**Содержание**

[**Задание (2.1)** 1](#_Toc52738338)

[**Задание (2.2)** 1](#_Toc52738339)

[**1. Постановка задачи** 1](#_Toc52738340)

[**2. Разработка программы** 2](#_Toc52738341)

[2.1 Разработка функций и методов классов 2](#_Toc52738342)

[2.2 Разработка интерфейса 5](#_Toc52738343)

[Разрабатывается консольное приложение, которое предоставляет пользователю меню](#_Toc52738344)

[**3. Реализация и тестирование программы** 5](#_Toc52738345)

[3.1 Описание разработанной программы 5](#_Toc52738346)

[3.2 Тестирование программы 6](#_Toc52738347)

[**Вывод** 8](#_Toc52738348)

[**Приложение. Код программы** 9](#_Toc52738349)

**Задание (2.1)**

Построить описание класса, содержащего информацию о почтовом адресе организации. Предусмотреть возможность раздельного изменения составных частей адреса и проверки допустимости вводимых значений. В случае недопустимых значений полей выбрасываются исключения. Написать программу, демонстрирующую все разработанные элементы класса.

**Задание (2.2)**

Создать дочерний класс, который содержит информацию о данной организации – наименование, количество сотрудников и т.п. Предусмотреть возможность раздельного изменения наименования организации и числа сотрудников и проверки допустимости вводимых значений

1. **Постановка задачи**

Разработать объектно-ориентированную программу в соответствии с заданиями 2.1 и 2.2.

**Функции программы:** Ввод и вывод данных об адресе компании (страна, город, улица, номер дома, почтовый индекс, название компании, количество работников) и реализация методов для редактирования этих данных вместе и по отдельности.

**Входные данные:** Адрес, название компании, число её работников:

(*переменные типа string: address, country, city, street, house, index, name\_company, population*)

**Выходные данные:** Адрес, название компании, число её работников:

Адрес (переменная address)

Название страны (переменная country)

Название города (переменная city)

Название улицы (переменная street)

Номер дома (переменная house)

Почтовый индекс (переменная index)

Название компании (переменная name\_company)

Количество работников (переменная population)

**Ограничения:** Названия страны, города и улицы могут состоять только из букв

Почтовый индекс может состоять только из 6 цифр

Количество работников является целым числом

*Вид приложения – консольное приложение на языке C++.*

*Среда разработки – JetBrains CLion 2020.2.4*

1. **Разработка программы**
   1. Разработка функций и методов классов

Для работы программы необходимо создать базовый класс Post, который будет представлять собой данные об адресе компании и методы их редактирования. Также требуется создать дочерний класс CompanyInfo, который будет содержать данные о компании (название, количество работников) и методы их редактирования. Диаграмма классов приведена на *Рис.1* для визуальной связи между классами. Буквой f обозначены методы, а F – поля. В таблицах Табл.1 Конструкторы, Табл. 2 Поля и методы описаны конструкторы, поля и методы.

