

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

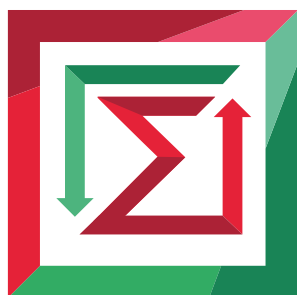


**НГТУ  
НЭТИ**

Кафедра теоретической и прикладной информатики

Лабораторная работа № 2  
по дисциплине «Объектно-ориентированные технологии разработки  
программного обеспечения»

### **Последовательные контейнеры STL**



ФАКУЛЬТЕТ:	ПМИ
Группа:	ПМИМ-01
СТУДЕНТЫ:	Ершов П.К., Малышкина Е.Д. Слободчикова А.Э.
ВАРИАНТ:	4
ПРЕПОДАВАТЕЛЬ:	Лисицин Д. В.

Новосибирск

2021

## **1. Цель работы**

Изучить последовательные контейнеры и способы работы с ними.

## **2. ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

1. Изучить элементы стандартной библиотеки шаблонов C++: последовательные контейнеры, адаптеры контейнеров, функциональные объекты, функциональные адаптеры, лямбда-выражения.
2. Переработать программу из работы №1 так, чтобы объекты разработанного класса хранились в одном из следующих контейнеров STL: список (list), односвязный список (forward\_list). Использовать функции-члены контейнера для реализации следующих возможностей:
  - 2.1. Добавление объектов в контейнер.
  - 2.2. Поиск объектов в контейнере по различным критериям.
  - 2.3. Модификация объектов в контейнере (возможно, в соответствии с некоторыми критериями).
  - 2.4. Удаление объектов из контейнера по различным критериям.
  - 2.5. Вывод всех объектов контейнера, отсортированных по различным критериям, в файл и/или на экран.
3. При формировании критериев поиска, удаления, сортировки использовать функциональные объекты (пользовательские и/или стандартные) и/или лямбда-выражения. Атрибуты класса, хранящиеся как строки, но по смыслу являющиеся числами, должны участвовать в сравнениях как числа.
4. Оформить отчет. Отчет должен содержать постановку задачи, алгоритм, описание и текст разработанной программы, результаты тестирования (со скриншотами) и выводы.
5. Защитить работу, ответив на вопросы преподавателя.

## **3. Вариант задания**

Класс с именем WORKER, содержащий следующие атрибуты:

- фамилия работника;
- имя работника;
- отчество работника;
- название занимаемой должности;
- год поступления на работу.

Находить работников, которые имеют

- стаж работы в организации, превышающий заданное значение;
- заданную должность

#### 4. Описание алгоритма

`Worker()`; - конструктор экземпляра класса по умолчанию

`Worker(string s_name, string name, string patronymic, string function, int day, int month, int year)`; - конструктор экземпляра класса с заданными параметрами

`int checkDate(int day, int month, int year)`; - функция проверки формата даты

`void set_Date(int day, int month, int year)`; - функция установки даты

`void setS_name(string s_name)`; - функция установки фамилии

`void setName(string name)`; - функция установки имени

`void setPatr(string patronymic)`; - функция установки отчества

`void setFunc(string function)`; - функция установки должности

`template <typename charT, typename traits> friend basic_ostream<charT, traits>& operator << (basic_ostream<charT, traits>& out, const Worker& c)`; - перегрузка функции потока вывода

`friend istream& operator>>(istream& in, Worker& c)`; - перегрузка функции потока ввода

`string getS_name()` - функция получения фамилии

`string getName()` - функция получения имени

`string getPatr()` - функция получения отчества

`string getFunc()` - функция получения должности

`int getDay()` - функция получения дня

`int getMonth()` - функция получения месяца

`int getYear()` - функция получения года

`void file_save(vector<Worker> workers, string filename)` - функция получения записей из файла

`void output_all(vector<Worker> workers, int format)` - функция сохранения записей в файл

`vector<Worker> search(vector<Worker> workers, string function)` – функция поиска рабочих по должности

`vector<Worker> search(vector<Worker> workers, int year)` – функция поиска рабочих по стажу

`void mod(vector<Worker> &workers, Worker buf, int num)` – функция модификации записи рабочего

