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.

2. 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
```

3. 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 итераторов
    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);

}
```

4. Пример работы программы 1

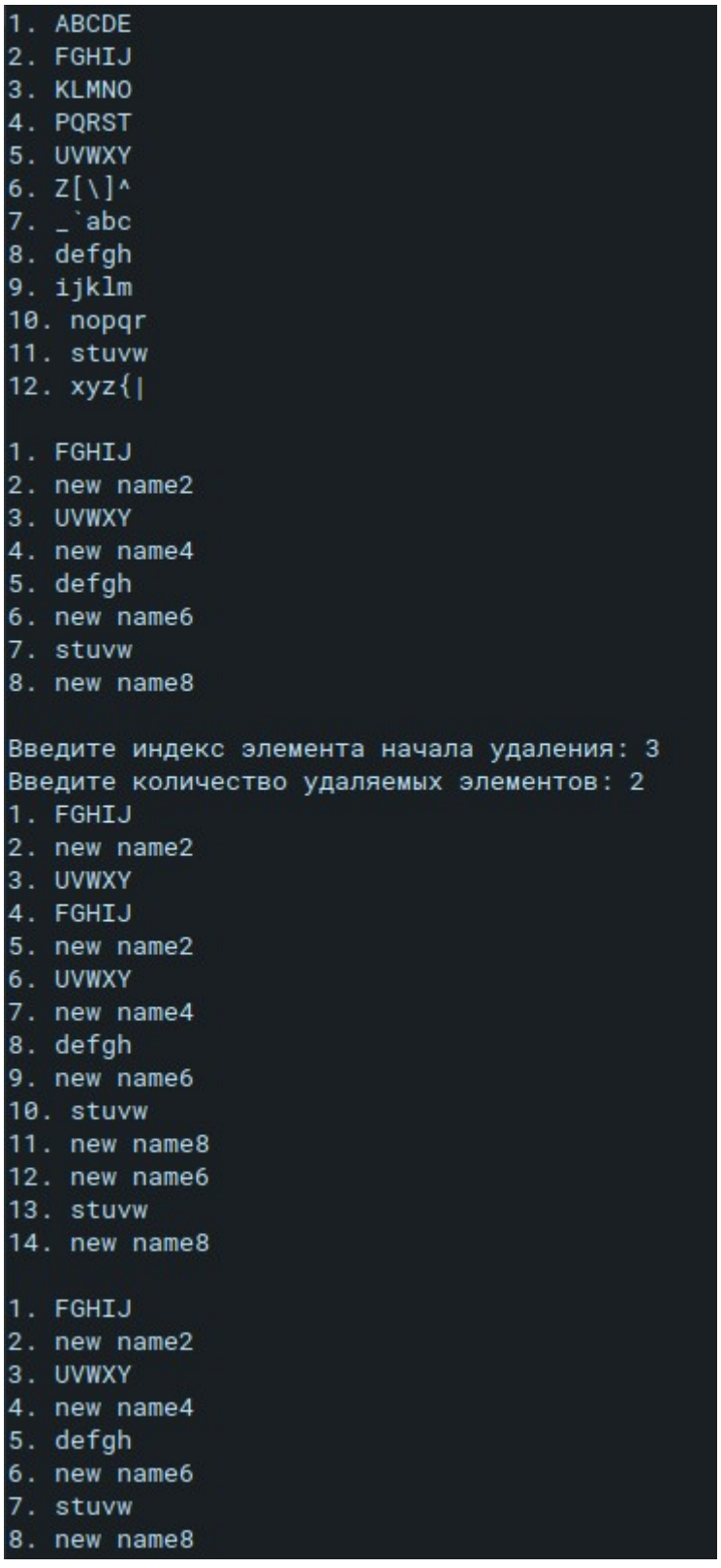
```
1. ABCDE
2. FGHIJ
3. KLMNO
4. PQRST
5. UVWXY
6. Z[\]^
7. _`abc
8. defgh
9. ijklm
10. nopqr
11. stuvw
12. xyz{|

1. FGHIJ
2. KL**MNO
3. UVWXY
4. Z[****\]^
5. defgh
6. ij******klm
7. stuvw
8. xy********z{|

Введите индекс элемента начала удаления: 3
Введите количество удаляемых элементов: 3
1. FGHIJ
2. KL**MNO
3. UVWXY
4. FGHIJ
5. KL**MNO
6. UVWXY
7. Z[****\]^
8. defgh
9. ij******klm
10. stuvw
11. xy********z{|
12. stuvw
13. xy********z{|

1. FGHIJ
2. KL**MNO
3. UVWXY
4. Z[****\]^
5. defgh
6. ij******klm
7. stuvw
8. xy********z{|
```

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 итераторов
    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

```
Vector, not sorted:
1. Name: ABCDE id:1234
2. Name: FGHIJ id:1234
3. Name: KLMNO id:1476
4. Name: PQRST id:5351
5. Name: UVWXY id:3660
6. Name: Z[\]^ id:13
7. Name: _`abc id:1223
8. Name: defgh id:5039
9. Name: ijklm id:8161
10. Name: nopqr id:5872
11. Name: stuvw id:6385
12. Name: xyz{| id:7440

Vector, sorted descending:
1. Name: ijklm id:8161
2. Name: xyz{| id:7440
3. Name: stuvw id:6385
4. Name: nopqr id:5872
5. Name: PQRST id:5351
6. Name: defgh id:5039
7. Name: UVWXY id:3660
8. Name: KLMNO id:1476
9. Name: FGHIJ id:1234
10. Name: ABCDE id:1234
11. Name: _`abc id:1223
12. Name: Z[\]^ id:13

Finding fax with id = 1234 in vector: Found!
Finding faxes with id > 5000 and copying them into stack
Stack:
1. Name: ijklm id:8161
2. Name: xyz{| id:7440
3. Name: stuvw id:6385
4. Name: nopqr id:5872
5. Name: PQRST id:5351
6. Name: defgh id:5039

Stack, sorted ascending
1. Name: defgh id:5039
2. Name: PQRST id:5351
3. Name: nopqr id:5872
4. Name: stuvw id:6385
5. Name: xyz{| id:7440
6. Name: ijklm id:8161
```

```
Vector, sorted ascending
1. Name: Z[\]^ id:13
2. Name: _`abc id:1223
3. Name: FGHIJ id:1234
4. Name: ABCDE id:1234
5. Name: KLMNO id:1476
6. Name: UVWXY id:3660
7. Name: defgh id:5039
8. Name: PQRST id:5351
9. Name: nopqr id:5872
10. Name: stuvw id:6385
11. Name: xyz{| id:7440
12. Name: ijklm id:8161

List, merged from stack and vector
1. Name: Z[\]^ id:13
2. Name: _`abc id:1223
3. Name: FGHIJ id:1234
4. Name: ABCDE id:1234
5. Name: KLMNO id:1476
6. Name: UVWXY id:3660
7. Name: defgh id:5039
8. Name: defgh id:5039
9. Name: PQRST id:5351
10. Name: PQRST id:5351
11. Name: nopqr id:5872
12. Name: nopqr id:5872
13. Name: stuvw id:6385
14. Name: stuvw id:6385
15. Name: xyz{| id:7440
16. Name: xyz{| id:7440
17. Name: ijklm id:8161
18. Name: ijklm id:8161

Faxes with id devided by 5 in list: 5
List had fax with id < 1000: YES
```