МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Использование библиотеки STL

Лабораторная работа №5

по дисциплине «Объекно-ориентированное программирование»

Выполнил студент Чекулаев В. Ю.

Факультет, группа ФКФН, ПО(аб)-81

Проверил Федосеев А. А.

Хабаровск – 2020г.

1. **Постановка задачи**

Написать и отладить три программы. Первая программа демонстрирует использование контейнерных классов для хранения встроенных типов данных. Вторая программа демонстрирует использование контейнерных классов для хранения пользовательских типов данных. Третья программа демонстрирует использование алгоритмов STL.

1. **Определение пользовательского класса**

*#ifndef CLASSES\_H*

#define CLASSES\_H

#include<iostream>

#include<string>

class PrintDevise{

public:

PrintDevise(){}

virtual ~*PrintDevise*(){}

virtual void *print*() const = 0;

virtual std::string *get\_info*() const = 0;

};

class Fax : public PrintDevise{

public:

Fax(std::string name = "Fax", int id = 0) : PrintDevise(), m\_name(name), m\_id(id){}

~*Fax*(){}

void *print*() const override { std::cout << m\_name << " is printing\n"; }

std::string *get\_info*() const override { return m\_name; }

friend std::ostream& operator<<(std::ostream& out, const Fax& elem);

void rename(std::string name){ m\_name = name; }

void send\_data(){ std::cout << m\_name << " is sending data\n"; }

void get\_data(){ std::cout << m\_name << " is getting data\n"; }

int get\_id() const{ return m\_id; }

bool operator<(const Fax& other) const{return m\_id < other.get\_id();}

private:

std::string m\_name;

int m\_id;

};

std::ostream& operator<<(std::ostream& out, const Fax& elem){

out << "Name: " << elem.*get\_info*() << " id:" << elem.get\_id();

return out;

}

#endif *//* *CLASSES\_H*

1. **Листинг программы 1**

#include<iostream>

#include<vector>

#include<string>

void reading(const std::vector<std::string>& v){

for(size\_t i = 0; i < v.size(); ++i){

std::cout << i+1 << ". " << v[i].data() << std::endl;

}

std::cout << std::endl;

}

void reading\_iter(const std::vector<std::string>& v){

std::vector<std::string>::const\_iterator citer(v.cbegin());

size\_t counter(1);

while(citer != v.cend()){

std::cout << counter << ". " << citer->data() << std::endl;

++counter;

++citer;

}

std::cout << std::endl;

}

int main()

{

*//создание* *и* *инициализация*

std::vector<std::string> myvect;

for(unsigned short i = 65; i <= 120; i+=5){

std::string tempstr;

for(char j = i; j < i+5; ++j){

tempstr+=j;

}

myvect.push\_back(tempstr);

}

*//чтение*

reading(myvect);

*//перезапись*

std::vector<std::string>::iterator iter(myvect.begin());

int counter(1);

while(iter != myvect.end()){

if(counter%2){

myvect.erase(iter);

}else{

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

iter->insert(2, "\*");

}

++counter;

++iter;

}

*//чтение* *с* *использованием* *итератеров*

reading\_iter(myvect);

*//создание* *копии* *объекта*

auto myvect\_cpy(myvect);

bool flag(true);

while(flag)

try{

std::cout << "Введите индекс элемента начала удаления: ";

size\_t start;

std::cin >> start;

if(start == 0 || start >= myvect.size()) throw 1;

std::cout << "Введите количество удаляемых элементов: ";

size\_t n;

std::cin >> n;

if(start+n >= myvect.size()) throw 2;

iter = myvect.begin();

myvect.erase(iter+start, iter+start+n);

myvect.insert(iter+start, myvect\_cpy.begin(), myvect\_cpy.end());

flag = false;

}

catch(int id){

switch (id){

case 1:

std::cout << "Индекс элемента начала удаления выходит за пределы размера вектора. Попробуйте снова.\n\n";

break;

case 2:

std::cout << "Такое количество элементов не может быть удалено. Попробуйте снова.\n\n";

break;

default:

break;