## 5. Примеры работы программы

### Рабочее меню программы

```
Выберете функцию:
1. Вывод списка рабочих.
2. Добавление нового рабочего.
3. Добавление рабочих из файла.
4. Удаление рабочих.
5. Поиск рабочих, чей стаж превышает заданное значение.
6. Поиск рабочих по конкретной должности.
7. Сохранение списка рабочих в указанный файл.
8. Модификация указанной записи рабочего.
9. Выход.
```

**Результат попытки вывести список рабочих, когда он ещё не загружен из файла (так же можно выбрать формат вывода, реализуемый пользовательскими флагами формата)**

```
1
Выберите формат вывода:
1. Стандартный в строку.
2. В формате таблицы.
1
Список рабочих пуст
```

### Добавление новой записи в список

```
2
Введите данные нового рабочего в формате: Фамилия Имя Отчество Должность (0.0.0000)
Romanov Nicolay Alecsandrovich tester (1.12.1998)
```

**Демонстрация, что запись уже в списке**

```
1
Выберите формат вывода:
1. Стандартный в строку.
2. В формате таблицы.
1
Romanov Nicolay Alecsandrovich tester (1.12.1998)
```

## Вывод списка рабочих после получения списка из файла (вывод форматированный)

```
1
Выберите формат вывода:
1. Стандартный в строку.
2. В формате таблицы.
2
0. Second name: Romanov
Name: Nicolay
Patronymic: Alecsandrovich
Position: tester
Beginning of work: 1.12.1998

1. Second name: Eliseev
Name: Alexander
Patronymic: Alexandrovich
Position: accountant
Beginning of work: 20.11.1998

2. Second name: Firisov
Name: Boris
Patronymic: Nikolaevich
Position: administrator
Beginning of work: 3.2.2005

3. Second name: Turok
Name: Ivan
Patronymic: Danilovich
Position: administrator
Beginning of work: 15.5.1990

4. Second name: Kirenko
Name: Yulia
Patronymic: Viktorovna
Position: programmer
Beginning of work: 2.12.2003

5. Second name: Yuriy
Name: Tavish
Patronymic: Romanovich
Position: programmer
Beginning of work: 3.11.1992

6. Second name: Arionova
Name: Mariya
Patronymic: Aleksandrovna
Position: tester
Beginning of work: 27.2.1995
```

## Демонстрация удаления выбранной записи

```
4
0. Eliseev Alexander Alexandrovich accountant (20.11.1998)
1. Firisov Boris Nikolaevich administrator (3.2.2005)
2. Turok Ivan Danilovich administrator (15.5.1990)
3. Kirenko Yulia Viktorovna programmer (2.12.2003)
4. Yuriy Tavish Romanovich programmer (3.11.1992)
5. Arionova Mariya Aleksandrovna tester (27.2.1995)
Выберите номер рабочего, чья запись будет удалена:
0
0. Firisov Boris Nikolaevich administrator (3.2.2005)
1. Turok Ivan Danilovich administrator (15.5.1990)
2. Kirenko Yulia Viktorovna programmer (2.12.2003)
3. Yuriy Tavish Romanovich programmer (3.11.1992)
4. Arionova Mariya Aleksandrovna tester (27.2.1995)
```

## Поиск рабочих, чей стаж превышает заданное число лет

```
5
Введите стаж:
20
0. Turok Ivan Danilovich administrator (15.5.1990)
1. Yuriy Tavish Romanovich programmer (3.11.1992)
2. Arionova Mariya Aleksandrovna tester (27.2.1995)
```

## Поиск сотрудников по их должности

```
Введите должность:
programmer
0. Kirenko Yulia Viktorovna programmer (2.12.2003)
1. Yuriy Tavish Romanovich programmer (3.11.1992)
```