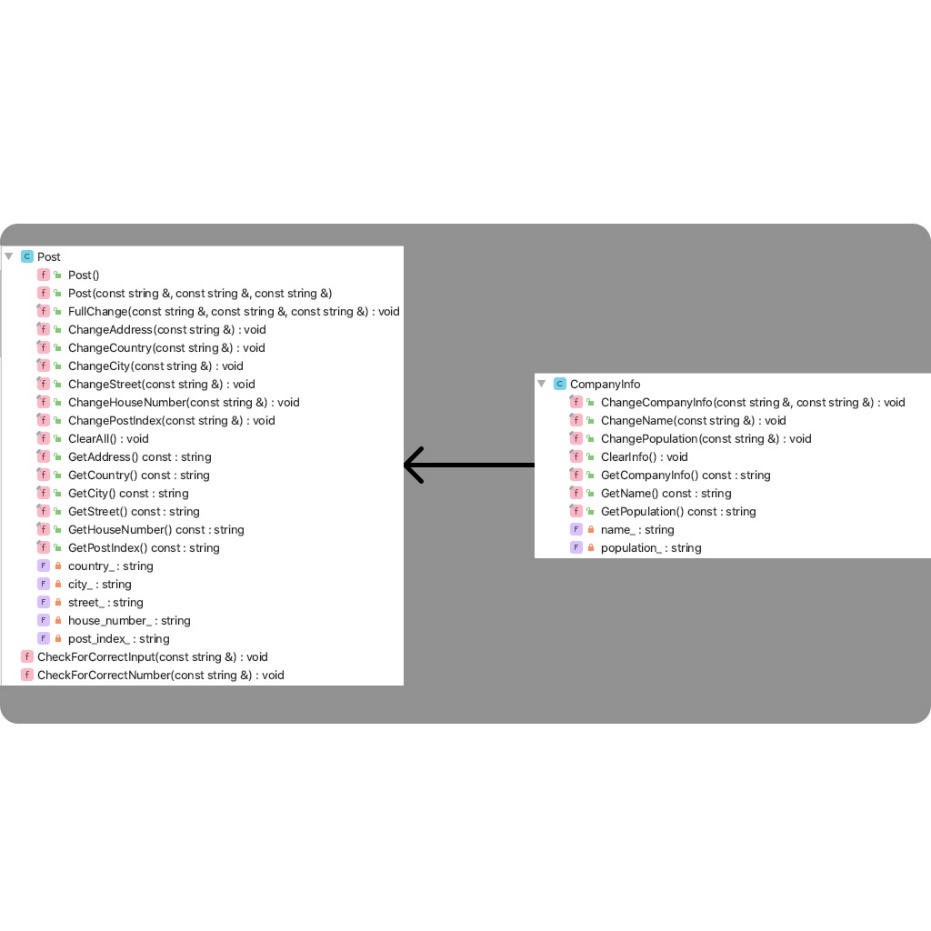


Рис. 1 Диаграмма классов

Табл. 1 Конструкторы

|  |  |  |
| --- | --- | --- |
| Название | Конструкторы | Описание |
| Post  Адрес | Post(); | Конструктор по умолчанию. Все значения пусты. |
| Post(const string & address, const string & name, const string & pop); | Параметризированный конструктор. Принимает на вход параметры комнаты и обрабатывает их. |
| CompanyInfo  Информация о компании (дочерний класс Post) | CompanyInfo(); | Конструктор по умолчанию. Все значения пусты. |

Табл. 2 Поля и методы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Поля | Методы | Входные данные | Выходные  данные |
| Post  (базовый класс) | country\_ - название страны  city\_ - название города  street\_ - название улицы  house\_number\_ - номер дома  post\_index\_ - почтовый индекс | void FullChange(const string & address, const string & name, const string & pop) – устанавливает значения всех полей | address = country+city+street+house+index, name, pop  Адрес(составная переменная), название компании и количество работников | Нет |
| void ChangeAddress(const string & address) – устанавливает значения полей класса Post | address = country+city+street+house+index - Адрес(составная переменная) | Нет |
| void ChangeCountry(const string & country) – устанавливает значение поля country\_ | country – название страны | Нет |
| void ChangeCity(const string & city) – устанавливает значение поля city\_ | city – название города | Нет |
| void ChangeStreet(const string & street) – устанавливает значение поля street\_ | street – название улицы | Нет |
|  |  | void ChangeHouseNumber(const string & number) – устанавливает значение поля house\_number\_ | number – номер дома | Нет |
|  |  | void ChangePostIndex(const string & index) – устанавливает значение поля post\_index\_ | index – почтовый индекс | Нет |
| void ClearAll() – очищает значения всех полей | Нет | Нет |
| string GetAddress() const – возвращает значение адреса (состоит из всех остальных полей) | Нет | address = country\_+city\_+street\_+house\_number\_+post\_index\_, name\_company, population  Адрес(составная переменная), название компании и количество работников |

