rub; kop = m.kop; };

~Money() {};

long get\_rub() { return rub; }

int get\_kop() { return kop; }

void set\_rub(long r) { rub = r; }

void set\_kop(int k) { kop = k; }

//перегруженные операции

Money& operator=(const Money&);

Money operator-(float value);

bool operator==(const Money& m);

bool operator!=(const Money& m);

bool operator>(const Money& m);

bool operator<(const Money& m) const;

Money& operator+(const Money& m);

Money operator+(const Money& m) const;

Money operator/(int value);

//глобальные функции ввода-вывода

friend istream& operator>>(istream& in, Money& m);

friend ostream& operator<<(ostream& out, const Money& m);

};

Money.cpp:

#include <iostream>

#include "money.h"

using namespace std;

void Money::recount()

{

if (kop >= 100) {

rub += kop / 100;

kop %= 100;

}

else if (kop < 0) {

rub -= (abs(kop) / 100 + 1);

kop = 100 - abs(kop) % 100;

}

}

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

{

if (this != &m)

{

rub = m.rub;

kop = m.kop;

}

return \*this;

}

Money Money::operator-(float value)

{

long r = static\_cast<long>(value);

int k = static\_cast<int>((value - r) \* 100);

return Money(rub - r, kop - k);

}

bool Money::operator==(const Money& m)

{

return rub == m.rub && kop == m.kop;

}

bool Money::operator!=(const Money& m)

{

return rub != m.rub || kop != m.kop;

}

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

{

bool fail = false;

cout << "rub: "; in >> m.rub;

do

{

cout << "kop(0-99): "; in >> m.kop;

if (m.kop > 99 || m.kop < 0)

{

cout << "Error! Enter correct value.\n";

fail = true;

}

else fail = false;

} while (fail);

return in;

}

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

{

out << m.rub << ",";

if (m.kop < 10)

out << "0";

out << m.kop;

return out;

}

bool Money::operator>(const Money& m)

{

if (rub > m.rub) return true;

if (rub == m.rub)

if (kop > m.kop) return true;

return false;

}

bool Money::operator<(const Money& m) const

{

if (rub != m.rub) return rub < m.rub;

return kop < m.kop;

}

Money& Money::operator+(const Money& m)

{

rub += m.rub;

kop += m.kop;

recount();

return \*this;

}

Money Money::operator+(const Money& m) const

{

Money temp;

temp.rub = rub + m.rub;

temp.kop = kop + m.kop;

temp.recount();

return temp;

}

Money Money::operator/(int value)

{

int m = rub \* 100 + kop;

m /= value;

rub = 0;

kop = m;

recount();

return \*this;

}

Lab11\_task2.cpp:

#include <iostream>

#include <deque>

#include "Money.h"

using namespace std;

typedef deque<Money>Deq;

Deq make\_deque(int n)

{

Deq d;

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

{

long r = rand() % 1000;

int k = rand() % 100;

Money m(r, k);

d.push\_back(m);

}

return d;

}

void print\_deque(Deq d)

{

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

{

cout << d[i] << " ";

}

cout << endl;

}

Deq push\_back\_max(Deq d)

{

Money max = d[0];

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

{

if (d[i] > max) max = d[i];

}

d.push\_back(max);

return d;

}

Deq remove\_elem(Deq d, Money value)

{

for (deque<Money>::iterator di = d.begin(); di != d.end();)

{

if (\*di == value)

di = d.erase(di);

else

di++;

}

return(d);

}

Deq add\_mean\_all(Deq d)

{

Money sum;

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

{

sum = sum + d[i];

}

cout << sum << "\n";

Money mean = sum / d.size();

cout << mean << "\n";

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

{

d[i] = d[i] + mean;

}

return d;

}

int main()

{

try

{

deque<Money> d;

int n;

cout << "Enter n: "; cin >> n;

d = make\_deque(n);

cout << "deque: "; print\_deque(d);

d = push\_back\_max(d);

cout << "push\_back\_max: "; print\_deque(d);

Money value;

cout << "Enter the value to remove: \n"; cin >> value;

d = remove\_elem(d, value);

cout << "remove\_elem: "; print\_deque(d);

d = add\_mean\_all(d);

cout << "add\_mean\_all: "; print\_deque(d);