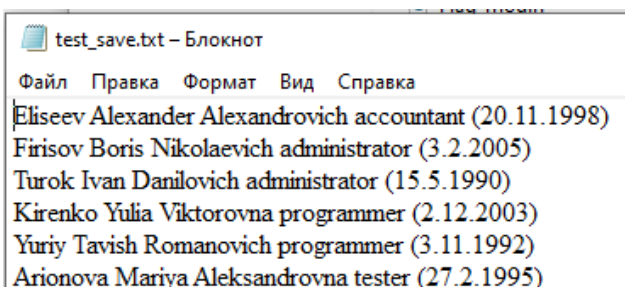
## Демонстрация результатов поиска для случаев, когда не найден ни один сотрудник

```
Введите стаж:
40
Рабочих с указанным стажем не обнаружено
```

```
Введите должность:
GOD
Рабочих с данной должностью не обнаружено
```

## Сохранение списка рабочих в указанный файл

```
7
Введите название файла сохранения
test_save.txt
```



test\_save.txt – Блокнот

Файл Правка Формат Вид Справка

Eliseev Alexander Alexandrovich accountant (20.11.1998)  
Firisov Boris Nikolaevich administrator (3.2.2005)  
Turok Ivan Danilovich administrator (15.5.1990)  
Kirenko Yulia Viktorovna programmer (2.12.2003)  
Yuriy Tavish Romanovich programmer (3.11.1992)  
Arionova Mariya Aleksandrovna tester (27.2.1995)

## Модификация выбранной записи

```
8
0. Eliseev Alexander Alexandrovich accountant (20.11.1998)
1. Firisov Boris Nikolaevich administrator (3.2.2005)
2. Turok Ivan Danilovich administrator (15.5.1990)
3. Kirenko Yulia Viktorovna programmer (2.12.2003)
4. Yuriy Tavish Romanovich programmer (3.11.1992)
5. Arionova Mariya Aleksandrovna tester (27.2.1995)
Выберите номер рабочего, чья запись будет изменена:
0

Eliseev Alexander Alexandrovich accountant (20.11.1998)
TEST TEST TEST TESTER (1.1.2021)

0. TEST TEST TEST TESTER (1.1.2021)
1. Firisov Boris Nikolaevich administrator (3.2.2005)
2. Turok Ivan Danilovich administrator (15.5.1990)
3. Kirenko Yulia Viktorovna programmer (2.12.2003)
4. Yuriy Tavish Romanovich programmer (3.11.1992)
5. Arionova Mariya Aleksandrovna tester (27.2.1995)
```

## Сортировка рабочих

### По фамилии

```
0. Eliseev Alexander Alexandrovich accountant (20.11.1998)
1. Firisov Boris Nikolaevich administrator (3.2.2005)
2. Turok Ivan Danilovich administrator (15.5.1990)
3. Kirenko Yulia Viktorovna programmer (2.12.2003)
4. Yuriy Tavish Romanovich programmer (3.11.1992)
5. Arionova Mariya Aleksandrovna tester (27.2.1995)
Выберите тип сортировки:
1. По фамилии.
2. По стажу в годах.
1

0. Arionova Mariya Aleksandrovna tester (27.2.1995)
1. Eliseev Alexander Alexandrovich accountant (20.11.1998)
2. Firisov Boris Nikolaevich administrator (3.2.2005)
3. Kirenko Yulia Viktorovna programmer (2.12.2003)
4. Turok Ivan Danilovich administrator (15.5.1990)
5. Yuriy Tavish Romanovich programmer (3.11.1992)
```

## По стажу

```
0. Arionova Mariya Aleksandrovna tester (27.2.1995)
1. Eliseev Alexander Alexandrovich accountant (20.11.1998)
2. Firisov Boris Nikolaevich administrator (3.2.2005)
3. Kirenko Yulia Viktorovna programmer (2.12.2003)
4. Turok Ivan Danilovich administrator (15.5.1990)
5. Yuriy Tavish Romanovich programmer (3.11.1992)
```

Выберите тип сортировки:

1. По фамилии.
  2. По стажу в годах.
- 2

```
0. Turok Ivan Danilovich administrator (15.5.1990)
1. Yuriy Tavish Romanovich programmer (3.11.1992)
2. Arionova Mariya Aleksandrovna tester (27.2.1995)
3. Eliseev Alexander Alexandrovich accountant (20.11.1998)
4. Kirenko Yulia Viktorovna programmer (2.12.2003)
5. Firisov Boris Nikolaevich administrator (3.2.2005)
```