Табл. 2 Поля и методы продолжение

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | string GetCountry() const – возвращает значение поля country\_ | Нет | country\_ – название страны |
| string GetCity() const – возвращает значение поля city\_ | Нет | city\_ – название города |
| string GetStreet() const – возвращает значение поля street\_ | Нет | street\_ – название улицы |
| string GetHouseNumber() const – возвращает значение поля house\_number\_ | Нет | number\_ – номер дома |
| string GetPostIndex() const – возвращает значение поля post\_index\_ | Нет | index\_ – почтовый индекс |
| CompanyInfo  (наследник класса Post) | name\_ - название компании  population\_ - количество работников | void ChangeCompanyInfo(const string & company, const string & pop) – устанавливает значения полей класса CompanyInfo | company, pop – название компании и количество работников | Нет |
| void ChangeName(const string & name) – устанавливает значение поля name\_ | name – название компании | Нет |
| void ChangePopulation(const string & population) – устанавливает значение поля population\_ | population - количество работников | Нет |
|  |  | void ClearInfo() – очищает значения полей класса CompanyInfo | Нет | Нет |
| string GetCompanyInfo() const – возвращает значение информации о компании (содержит name\_ и population\_) | Нет | company, pop – название компании и количество работников |
| string GetName() const – возвращает значение поля name\_ | Нет | name – название компании |
| string GetPopulation() const – возвращает значение поля population\_ | Нету | population - количество работников |

* 1. Разработка интерфейса

Разрабатывается консольное приложение, которое вначале в форме диалога предлагает ввести полный адрес компании, её название и количество работников (Рис. 2). После того, как пользователь введет полный адрес появится меню выбора действий (Рис. 3), где он может изменить адрес полностью или поэлементно, вывести его на экран или очистить.

|  |  |
| --- | --- |
|  |  |
| Рис. 2 Пример диалога | Рис. 3 Пример меню |

1. **Реализация и тестирование программы**
   1. Описание разработанной программы

Для реализации программы в качестве подхода было выбрано ООП.

Разработано консольное приложение на языке C++

Среда разработки: JetBrains CLion 2020.2.4

В .h файлах содержатся объявления классов, а в .cpp содержится описание классов

Класс Post содержит два конструктора: параметризованный и по умолчанию. Первый принимает параметры, а второй присваивает значения полей по умолчанию (все значения полей нулевые). В классе реализовано 14 методов:

FullChange – изменяет весь адрес и информацию о компании, ChangeAddress – изменяет весь адрес, ChangeCountry – изменяет название страны, ChangeCity – изменяет название города, ChangeStreet – изменяет название улицы, ChangeHouseNumber – изменяет номер дома, ChangePostIndex – изменяет почтовый индекс, ClearAll – удаляет всю информацию, GetAddress – возвращает адрес, GetCountry – возвращает название страны, GetCity – возвращает название города, GetStreet – возвращает название улицы, GetHouseNumber – возвращает номер дома, GetPostIndex – возвращает почтовый индекс

Класс CompanyInfo является дочерним классом класса Post. Содержит один конструктор – по умолчанию (все значения полей нулевые). В дочернем классе реализованы 7 методов:

ChangeCompanyInfo – изменяет информацию о компании, ChangeName – изменяет название компании, ChangePopulation – изменяет количество работников, ClearInfo – удаляет информацию о компании, GetCompanyInfo – возвращает информацию о компании, GetName – возвращает название компании, GetPopulation – возвращает количество работников

Если параметры адреса не будут удовлетворять условиям (определенные названия состоят не только из букв, длина индекса больше или меньше шести), то выброситься ошибка с помощью throw invalid\_argument, далее оно ловится в блоке try-catch и в консоль выводится соответствующее сообщение.

* 1. Тестирование программы

Объект испытаний – программа Lab2-3

Цель испытаний – проверить работу программы

Средства испытаний – локальный отладчик XCode

Метод испытаний – функциональное тестирование при средних и аномальных значениях.