}

catch (int)

{

cout << "Error!";

}

}

Проект lab11OOP\_task3:

Money.h:

#include <iostream>

#include <string>

using namespace std;

class Money

{

long rub;

int kop;

private:

void recount();

public:

Money() { rub = 0; kop = 0; };

Money(long r, int k) { rub = r; kop = k; recount(); };

Money(const Money& m) { rub = m.rub; kop = m.kop; };

~Money() {};

long get\_rub() { return rub; }

int get\_kop() { return kop; }

void set\_rub(long r) { rub = r; }

void set\_kop(int k) { kop = k; }

//перегруженные операции

Money& operator=(const Money&);

Money operator-(float value);

bool operator==(const Money& m);

bool operator!=(const Money& m);

bool operator>(const Money& m);

bool operator<(const Money& m) const;

Money& operator+(const Money& m);

Money operator+(const Money& m) const;

Money operator/(int value);

//глобальные функции ввода-вывода

friend istream& operator>>(istream& in, Money& m);

friend ostream& operator<<(ostream& out, const Money& m);

};

Money.cpp:

#include <iostream>

#include "money.h"

using namespace std;

void Money::recount()

{

if (kop >= 100) {

rub += kop / 100;

kop %= 100;

}

else if (kop < 0) {

rub -= (abs(kop) / 100 + 1);

kop = 100 - abs(kop) % 100;

}

}

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

{

if (this != &m)

{

rub = m.rub;

kop = m.kop;

}

return \*this;

}

Money Money::operator-(float value)

{

long r = static\_cast<long>(value);

int k = static\_cast<int>((value - r) \* 100);

return Money(rub - r, kop - k);

}

bool Money::operator==(const Money& m)

{

return rub == m.rub && kop == m.kop;

}

bool Money::operator!=(const Money& m)

{

return rub != m.rub || kop != m.kop;

}

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

{

bool fail = false;

cout << "rub: "; in >> m.rub;

do

{

cout << "kop(0-99): "; in >> m.kop;

if (m.kop > 99 || m.kop < 0)

{

cout << "Error! Enter correct value.\n";

fail = true;

}

else fail = false;

} while (fail);

return in;

}

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

{

out << m.rub << ",";

if (m.kop < 10)

out << "0";

out << m.kop;

return out;

}

bool Money::operator>(const Money& m)

{

if (rub > m.rub) return true;

if (rub == m.rub)

if (kop > m.kop) return true;

return false;

}

bool Money::operator<(const Money& m) const

{

if (rub != m.rub) return rub < m.rub;

return kop < m.kop;

}

Money& Money::operator+(const Money& m)

{

rub += m.rub;

kop += m.kop;

recount();

return \*this;

}

Money Money::operator+(const Money& m) const

{

Money temp;

temp.rub = rub + m.rub;

temp.kop = kop + m.kop;

temp.recount();

return temp;

}

Money Money::operator/(int value)

{

int m = rub \* 100 + kop;

m /= value;

rub = 0;

kop = m;

recount();

return \*this;

}

Vector.h:

#pragma once

#include <iostream>

#include <vector>

using namespace std;

template<class T>

class Vector

{

vector <T> v;

int len;

public:

Vector(void);

Vector(int n);

void print\_vector();

~Vector(void);

void push\_back\_max();

void remove\_elem(const T& value);

void add\_mean\_all();

};

template <class T>

Vector<T>::Vector()

{

len = 0;

}

template <class T>

Vector<T>::~Vector(void) {}

template <class T>

Vector<T>::Vector(int n)

{

T a;

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

{

cin >> a;

v.push\_back(a);

}

len = v.size();

}

template <class T>

void Vector<T>::print\_vector()

{

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

cout << v[i] << " ";

cout << "\n";

}

template<class T>

void Vector<T>::push\_back\_max()

{

T max;

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

{

if (v[i] > max) max = v[i];

}

v.push\_back(max);

}

template<class T>

void Vector<T>::remove\_elem(const T& value)

{

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

{

if (v[i] == value) v.erase(v.begin() + i);