## 6. Текст программы

### Файл Worker.h

```
#ifndef WORKER
#define WORKER

#pragma once
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <ctime>

using namespace std;

namespace Workers
{
    const int spec_index = ios_base::xalloc();

    template <typename charT, typename traits> inline basic_ostream <charT, traits>&
        tab_out(basic_ostream<charT, traits>& out)
    {
        out.iword(spec_index) = true;
        return out;
    }

    template <typename charT, typename traits> inline basic_ostream<charT, traits>&
        stand_out(basic_ostream<charT, traits>& out)
    {
        out.iword(spec_index) = false;
        return out;
    }

    class Worker
    {
    private:
        string s_name = "",
            name = "",
            patronymic = "",
            function = "";

        int day,
            month,
            year;

    public:
        Worker();

        Worker(string s_name, string name, string patronymic, string
function, int day, int month, int year);

        int checkDate(int day, int month, int year);

        void set_Date(int day, int month, int year);

        void setS_name(string s_name);
    }
}
```

```

        void setName(string name);

        void setPatr(string patronymic);

        void setFunc(string function);

        template <typename charT, typename traits> friend
basic_ostream<charT, traits>&
        operator << (basic_ostream<charT, traits>& out, const Worker&
c);

        friend istream& operator>>(istream& in, Worker& c);

        string getS_name()
        {
            return this->s_name;
        }

        string getName()
        {
            return this->name;
        }

        string getPatr()
        {
            return this->patronymic;
        }

        string getFunc()
        {
            return this->function;
        }

        int getDay()
        {
            return this->day;
        }

        int getMonth()
        {
            return this->month;
        }

        int getYear()
        {
            return this->year;
        }

};

template <typename charT, typename traits>
inline basic_ostream<charT, traits>& operator << (basic_ostream<charT, traits>&
out, const Worker& c)
{
    basic_ostringstream <charT, traits> s;

    s.copyfmt(out);
    s.width(0);

    if (s.iword(spec_index)) {
        s << "Second name: " << c.s_name << endl;
        s << "Name: " << c.name << endl;
        s << "Patronymic: " << c.patronymic << endl;
        s << "Position: " << c.function << endl;
    }
}

```

```

        s << "Beginning of work: " << c.day << "." << c.month << "." <<
c.year << endl;
    }
    else
        s << c.s_name << " " << c.name << " " << c.patronymic << " " <<
c.function << " (" << c.day << "." << c.month << "." << c.year << ")";
    out << s.str();
    return out;
}

}
#endif

```

## Файл Worker.cpp

```

#define _CRT_SECURE_NO_WARNINGS
#include "Worker.h"

using namespace Workers;

Worker::Worker()
{
    setS_name(string());
    setName(string());
    setPatr(string());
    this->day = 0;
    this->month = 0;
    this->year = 0;
}

Worker::Worker(string s_name, string name, string patronymic, string function, int day,
int month, int year)
{
    setS_name(s_name);
    setName(name);
    setPatr(patronymic);
    setFunc(function);
    set_Date(day, month, year);
}

int Worker::checkDate(int day, int month, int year)
{
    struct tm* date;
    time_t t = time(NULL);
    date = gmtime(&t);

    if (day > 0)
    {
        if ((month == 1 || month == 3 || month == 5 || month == 7 ||
            month == 8 || month == 10 || month == 12) && day > 31)
            return 1;

        if ((month == 4 || month == 6 || month == 9 || month == 11) && day > 30)
            return 1;

        if (month == 2 && year % 4 == 0 && day > 29)
            return 1;

        if (month == 2 && year % 4 != 0 && day > 28)
            return 1;
    }
    else
        return 1;
}