Табл. 3 Тесты

|  |  |  |
| --- | --- | --- |
| № Теста | Тест | Описание теста |
| 1 | f  Enter address  Enter country:  Russia  Russia  Enter city:  Moscow  Moscow  Enter street:  Redbarracs  Redbarracs  Enter house number:  17/3  17/3  Enter post index:  111250  111250  Enter name of company:  MPEI  MPEI  Enter population of company:  1000  1000  Address have been added:  Address: Russia, Moscow, Redbarracs, 17/3, 111250  Company: MPEI (1000);  c  Enter country:  Rusland  Rusland  t  Enter city:  Maskau  Maskau  s  Enter street:  Krasnokazarmennaya  Krasnokazarmennaya  h  Enter house number:  18a  18a  i  Enter post index:  100200  100200  n  Enter name of company:  MoscowPowerEngineeringInstitute  MoscowPowerEngineeringInstitute  p  Enter population of company:  2001  2001  v  Full address with company info:  Address: Rusland, Maskau, Krasnokazarmennaya, 18a, 100200  Company: MoscowPowerEngineeringInstitute (2001);  b  All clear | Полностью корректный ввод с последующим поэлементным изменением всех переменных и выводом результата на экран |
| 2 | c  Enter country:  R239-0  R239-0  Incorrect symbol | попытка ввода страны с использованием некорректных символов |
| 3 | t  Enter city:  Moscow2  Moscow2  Incorrect symbol | попытка ввода города с использованием некорректных символов |
| 4 | s  Enter street:  Kuznezkaya2  Kuznezkaya2  Incorrect symbol | попытка ввода улицы с использованием некорректных символов |
| 5 | h  Enter house number:  e20  e20  Invalid number of house | попытка ввода номера дома, начиная не с цифры |
| 6 | i  Enter post index:  10230  10230  Invalid index of post | попытка ввода слишком короткого индекса |
| 7 | i  Enter post index:  11111111  11111111  Invalid index of post | попытка ввода слишком длинного индекса |
| 8 | i  Enter post index:  e20302  e20302  Incorrect symbol | попытка ввода индекса с использованием некорректного символа |
| 9 | p  Enter population of company:  many  many  Invalid population | попытка ввода числа работников с использованием некорректных символов |
| 10 | v  Full address with company info:  No country name | попытка вывода на экран после некорректных вводов |
| 11 | f  Enter address  Enter country:  Rus3ia  Rus3ia  Incorrect symbol | попытка ввода адреса полностью с некорректной страной (Аналогично со всеми данными) |

Табл. 3 Тесты продолжение

**Вывод**

Данная программа выполняет требования, данные в задании: имеет основной и дочерний классы, которые содержат данные и методы их обработки, присутствует возможность поэлементного редактирования данных, их просмотра, а также очистки информации, содержащейся в классе

**Приложение. Код программы**

Табл. 4 Код

**post.cpp**

#include **“post.h”**#include **<sstream>  
void** CheckForCorrectInput(**const** string & input)  
{  
 **for** (**const auto** & character : input)  
 {  
 **if** (!isalpha(character))  
 {  
 **throw** invalid\_argument(**“Incorrect symbol”**);  
 }  
 }  
}  
**void** CheckForCorrectNumber(**const** string & number)  
{  
 **for** (**const auto** & character : number)  
 {  
 **if** (!isdigit(character))  
 {  
 **throw** invalid\_argument(**“Incorrect symbol”**);  
 }  
 }  
}  
Post::Post() = **default**;  
Post::Post(**const** string &address, **const** string & name, **const** string & pop)  
{  
 FullChange(address, name, pop);  
}  
**void** Post::FullChange(**const** string & address, **const** string & name, **const** string & pop)  
{  
 ChangeCompanyInfo(name, pop);  
 ChangeAddress(address);  
}  
**void** Post::ChangeAddress(**const** string & address)  
{  
 stringstream stream(address); *// Создание потока из строки адреса* string country, city, street, number, index; *// Отдельные переменные адреса* getline(stream, country, **‘,’**); *// Считывание переменной до знака запятой* **if** (stream.peek() == **‘ ‘**) *// Пропуск пробела, если существует* {  
 stream.ignore(1);  
 }  
 *//// Аналогично с остальными переменными* getline(stream, city, **‘,’**);  
 **if** (stream.peek() == **‘ ‘**)  
 {  
 stream.ignore(1);  
 }  
 getline(stream, street, **‘,’**);  
 **if** (stream.peek() == **‘ ‘**)  
 {  
 stream.ignore(1);  
 }  
 getline(stream, number, **‘,’**);  
 **if** (stream.peek() == **‘ ‘**)  
 {  
 stream.ignore(1);  
 }  
 getline(stream, index, **‘\n’**);  
 *//// Обработка отдельных переменных адреса с помощью соответствующих методов* ChangeCountry(country);  
 ChangeCity(city);  
 ChangeStreet(street);  
 ChangeHouseNumber(number);  
 ChangePostIndex(index);  
}