}

}

template<class T>

void Vector<T>::add\_mean\_all()

{

T sum;

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

{

sum = sum + v[i];

}

cout << sum << "\n";

T mean = sum / v.size();

cout << mean << "\n";

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

{

v[i] = v[i] + mean;

}

}

Lab11\_task3.cpp:

#include <iostream>

#include "vector.h"

#include "C:\Users\emnos\Desktop\VS2022\LAB11OOP\lab11OOP\_task2\money.h"

int main()

{

Vector<Money>vec(5);

cout << "vector: "; vec.print\_vector();

vec.push\_back\_max();

cout << "push\_back\_max: "; vec.print\_vector();

Money value;

cout << "Enter the value to remove: \n"; cin >> value;

vec.remove\_elem(value);

cout << "remove\_elem: "; vec.print\_vector();

vec.add\_mean\_all();

cout << "add\_mean\_all: "; vec.print\_vector();

}

Проект lab11OOP\_task4:

Money.h:

#include <iostream>

#include <string>

using namespace std;

class Money

{

long rub;

int kop;

private:

void recount();

public:

Money() { rub = 0; kop = 0; };

Money(long r, int k) { rub = r; kop = k; recount(); };

Money(const Money& m) { rub = m.rub; kop = m.kop; };

~Money() {};

long get\_rub() { return rub; }

int get\_kop() { return kop; }

void set\_rub(long r) { rub = r; }

void set\_kop(int k) { kop = k; }

//перегруженные операции

Money& operator=(const Money&);

Money operator-(float value);

bool operator==(const Money& m);

bool operator!=(const Money& m);

bool operator>(const Money& m);

bool operator<(const Money& m) const;

Money& operator+(const Money& m);

Money operator+(const Money& m) const;

Money operator/(int value);

//глобальные функции ввода-вывода

friend istream& operator>>(istream& in, Money& m);

friend ostream& operator<<(ostream& out, const Money& m);

};

Money.cpp:

#include <iostream>

#include "money.h"

using namespace std;

void Money::recount()

{

if (kop >= 100) {

rub += kop / 100;

kop %= 100;

}

else if (kop < 0) {

rub -= (abs(kop) / 100 + 1);

kop = 100 - abs(kop) % 100;

}

}

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

{

if (this != &m)

{

rub = m.rub;

kop = m.kop;

}

return \*this;

}

Money Money::operator-(float value)

{

long r = static\_cast<long>(value);

int k = static\_cast<int>((value - r) \* 100);

return Money(rub - r, kop - k);

}

bool Money::operator==(const Money& m)

{

return rub == m.rub && kop == m.kop;

}

bool Money::operator!=(const Money& m)

{

return rub != m.rub || kop != m.kop;

}

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

{

bool fail = false;

cout << "rub: "; in >> m.rub;

do

{

cout << "kop(0-99): "; in >> m.kop;

if (m.kop > 99 || m.kop < 0)

{

cout << "Error! Enter correct value.\n";

fail = true;

}

else fail = false;

} while (fail);

return in;

}

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

{

out << m.rub << ",";

if (m.kop < 10)

out << "0";

out << m.kop;

return out;

}

bool Money::operator>(const Money& m)

{

if (rub > m.rub) return true;

if (rub == m.rub)

if (kop > m.kop) return true;

return false;

}

bool Money::operator<(const Money& m) const

{

if (rub != m.rub) return rub < m.rub;

return kop < m.kop;

}

Money& Money::operator+(const Money& m)

{

rub += m.rub;

kop += m.kop;

recount();

return \*this;

}

Money Money::operator+(const Money& m) const

{

Money temp;

temp.rub = rub + m.rub;

temp.kop = kop + m.kop;

temp.recount();

return temp;

}

Money Money::operator/(int value)

{

int m = rub \* 100 + kop;

m /= value;

rub = 0;

kop = m;

recount();

return \*this;

}

Lab11\_task4.cpp:

#include "C:\Users\emnos\Desktop\VS2022\LAB11OOP\lab11OOP\_task2\money.h"