}

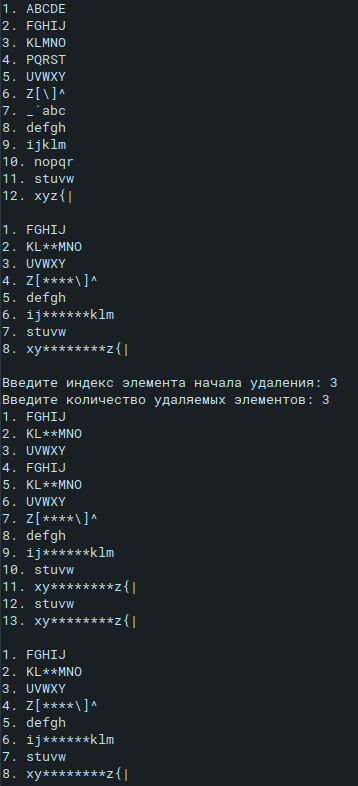
}

reading\_iter(myvect);

reading\_iter(myvect\_cpy);

}

1. **Пример работы программы 1**



**5. Листинг программы 2**

#include<iostream>

#include<vector>

#include<string>

#include"classes.h"

void reading(const std::vector<Fax>& v){

for(size\_t i = 0; i < v.size(); ++i){

std::cout << i+1 << ". " << v[i] << std::endl;

}

std::cout << std::endl;

}

void reading\_iter(const std::vector<Fax>& v){

std::vector<Fax>::const\_iterator citer(v.cbegin());

size\_t counter(1);

while(citer != v.cend()){

std::cout << counter << ". " << \*citer << std::endl;

++counter;

++citer;

}

std::cout << std::endl;

}

int main()

{

*//создание* *и* *инициализация*

std::vector<Fax> myvect;

for(unsigned short i = 65; i <= 120; i+=5){

std::string tempstr;

for(char j = i; j < i+5; ++j){

tempstr+=j;

}

myvect.push\_back(Fax(tempstr));

}

*//чтение*

reading(myvect);

*//перезапись*

std::vector<Fax>::iterator iter(myvect.begin());

int counter(1);

while(iter != myvect.end()){

if(counter%2){

myvect.erase(iter);

}else{

iter->rename("new name" + std::to\_string(counter));

}

++counter;

++iter;

}

*//чтение* *с* *использованием* *итератеров*

reading\_iter(myvect);

*//создание* *копии* *объекта*

auto myvect\_cpy(myvect);

bool flag(true);

while(flag)

try{

std::cout << "Введите индекс элемента начала удаления: ";

size\_t start;

std::cin >> start;

if(start == 0 || start >= myvect.size()) throw 1;

std::cout << "Введите количество удаляемых элементов: ";

size\_t n;

std::cin >> n;

if(start+n >= myvect.size()) throw 2;

iter = myvect.begin();

myvect.erase(iter+start, iter+start+n);

myvect.insert(iter+start, myvect\_cpy.begin(), myvect\_cpy.end());

flag = false;

}

catch(int id){

switch (id){

case 1:

std::cout << "Индекс элемента начала удаления выходит за пределы размера вектора. Попробуйте снова.\n\n";

break;

case 2:

std::cout << "Такое количество элементов не может быть удалено. Попробуйте снова.\n\n";

break;

default:

break;

}

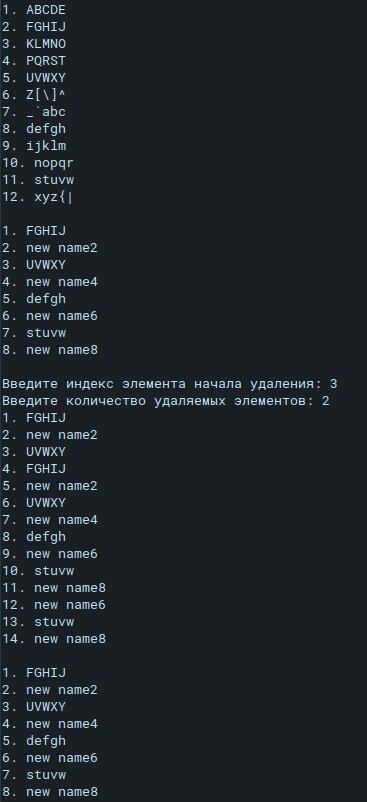
}

reading\_iter(myvect);

reading\_iter(myvect\_cpy);