**void** Post::ChangeCountry(**const** string & country)  
{  
 **if** (!country.empty()) *// Проверка на пустоту* {  
 CheckForCorrectInput(country);  
 country\_ = country; *// Присвоение переменной* } **else  
 throw** invalid\_argument(**“Invalid name of country”**); *// Выброс исключения*}  
**void** Post::ChangeCity(**const** string &city)  
{  
 **if** (!city.empty())  
 {  
 CheckForCorrectInput(city);  
 city\_ = city;  
 } **else  
 throw** invalid\_argument(**“Invalid name of city”**);  
}  
**void** Post::ChangeStreet(**const** string &street)  
{  
 **if** (!street.empty())  
 {  
 CheckForCorrectInput(street);  
 street\_ = street;  
 } **else  
 throw** invalid\_argument(**“Invalid name of street”**);  
}  
**void** Post::ChangeHouseNumber(**const** string &number)  
{  
 **if** (!number.empty())  
 {  
 **if** (!isdigit(number[0]))  
 {  
 **throw** invalid\_argument(**“Invalid number of house”**);  
 }  
 house\_number\_ = number;  
 } **else  
 throw** invalid\_argument(**“Invalid number of house”**);  
}  
**void** Post::ChangePostIndex(**const** string &index)  
{  
 **if** (index.size() == 6)  
 {  
 CheckForCorrectNumber(index);  
 post\_index\_ = index;  
 } **else  
 throw** invalid\_argument(**“Invalid index of post”**);  
}  
**void** Post::ClearAll()  
{  
 country\_.clear();  
 city\_.clear();  
 street\_.clear();  
 house\_number\_.clear();  
 post\_index\_.clear();  
 ClearInfo();  
}  
  
string Post::GetAddress() **const**{  
 **return “Address: “** + GetCountry() + **“, “** + GetCity()  
 + **“, “** + GetStreet()+ **“, “** + GetHouseNumber() + **“, “** + GetPostIndex() + **“\nCompany: “** + GetName() + **“ (“** + GetPopulation() + **“);”**;  
}  
string Post::GetCountry() **const**{  
 **if** (!country\_.empty())  
 {  
 **return** country\_;  
 } **else  
 throw** invalid\_argument(**“No country name”**);  
}

string Post::GetCity() **const**{  
 **if** (!city\_.empty())  
 {  
 **return** city\_;  
 } **else  
 throw** invalid\_argument(**“No city name”**);  
}  
string Post::GetStreet() **const**{  
 **if** (!street\_.empty())  
 {  
 **return** street\_;  
 } **else  
 throw** invalid\_argument(**“No street name”**);  
}  
string Post::GetHouseNumber() **const**{  
 **if** (!house\_number\_.empty())  
 {  
 **return** house\_number\_;  
 } **else  
 throw** invalid\_argument(**“No house number”**);  
}  
string Post::GetPostIndex() **const**{  
 **if** (!post\_index\_.empty())  
 {  
 **return** post\_index\_;  
 } **else  
 throw** invalid\_argument(**“No post index”**);  
}

**post.h**

#include **<string>**#include **"company.h"  
  
using namespace** std;  
  
**void** CheckForCorrectInput(**const** string & input); *// Функция проверки ввода на буквенные значения***void** CheckForCorrectNumber(**const** string & number); *// Функция проверки ввода на числовые значения  
  