#include <iostream>

#include <stack>

#include <vector>

using namespace std;

typedef stack<Money>St;

typedef vector<Money>Vec;

St make\_stack(int n)

{

St s;

Money m;

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

{

cin >> m;

s.push(m);

}

return s;

}

Vec copy\_stack\_to\_vector(St s)

{

Vec v;

while (!s.empty())

{

v.push\_back(s.top());

s.pop();

}

return v;

}

St copy\_vector\_to\_stack(Vec v)

{

St s;

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

{

s.push(v[i]);

}

return s;

}

void print\_stack(St& s)

{

Vec v = copy\_stack\_to\_vector(s);

while (!s.empty())

{

cout << s.top() << " ";

s.pop();

}

s = copy\_vector\_to\_stack(v);

cout << endl;

}

void push\_max(St& s)

{

Vec v = copy\_stack\_to\_vector(s);

Money max = s.top();

s.pop();

while (!s.empty())

{

if (s.top() > max) max = s.top();

s.pop();

}

s = copy\_vector\_to\_stack(v);

s.push(max);

}

void remove\_elem(St& s)

{

Money value;

cout << "Enter the value to remove : \n"; cin >> value;

Vec v;

while (!s.empty())

{

if (s.top() != value) v.push\_back(s.top());

s.pop();

}

s = copy\_vector\_to\_stack(v);

}

void add\_mean\_all(St& s)

{

Vec v = copy\_stack\_to\_vector(s);

Money sum;

while (!s.empty())

{

sum = sum + s.top();

s.pop();

}

cout << sum << "\n";

Money mean = sum / v.size();

cout << mean << "\n";

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

{

v[i] = v[i] + mean;

}

s = copy\_vector\_to\_stack(v);

}

int main()

{

Money m;

St s;

int n;

cout << "N: ";

cin >> n;

s = make\_stack(n);

cout << "Stack: "; print\_stack(s);

push\_max(s);

cout << "push\_max: "; print\_stack(s);

remove\_elem(s);

cout << "remove\_elem: "; print\_stack(s);

add\_mean\_all(s);

cout << "add\_mean\_all: "; print\_stack(s);

}

Проект lab11OOP\_task5:

Money.h:

#include <iostream>

#include <string>

using namespace std;

class Money

{

long rub;

int kop;

private:

void recount();

public:

Money() { rub = 0; kop = 0; };

Money(long r, int k) { rub = r; kop = k; recount(); };

Money(const Money& m) { rub = m.rub; kop = m.kop; };

~Money() {};

long get\_rub() { return rub; }

int get\_kop() { return kop; }

void set\_rub(long r) { rub = r; }

void set\_kop(int k) { kop = k; }

//перегруженные операции

Money& operator=(const Money&);

Money operator-(float value);

bool operator==(const Money& m);

bool operator!=(const Money& m);

bool operator>(const Money& m);

bool operator<(const Money& m) const;

Money& operator+(const Money& m);

Money operator+(const Money& m) const;

Money operator/(int value);

//глобальные функции ввода-вывода

friend istream& operator>>(istream& in, Money& m);

friend ostream& operator<<(ostream& out, const Money& m);

};

Money.cpp:

#include <iostream>

#include "money.h"

using namespace std;

void Money::recount()

{

if (kop >= 100) {

rub += kop / 100;

kop %= 100;

}

else if (kop < 0) {

rub -= (abs(kop) / 100 + 1);

kop = 100 - abs(kop) % 100;

}

}

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

{

if (this != &m)

{

rub = m.rub;

kop = m.kop;

}

return \*this;

}

Money Money::operator-(float value)

{

long r = static\_cast<long>(value);

int k = static\_cast<int>((value - r) \* 100);

return Money(rub - r, kop - k);

}

bool Money::operator==(const Money& m)

{

return rub == m.rub && kop == m.kop;

}

bool Money::operator!=(const Money& m)

{

return rub != m.rub || kop != m.kop;

}

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

{

bool fail = false;

cout << "rub: "; in >> m.rub;

do

{

cout << "kop(0-99): "; in >> m.kop;

if (m.kop > 99 || m.kop < 0)

{

cout << "Error! Enter correct value.\n";

fail = true;