```

```

        if (month < 0 || month > 12)
            return 2;

        if (year < 0 || year > date->tm_year + 1900)
            return 3;

        return 0;
    }

    void Worker::set_Date(int day, int month, int year)
    {
        if (checkDate(day, month, year) == 0)
        {
            this->day = day;
            this->month = month;
            this->year = year;
        }
    }

    void Worker::setS_name(string s_name)
    {
        this->s_name = s_name;
    }

    void Worker::setName(string name)
    {
        this->name = name;
    }

    void Worker::setPatr(string patronymic)
    {
        this->patronymic = patronymic;
    }

    void Worker::setFunc(string function)
    {
        this->function = function;
    }

namespace Workers
{
    istream& operator>>(istream& in, Worker& input)
    {
        string s_name, name, part, func;
        int day, month, year;
        char open, close, point1, point2;
        bool flag = true;

        in >> s_name >> name >> part >> func >> open >> day >> point1 >> month >>
point2 >> year >> close;

        Worker buf("0", "0", "0", "0", day, month, year);
        if (open != '(' || close != ')' || point1 != '.' || point2 != '.')
        {
            flag = false;
        }

        if (flag && buf.checkDate(day, month, year) == 0)
        {
            input.setS_name(s_name);
            input.setName(name);
            input.setPatr(part);

```

```

        input.setFunc(func);
        input.setDate(day, month, year);
    }
    else
    {
        in.clear(ios::failbit);
        return in;
    }

    return in;
}
}

```

## Файл WORKER\_Class.cpp

```

#include <iostream>
#include "Worker.h"
#include <fstream>
#include <vector>
#include <list>

using namespace std;
using namespace Workers;

void file_load(list<Worker>& workers, string filename)
{
    Worker wr;
    ifstream f(filename);

    if (!f.is_open())
    {
        cout << "Error! No file with name " << filename << endl;
        return;
    }

    while (!f.eof())
    {
        f >> wr;
        if (f.fail())
        {
            f.clear(ios::eofbit);
            break;
        }
        workers.push_back(wr);
    }
    f.close();
};

void file_save(list<Worker> workers, string filename)
{
    Worker wr;
    ofstream f(filename);

    if (!f.is_open())
    {
        cout << "Error! No file with name " << filename << endl;
        return;
    }

    for (auto iter = workers.begin(); iter != workers.end(); iter++)
    {
        f << *iter << endl;
    }
}

```

```

    }
    f.close();
};

void output_all(list<Worker> workers, int format)
{
    int i = 0;
    for (auto iter = workers.begin(); iter != workers.end(); iter++)
    {
        if (format == 1)
            cout << stand_out << i << ". " << *iter << endl;
        else
            cout << tab_out << i << ". " << *iter << endl;
        i++;
    }
}

list<Worker> search(list<Worker> workers, string function)
{
    list<Worker> out;
    list<Worker>::iterator iter;
    iter = find_if(workers.begin(),
        workers.end(),
        [&](Worker w)
        {
            return w.getFunc() == function;
        });

    while (iter != workers.end())
    {
        out.push_back(*iter);
        iter++;
        iter = find_if(iter, workers.end(), [&](Worker w)
            {
                return w.getFunc() == function;
            });
    }

    return out;
}

list<Worker> search(list<Worker> workers, int year)
{
    struct tm* date;
    time_t t = time(NULL);
    date = gmtime(&t);
    int tt = date->tm_year + 1900;
    list<Worker> out;
    list<Worker>::iterator iter;
    iter = find_if(workers.begin(),
        workers.end(),
        [&](Worker w)
        {
            return (tt - w.getYear()) > year;
        });

    while (iter != workers.end())
    {
        out.push_back(*iter);
        iter++;
        iter = find_if(workers.begin(),
            workers.end(),
            [&](Worker w)
            {
                return (tt - w.getYear()) > year;
            });
    }
}