//Основной класс с информацией об адресе***class** Post : **public** CompanyInfo  
{  
**public**:  
 Post(); *// Конструктор по умолчанию* Post(**const** string & address, **const** string & name, **const** string & pop); *// Конструктор класса* **void** FullChange(**const** string & address, **const** string & name, **const** string & pop); *// Конструктор с входными данными (строка адреса, строка имени компании, строка с количеством работников* **void** ChangeAddress(**const** string & address); *// Функция изменения всего адреса* **void** ChangeCountry(**const** string & country); *// Функция изменения адреса без изменения информации о компании* **void** ChangeCity(**const** string & city); *// Функция изменения страны* **void** ChangeStreet(**const** string & street); *// Функция изменения города* **void** ChangeHouseNumber(**const** string & number); *// Функция изменения района* **void** ChangePostIndex(**const** string & index); *// Функция изменения улицы* **void** ClearAll(); *// Функция очистки всего адреса* string GetAddress() **const**; *// Функция возврата всего адреса* string GetCountry() **const**; *// Функция возврата страны* string GetCity() **const**; *// Функция возврата города* string GetStreet() **const**; *// Функция возврата улицы* string GetHouseNumber() **const**; *// Функция возврата номера дома* string GetPostIndex() **const**; *// Функция возврата почтового индекса***private**:  
 string country\_; *// Переменная, содержащая название страны* string city\_; *// Переменная, содержащая название города* string street\_; *// Переменная, содержащая название улицы* string house\_number\_; *// Переменная, содержащая номер дома* string post\_index\_; *// Переменная, содержащая почтовый индекс*};

**company.cpp**

#include **"company.h"**CompanyInfo::CompanyInfo()= **default**;  
  
**void** CompanyInfo::ChangeCompanyInfo(**const** string & company, **const** string & pop)  
{  
 ChangeName(company);  
 ChangePopulation(pop);  
}  
**void** CompanyInfo::ChangeName(**const** string & name)  
{  
 **if** (!name.empty())  
 {  
 name\_ = name;  
 } **else  
 throw** invalid\_argument(**"Invalid name"**);  
}  
**void** CompanyInfo::ChangePopulation(**const** string & population)  
{  
 **bool** d = **true**;  
 **for** (**auto** & p : population)  
 {  
 **if** (!isdigit(p))  
 {  
 d = **false**;  
 **break**;  
 }  
 }  
 **if** (population != **"0"** && d)  
 {  
 population\_ = population;  
 } **else  
 throw** invalid\_argument(**"Invalid population"**);  
}  
**void** CompanyInfo::ClearInfo()  
{  
 name\_.clear();  
 population\_.clear();  
}  
  
string CompanyInfo::GetCompanyInfo() **const**{  
 **return** name\_ + **' '** + population\_;  
}  
string CompanyInfo::GetName() **const**{  
 **return** name\_;  
}  
string CompanyInfo::GetPopulation() **const**{  
 **return** population\_;  
}

**company.h**

#include **<string>  
using namespace** std;  
*// Дочерний класс, содержащий информацию о компании***class** CompanyInfo  
{  
**public**:  
 CompanyInfo();  
 **void** ChangeCompanyInfo(**const** string & company, **const** string & pop); *// Функци изменения названия и числа работников компании* **void** ChangeName(**const** string & name); *// Функция изменения названия компании* **void** ChangePopulation(**const** string & population); *// Функция изменения числа работников компании* **void** ClearInfo(); *// Функция очистки информации, содержащейся в классе* string GetCompanyInfo() **const**; *// Функция возврата информации, содержащейся в классе* string GetName() **const**; *// Функция возврата названия компании* string GetPopulation() **const**; *// Функция возврата числа работников***private**:  
 string name\_; *// Переменная, содержащая название компании* string population\_; *// Переменная, содержащая число работников*};

**main.cpp**

#include **<iostream>**#include **<vector>**#include **"post.h"  
  
using namespace** std;  
  
  
**void** AddFullAddress(Post & post)  
{  
 string country, city, street, house; *// Переменные почтового адреса* string name\_company; *// Переменная названия компании* string index;  
 string population; *// Переменная количества работников* cout << **"Enter address"** << endl;  
 cout << **"Enter country:"** << endl;  
 cin >> country;  
 cout << country << endl;  
 post.ChangeCountry(country);  
 cout << **"Enter city:"** << endl;  
 cin >> city;  
 cout << city << endl;  
 post.ChangeCity(city);  
 cout << **"Enter street:"** << endl;  
 cin >> street;  
 cout << street << endl;  
 post.ChangeStreet(street);  
 cout << **"Enter house number:"** << endl;  
 cin >> house;  
 cout << house << endl;  
 post.ChangeHouseNumber(house);  
 cout << **"Enter post index:"** << endl;  
 cin >> index;  
 cout << index << endl;  
 post.ChangePostIndex(index);  
 cout << **"Enter name of company:"** << endl;  
 cin >> name\_company; *// Считывание названия компании из консоли* cout << name\_company << endl;  
 post.ChangeName(name\_company);  
 cout << **"Enter population of company:"** << endl;  
 cin >> population;  
 cout << population << endl;  
 post.ChangePopulation(population);  
 cout << **"Address have been added:"** << endl;  
 cout << post.GetAddress() << endl; *// Вывод обработанной информации*}  
  