}

else fail = false;

} while (fail);

return in;

}

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

{

out << m.rub << ",";

if (m.kop < 10)

out << "0";

out << m.kop;

return out;

}

bool Money::operator>(const Money& m)

{

if (rub > m.rub) return true;

if (rub == m.rub)

if (kop > m.kop) return true;

return false;

}

bool Money::operator<(const Money& m) const

{

if (rub != m.rub) return rub < m.rub;

return kop < m.kop;

}

Money& Money::operator+(const Money& m)

{

rub += m.rub;

kop += m.kop;

recount();

return \*this;

}

Money Money::operator+(const Money& m) const

{

Money temp;

temp.rub = rub + m.rub;

temp.kop = kop + m.kop;

temp.recount();

return temp;

}

Money Money::operator/(int value)

{

int m = rub \* 100 + kop;

m /= value;

rub = 0;

kop = m;

recount();

return \*this;

}

Vector.h:

#pragma once

#include <iostream>

#include <stack>

#include <vector>

#include "C:\Users\emnos\Desktop\VS2022\LAB11OOP\lab11OOP\_task2\money.h"

using namespace std;

template<class T>

class Vector

{

stack <T> s;

int len;

public:

Vector();

Vector(int n);

Vector(const Vector<T>&);

void print\_stack();

void push\_max();

void remove\_elem();

void add\_mean\_all();

};

template <class T>

vector<T> copy\_stack\_to\_vector(stack<T> s)

{

vector<T> v;

while (!s.empty())

{

v.push\_back(s.top());

s.pop();

}

return v;

}

template <class T>

stack<T> copy\_vector\_to\_stack(vector<T> v)

{

stack<T> s;

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

{

s.push(v[i]);

}

return s;

}

template <class T>

Vector<T>::Vector()

{

len = 0;

}

template <class T>

Vector<T>::Vector(int n)

{

T a;

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

{

cin >> a;

s.push(a);

}

len = s.size();

}

template <class T>

Vector<T>::Vector(const Vector<T>& Vec)

{

len = Vec.len;

vector<T> v = copy\_stack\_to\_vector(Vec.s);

s = copy\_vector\_to\_stack(v);

}

template <class T>

void Vector<T>::print\_stack()

{

vector<T> v = copy\_stack\_to\_vector(s);

while (!s.empty())

{

cout << s.top() << " ";

s.pop();

}

cout << endl;

s = copy\_vector\_to\_stack(v);

}

template <class T>

void Vector<T>::push\_max()

{

vector<T> v = copy\_stack\_to\_vector(s);

Money max = s.top();

s.pop();

while (!s.empty())

{

if (s.top() > max) max = s.top();

s.pop();

}

s = copy\_vector\_to\_stack(v);

s.push(max);

}

template <class T>

void Vector<T>::remove\_elem()

{

Money value;

cout << "Enter the value to remove : \n"; cin >> value;

vector<T> v;

while (!s.empty())

{

if (s.top() != value) v.push\_back(s.top());

s.pop();

}

s = copy\_vector\_to\_stack(v);

}

template <class T>

void Vector<T>::add\_mean\_all()

{

vector<T> v = copy\_stack\_to\_vector(s);

Money sum;

while (!s.empty())

{

sum = sum + s.top();

s.pop();

}

cout << sum << "\n";

Money mean = sum / v.size();

cout << mean << "\n";

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

{

v[i] = v[i] + mean;

}

s = copy\_vector\_to\_stack(v);

}

Lab11\_task5:

#include "vector.h"

int main()

{

Vector<Money>v(3);

cout << "Stack: "; v.print\_stack();

v.push\_max();

cout << "push\_max: "; v.print\_stack();

v.remove\_elem();