}

**6. Пример работы программы 2**



**7. Листинг программы 3**

**#include<iostream>**

#include<vector>

#include<stack>

#include<list>

#include<string>

#include<algorithm>

#include<ctime>

#include<functional>

#include<stdlib.h>

#include"classes.h"

template <class T>

void reading\_iter(const T& v){

typename T::const\_iterator citer(v.cbegin());

size\_t counter(1);

while(citer != v.cend()){

std::cout << counter << ". " << \*citer << std::endl;

++counter;

++citer;

}

std::cout << std::endl;

}

bool comp(const Fax& first, const Fax& second){

return !(first < second);

}

void reading\_stack(const std::stack<Fax>& st){

if(st.empty()){

std::cout << "Stack is empty!/n";

return;

}

auto copy(st);

size\_t counter(1);

while(!copy.empty()){

std::cout << counter << ". " << copy.top() << std::endl;

copy.pop();

++counter;

}

std::cout << std::endl;

}

void copy\_if(std::vector<Fax>& vect, std::stack<Fax>& st, const Fax& elem, std::function<bool(const Fax&,const Fax&)> pred){

std::vector<Fax>::const\_iterator iter(vect.cbegin());

while(iter != vect.cend()){

if(pred(\*iter, elem)){

st.push(\*iter);

}

++iter;

}

}

void sort\_stack(std::stack<Fax>& st, std::function<bool(const Fax&,const Fax&)> comparator){

if(st.empty()){

return;

}

std::vector<Fax> vect;

while(!st.empty()){

vect.push\_back(st.top());

st.pop();

}

std::sort(vect.begin(), vect.end(), comparator);

std::vector<Fax>::iterator iter(vect.end()-1);

while(iter != vect.begin()-1){

st.push(\*iter);

--iter;

}

}

void merge(const std::vector<Fax>& vect, const std::stack<Fax>& st, std::list<Fax>& lst){

auto st\_cpy(st);

std::vector<Fax> temp\_v;

while(!st\_cpy.empty()){

temp\_v.push\_back(st\_cpy.top());

st\_cpy.pop();

}

std::merge(vect.begin(), vect.end(), temp\_v.begin(), temp\_v.end(), std::back\_inserter(*lst*));

}

int main()

{

time\_t timer;

std::srand(std::time(&timer));

*//создание* *и* *инициализация*

std::vector<Fax> myvect;

for(unsigned short i = 65; i <= 120; i+=5){

std::string tempstr;

for(char j = i; j < i+5; ++j){

tempstr+=j;

}

int id(std::rand()%10000);

if(id >= 9000){

id = 1234;

}

myvect.push\_back(Fax(tempstr, id));

}

*//чтение*

std::cout << "Vector, not sorted:\n";

reading\_iter(myvect);

*//сортировка*

std::sort(myvect.begin(), myvect.end(), comp);

std::cout << "Vector, sorted descending:\n";

reading\_iter(myvect);

*//поиск*

auto cmp\_asc(

[](const Fax& first,const Fax& second){

return first < second;

});

std::sort(myvect.begin(), myvect.end(), cmp\_asc);

std::cout << "Finding fax with id = 1234 in vector: ";

Fax fax("Fax", 1234);

if(std::binary\_search(myvect.begin(), myvect.end(), fax)){

std::cout << "Found!\n";

} else{

std::cout << "Not found!\n";

}

std::stack<Fax> mystack;

std::cout << "Finding faxes with id > 5000 and copying them into stack\nStack:\n";

copy\_if(*myvect*, *mystack*, Fax("Fax1", 5000), comp);

reading\_stack(mystack);

std::cout << "Stack, sorted ascending\n";

sort\_stack(*mystack*, cmp\_asc);

reading\_stack(mystack);

std::cout << "Vector, sorted ascending\n";

reading\_iter(myvect);

std::list<Fax> mylist;

merge(myvect, mystack, *mylist*);

std::cout << "List, merged from stack and vector\n";

reading\_iter(mylist);

std::cout << "Faxes with id devided by 5 in list: " << std::count\_if(mylist.begin(), mylist.end(), [](const Fax& elem){return elem.get\_id() % 5 == 0;}) << "\n";

std::cout << "List had fax with id < 1000: ";

if(std::count\_if(mylist.begin(), mylist.end(), [](const Fax& elem){return elem.get\_id() < 1000;})){

std::cout << "YES\n";

} else{

std::cout << "NO\n";

}

}

**8. Пример работы программы 3**