**int** main()  
{  
 Post post;  
 **char** input; *// Переменная выбора* **bool** end = **false**; *// Переменная, отвечающая за окончание работы программы* **bool** check = **true**;  
 **while** (check)  
 **try** {  
 AddFullAddress(post);  
 check = **false**;  
 } **catch** (**const** invalid\_argument & invalid\_argument) {  
 cout << invalid\_argument.what() << endl;  
 check = **true**;  
 }  
 **while** (!end) {  
 cout  
 << **"Choose, what you want to do:\n"  
 "F - add new full address with company info\n"  
 "C - change country\n"  
 "T - change city\n"  
 "S - change street\n"  
 "H - change house number\n"  
 "I - change post index\n"  
 "N - change company name\n"  
 "P - change population of company\n"  
 "V - show full address\n"  
 "B - to clear all\n"  
 "E - to exit"** << endl;  
 cin.get(input); *// Считывание переменной второго выбора из консоли* input = toupper(input); *// Перевод переменной в верхний регистр* **try** {  
 **switch** (input) *// Меню действий, которые может совершить программа над переменной класса* {  
 **case 'F'**: {  
 AddFullAddress(post);  
 **break**;  
 }  
 **case 'C'**: {  
 string country; *// Переменная страны* cout << **"Enter country:"** << endl;  
 cin >> country;  
 cout << country << endl;  
 post.ChangeCountry(country);  
 **break**;  
 }  
 *//// Аналогично стране происходит обработка города, района, улицы, номера дома, почтового индекса, названия компании и количества работников* **case 'T'**: {  
 string city;  
 cout << **"Enter city:"** << endl;  
 cin >> city;  
 cout << city << endl;  
 post.ChangeCity(city);  
 **break**;  
 }  
 **case 'S'**: {  
 string street;  
 cout << **"Enter street:"** << endl;  
 cin >> street;  
 cout << street << endl;  
 post.ChangeStreet(street);  
 **break**;  
 }  
 **case 'H'**: {  
 string house\_number;  
 cout << **"Enter house number:"** << endl;  
 cin >> house\_number;  
 cout << house\_number << endl;  
 post.ChangeHouseNumber(house\_number);  
 **break**;  
 }  
 **case 'I'**: {  
 string index;  
 cout << **"Enter post index:"** << endl;  
 cin >> index;  
 cout << index << endl;  
 post.ChangePostIndex(index);  
 **break**;  
 }  
 **case 'N'**: {  
 string name\_company;  
 cout << **"Enter name of company:"** << endl;  
 cin >> name\_company; *// Считывание названия компании из консоли* cout << name\_company << endl;  
 post.ChangeName(name\_company);  
 **break**;  
 }  
 **case 'P'**: {  
 string population;  
 cout << **"Enter population of company:"** << endl;  
 cin >> population;  
 cout << population << endl;  
 post.ChangePopulation(population);  
 **break**;  
 }  
 **case 'V'**: {  
 cout << **"Full address with company info:"** << endl;  
 cout << post.GetAddress() << endl; *// Вывод информации, которую содержит класс адреса  
 // с помощью специального метода этого класса* **break**;  
 }  
 **case 'B'**: {  
 post.ClearAll();  
 cout << **"All clear"** << endl;  
 **break**;  
 }  
 **case 'E'**: {  
 end = **true**; *// Переменная окончания программы переводится в положение true* **break**;  
 }  
 **default**: {  
 cout << **"Unknown command, please try again."** << endl;  
 **break**;  
 }  
 }  
 } **catch** (**const** invalid\_argument &i) *// Поиск выброса исключений* {  
 cout << i.what() << endl; *// Вывод расшифровки исключений* }  
 }  
 **return** 0;  
}