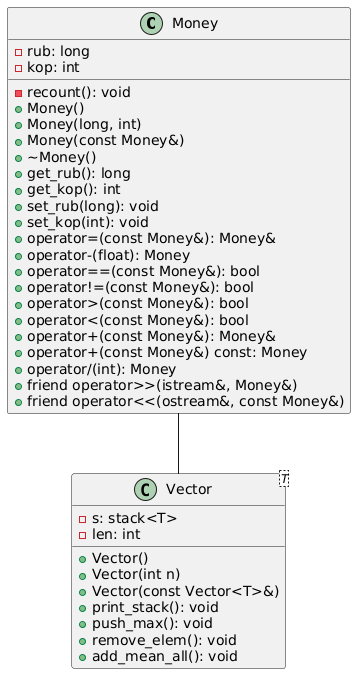
cout << "remove\_elem: "; v.print\_stack();

v.add\_mean\_all();

cout << "add\_mean\_all: "; v.print\_stack();

}

**UML-диаграмма классов**

****

**Контрольные вопросы**

1. Из каких частей состоит библиотека STL?  
**Ответ:** Библиотека STL состоит из трёх основных частей:  
- **Контейнеры** (например, vector, list, deque, set, map)  
- **Адаптеры контейнеров** (например, stack, queue, priority\_queue)  
- **Итераторы** (обеспечивают доступ к элементам контейнеров)  
- **Алгоритмы** (например, сортировка, поиск, перестановка и т. д.)  
- **Функции-объекты** (функции, используемые как объекты)  
  
2. Какие типы контейнеров существуют в STL?  
В STL существуют основные типы контейнеров:  
- **Последовательные контейнеры:** vector, list, deque, array  
- **Ассоциативные контейнеры:** set, multiset, map, multimap  
- **Контейнеры хеширования:** unordered\_set, unordered\_map, unordered\_multiset, unordered\_multimap  
- **Контейнеры адаптеры:** stack, queue, priority\_queue  
  
3. Что нужно сделать для использования контейнера STL в своей программе?  
Для использования контейнера STL необходимо:  
- Подключить соответствующий заголовочный файл, например: #include <vector>, #include <list> и т. д.  
- Объявить контейнер, например: std::vector<int> v;  
- Использовать его методы для добавления, удаления и обработки элементов.  
  
4. Что представляет собой итератор?  
Итератор — это объект, который обеспечивает доступ к элементам контейнера и позволяет перемещаться по нему. Он похож на указатель и поддерживает операции разыменовывания, инкремента и сравнения.  
  
5. Какие операции можно выполнять над итераторами?  
Основные операции:  
- Разыменование (\*it)  
- Инкремент (it++, ++it)  
- Декремент (it--, --it)  
- Сравнение (it1 == it2, it1 != it2)  
  
6. Каким образом можно организовать цикл для перебора контейнера с использованием итератора?  
Обычно используют цикл for или while:  
for (auto it = container.begin(); it != container.end(); ++it) {  
// действия с \*it  
}  
  
7. Какие типы итераторов существуют?  
- **Input iterator** — только для чтения  
- **Output iterator** — только для записи  
- **Forward iterator** — односторонний, для прохода вперёд  
- **Bidirectional iterator** — двунаправленный  
- **Random access iterator** — произвольный доступ (например, у vector)  
  
8. Перечислите операции и методы, общие для всех контейнеров.  
Общие методы:  
- begin(), end()  
- size()  
- empty()  
- clear()  
- insert()  
- erase()  
- swap()  
- find() (для ассоциативных контейнеров)  
  
9. Какие операции являются эффективными для контейнера vector? Почему?  
Эффективные операции:  
- Добавление в конец (push\_back())  
- Доступ по индексу (operator[])  
- Размер (size())  
Потому что vector реализован как динамический массив, что обеспечивает быстрый доступ и добавление в конец.  
  
10. Какие операции являются эффективными для контейнера list? Почему?  
Эффективные операции:  
- Вставка и удаление элементов в любой позиции (insert(), erase())  
Потому что list реализован как двусвязный список, что позволяет быстро вставлять и удалять элементы в любой точке.  
  
11. Какие операции являются эффективными для контейнера deque? Почему?  
Эффективные операции:  
- Добавление и удаление элементов в начале и конце (push\_front(), push\_back(), pop\_front(), pop\_back())  
Потому что deque реализован как двусторонняя очередь с динамической памятью.  
  