```

```

        });
    }

    return out;
};

void sort(list<Worker>& workers, int num)
{
    if (num == 1)
        workers.sort([](Worker& a, Worker& b) { return a.getS_name() < b.getS_name(); });
    if (num == 2)
        workers.sort([](Worker& a, Worker& b) { return a.getYear() < b.getYear(); });
}

void mod(list<Worker>& workers, Worker buf, int num)
{
    auto w_i = workers.begin();
    advance(w_i, num);
    w_i->setS_name(buf.getS_name());
    w_i->setName(buf.getName());
    w_i->setPatr(buf.getPatr());
    w_i->setFunc(buf.getFunc());
    w_i->setDate(buf.getDay(), buf.getMonth(), buf.getYear());
};

int main()
{
    setlocale(LC_ALL, "Russian");

    bool exit = false;
    int option;

    list<Worker> workers;

    while (!exit)
    {
        cout << "Выберете функцию:" << endl;
        cout << "1. Вывод списка рабочих." << endl;
        cout << "2. Добавление нового рабочего." << endl;
        cout << "3. Добавление рабочих из файла." << endl;
        cout << "4. Удаление рабочих." << endl;
        cout << "5. Поиск рабочих, чей стаж превышает заданное значение." << endl;
        cout << "6. Поиск рабочих по конкретной должности." << endl;
        cout << "7. Сохранение списка рабочих в указанный файл." << endl;
        cout << "8. Модификация указанной записи рабочего." << endl;
        cout << "9. Сортировка списка." << endl;
        cout << "10. Выход." << endl;

        cin >> option;

        while (cin.fail())
        {
            cin.clear();
            cout << "Не верное значение!" << endl;
            cin.ignore();
            cin >> option;
        }

        switch (option)
        {
        case 1:
        {
            int format;
            cout << "Выберите формат вывода:" << endl;
            cout << "1. Стандартный в строку." << endl;

```

```

        cout << "2. В формате таблицы." << endl;
        cin >> format;
        if (!workers.empty())
            output_all(workers, format);
        else
            cout << "Список рабочих пуст" << endl;
        cout << endl;
        break;
    }
    case 2:
    {
        Worker buf;
        cout << "Введите данные нового рабочего в формате: Фамилия Имя Отчество  
Должность (0.0.0000)" << endl;
        cin >> buf;
        /*if (cin.fail())
            cout << "Ошибка. Неверный формат ввода." << endl;
        else*/
        workers.push_back(buf);
        cout << endl;
        break;
    }
    case 3:
    {
        file_load(workers, "data.txt");
        cout << endl;
        break;
    }
    case 4:
    {
        if (!workers.empty())
            output_all(workers, 1);
        else
        {
            cout << "Список рабочих пуст" << endl;
            break;
        }

        int num;
        cout << "Выберите номер рабочего, чья запись будет удалена:" << endl;
        cin >> num;
        auto w_i = workers.begin();
        advance(w_i, num);
        workers.erase(w_i);
        cout << endl;
        output_all(workers, 1);
        cout << endl;
        break;
    }
    case 5:
    {
        list<Worker> out;
        int year;
        cout << "Введите стаж:" << endl;
        cin >> year;
        out = search(workers, year);
        if (out.size() != 0)
            output_all(out, 1);
        else
            cout << "Рабочих с указанным стажем не обнаружено" << endl;
        cout << endl;
        break;
    }
    case 6:
    {

```



```

    list<Worker> out;
    string func;
    cout << "Введите должность:" << endl;
    cin >> func;
    out = search(workers, func);
    if(out.size() != 0)
        output_all(out, 1);
    else
        cout << "Рабочих с данной должностью не обнаружено" << endl;
    cout << endl;
    break;
}
case 7:
{
    string n_file;
    cout << "Введите название файла сохранения" << endl;
    cin >> n_file;
    file_save(workers, n_file);
    cout << endl;
    break;
}
case 8:
{
    if (workers.size() != 0)
        output_all(workers, 1);
    else
    {
        cout << "Список рабочих пуст" << endl;
        break;
    }
    Worker buf;
    int num;
    cout << "Выберите номер рабочего, чья запись будет изменена:" << endl;
    cin >> num;
    cout << endl;
    auto w_i = workers.begin();
    advance(w_i, num);
    cout << stand_out << *w_i << endl;
    cin >> buf;
    mod(workers, buf, num);
    cout << endl;
    output_all(workers, 1);
    cout << endl;
    break;
}
case 9:
{
    if (workers.size() != 0)
        output_all(workers, 1);
    else
    {
        cout << "Список рабочих пуст" << endl;
        break;
    }
    int num;
    cout << "Выберите тип сортировки:" << endl;
    cout << "1. По фамилии." << endl;
    cout << "2. По стажу в годах." << endl;
    cin >> num;
    cout << endl;
    sort(workers, num);
    output_all(workers, 1);
    cout << endl;
    break;
}

```

```
        case 10:
        {
            exit = true;
            break;
        }
        default:
            break;
    }
}
```