12. Перечислите методы, которые поддерживают последовательный контейнер vector.  
- push\_back()  
- pop\_back()  
- insert()  
- erase()  
- clear()  
- resize()  
- swap()  
  
13. Перечислите методы, которые поддерживают последовательный контейнер list.  
- push\_front()  
- push\_back()  
- pop\_front()  
- pop\_back()  
- insert()  
- erase()  
- clear()  
  
14. Перечислите методы, которые поддерживают последовательный контейнер deque.  
- push\_front()  
- push\_back()  
- pop\_front()  
- pop\_back()  
- insert()  
- erase()  
  
15. Задан контейнер vector. Как удалить из него элементы со 2 по 5?  
Можно использовать метод erase():  
auto it\_start = vec.begin() + 1; // индекс 2  
auto it\_end = vec.begin() + 5; // индекс 6 (не включительно)  
vec.erase(it\_start, it\_end);

16. Задан контейнер vector. Как удалить из него последний элемент?  
vec.pop\_back();

17. Задан контейнер list. Как удалить из него элемент с заданным номером?  
Нужно пройти к нужной позиции и вызвать erase():  
auto it = list.begin();  
std::advance(it, index); // index — номер элемента  
list.erase(it);  
  
18. Задан контейнер list. Как удалить из него последний элемент?  
list.pop\_back();  
  
19. Задан контейнер deque. Как удалить из него элемент с заданным номером?  
Аналогично списку:  
auto it = deque.begin();  
std::advance(it, index);  
deque.erase(it);  
  
20. Задан контейнер deque. Как удалить из него последний элемент?  
deque.pop\_back();  
  
21. Написать функцию для печати последовательного контейнера с использованием итератора.  
Пример функции для vector:   
template <typename Container>  
void printContainer(const Container& c) {  
for (auto it = c.begin(); it != c.end(); ++it) {  
std::cout << \*it << ' ';  
}  
std::cout << std::endl;  
}  
  
22. Что представляют собой адаптеры контейнеров?  
Адаптеры контейнеров — это обертки над существующими контейнерами, предоставляющие ограниченный интерфейс:  
- **stack** (стек)  
- **queue** (очередь)  
- **priority\_queue** (очередь с приоритетом)

23. Чем отличаются stack<int> и stack<int, list<int>>?  
Отличие в реализации — по умолчанию stack использует deque как контейнер-основу, а можно указать другой контейнер, например, list.  
  
24. Перечислите методы, которые поддерживает контейнер stack.  
- push()  
- pop()  
- top()  
- empty()  
- size()  
  
25. Перечислите методы, которые поддерживает контейнер queue.  
- push()  
- pop()  
- front()  
- back()  
- empty()  
- size()  
  
26. Чем отличаются друг от друга контейнеры queue и priority\_queue?  
- **queue:** реализует очередь FIFO (первым пришёл — первым обслужен)  
- **priority\_queue:** реализует очередь с приоритетом, где элемент с наивысшим приоритетом извлекается первым  
  
27. Задан контейнер stack. Как удалить из него элемент с заданным номером?  
В стеке нет прямого доступа по индексу, поэтому нужно:  
- скопировать стек в временную структуру  
- удалить нужный элемент  
- восстановить стек  
  
28. Задан контейнер queue. Как удалить из него элемент с заданным номером?  
Аналогично, так как очередь не поддерживает случайный доступ, нужно:  
- извлечь все элементы до нужного  
- удалить его  
- вернуть остальные обратно в очередь  
  
29. Написать функцию для печати контейнера stack с использованием итератора.  
В стандартной библиотеке stack не поддерживает итераторы, поэтому нужно использовать вспомогательный контейнер или копировать содержимое:   
template <typename T>  
void printStack(std::stack<T> s) {  
while (!s.empty()) {  
std::cout << s.top() << ' ';  
s.pop();  
}  
std::cout << std::endl;  
}  
  
30. Написать функцию для печати контейнера queue с использованием итератора.  
Аналогично, у queue нет итераторов, поэтому можно:  
template <typename T>  
void printQueue(std::queue<T> q) {  
while (!q.empty()) {  
std::cout << q.front() << ' ';  
q.pop();  
}  
std::cout << std::endl;  
}

**Ссылка на репозиторий с отчетом Github**

****