

Лабораторная работа №1

Тема: Повторение пройденного материала: типы данных и функции в структурном программировании

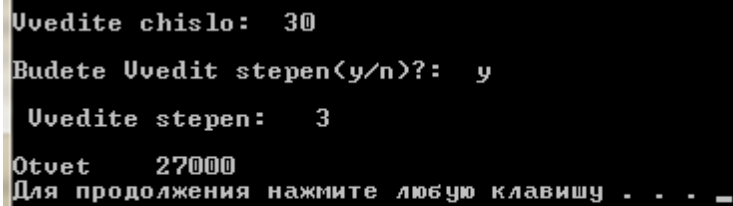
Задания

1. Возведение числа **n** в степень **p** – это умножение числа **n** на себя **p** раз.

Напишите функцию с именем **power()**, которая в качестве аргументов принимает значение типа **double** для **n** и значение типа **int** для **p** и возвращает значение типа **double**.

Для аргумента, соответствующего степени числа, задать значение по умолчанию, равное **2**, чтобы при отсутствии показателя степени при вызове функции число **n** возводилось в квадрат. Сделайте проверку, будет ли пользователь вводить степень числа. Если будет, то эта степень – число, большее 2. Напишите функцию **main()**, которая запрашивает у пользователя ввод аргументов для функции **power()** и отображает на экране результаты её работы. Напишите код первого задания с *объявлением* (прототипом) функции **power()**.

Результат работы:



```
Uvedite chislo: 3
Budete Uvedit stepen(y/n)? : y
Uvedite stepen: 3
Otvet 27
Для продолжения нажмите любую клавишу . . .
```

2. Взяв в качестве основы функцию **power()** из первого задания, работающую только со значением типа **double**. Создайте перегруженные функции с этим же именем, принимающими в качестве аргумента значения типа **char**, **int**, **long**, **float**. Напишите программу, вызывающую функцию **power()** со всеми возможными типами аргументов. Напишите код второго задания без *объявления* функции **power()**.

Задание

3. Описать структуру с именем **AEROFLOT**, содержащую следующие поля:

- Название пункта назначения рейса;
- Номер рейса;
- Тип самолета;

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из семи элементов типа **AEROFLOT**; записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;
- вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры;
- если таких рейсов нет, выдать соответствующее сообщение.

4. Все задания объединить с помощью меню, т.е. представить как готовый программный продукт.

Контрольные вопросы

1. Какие стандартные типы используются в C++? Сколько под них резервируется памяти?
2. К каким элементам программы относятся следующие:

- а) 12;
- б) 'a';
- в) 4.28915;
- г) JungleJim;
- д) JungleJim().

3. Что такое функция? Какова роль функций в языке C++?

4. Напишите синтаксис функции.

5. Опишите способы использования функций в программах с объявлением функций и без объявления функций.

6. Объясните механизмы передачи аргументов по значению и по ссылке в функцию. Объясните результаты работы программ.

1)

```
#include <iostream>
using namespace std;

int incr(int m){
    m=m+1;
    return m;
}

int main(){
    int n=5;
    cout<<"n ="<<incr(n)<<endl;
    cout<<"n ="<<n<<endl;
    return 0;
}
```

2)

```
#include <iostream>
using namespace std;
int incr(int &m){
    m=m+1;
    return m;
}

int main(){
    int n=5;
    cout<<"n ="<<incr(n)<<endl;
    cout<<"n ="<<n<<endl;
    return 0;
}
```

7) Назовите разновидности аргументов, которые могут быть переданы параметрам функции?

8. Что такое аргументы по умолчанию?

Объясните результат работы программы:

```
#include <iostream>
using namespace std;
//Аргумент функции имеет значение по умолчанию:
void showX(int x=0){
    cout<<"x = "<<x<<endl;
}

//Два аргумента функции в прототипе имеют значение по
//умолчанию,
//сама функция описана в конце программы:
void showXYZ(int x,int y=1,int z=2);

int main(){
    showX(3);
    showX();
    showXYZ(4,5,6);
    showXYZ(7,8);
    showXYZ(9);
    return 0;
}

//При описании функции значения по умолчанию не указываются:
void showXYZ(int x,int y,int z){
    cout<<"x = "<<x<<" ";
    cout<<"y = "<<y<<" ";
    cout<<"z = "<<z<<endl;
}
```

9. Что такое перегрузка функций?

Литература

1) Р. Лафоре «Объектно-ориентированное программирование в C++», 4-е издание, Питер, 2004 г., гл.2, стр. 48 «Основы программирования на C++»

гл. 5. стр. 168. «Функции»

гл. 5. стр. 192 «Перегруженные функции»

гл. 5. стр. 200 «Аргументы по умолчанию»

2) Т.А. Павловская «C/C++ программирование на язык высокого уровня», гл. 1, стр.13-22 «Типы данных C++», стр.73 «Функции», стр. 77 «Параметры функции».

Лабораторная работа № 2

Тема: Исследование на практике программирования классов с закрытыми полями и открытыми методами.

Литература

Р. Лафоре «Объектно-ориентированное программирование в C++», 4-е издание, Питер, 2004 г., гл. 6 стр.217 «Объекты и классы», стр.282 «Строки», стр.288 «Копирование строк более простым способом», стр.294 «Стандартный класс **string** языка C++»

Гл. 7 стр. 275 «Массивы как члены классов»

Задания

1. Создайте класс **card**, который поддерживает каталог библиотечных карточек.

Этот класс должен хранить заглавие книги, имя автора и выданное на руки число экземпляров книги. Заглавие и имя автора храните в виде строки символов, а количество экземпляров—в виде целого числа. Используйте открытый метод **set_card()** для запоминания информации о книгах и открыт метод **show_card()** для вывода информации на экран. В методе **set_card()** можете использовать библиотечную функцию **strcpy()** для копирования символьной константы в символьную переменную. Поля класса закрыты. В функцию **main()** включите краткую демонстрацию работы созданного класса.

2. Создайте класс **Student**, который содержит информацию об имени студента - **name**, его фамилии – **last_name**, оценок, полученных на экзамене и среднего арифметического значения оценок. Имя и фамилию студента храните в виде строки символов **string**, оценки в виде массива **scores[5]**, тип среднего арифметического значения - **float**. Поля – закрыты. Используйте открытые методы. Назовите их **set_name()** – установка значения имени, **set_last_name()**- установка значения фамилии, **set_scores()** – установка значений для массива оценок, **set_average_scores(float scores)** – установка среднего балла, а также метод отображения массива оценок **show_scores()**. Организуйте вывод значений всех полей объекта и расчетного среднего значения на экран.

В функцию **main()** включите краткую демонстрацию работы созданного класса.

```
Name: Стас
Last name: Иманкулов
Score 1: 5
Score 2: 4
Score 3: 5
Score 4: 4
Score 5: 5
Scores: 5 4 5 4 5
Average ball for Стас Иманкулов is 4.6
Для продолжения нажмите любую клавишу . . . _
```

3. Контрольные вопросы

1. Как вы понимаете, что такое класс и объект? Приведите примеры.
2. Каков синтаксис объявления класса? Как называются составные части класса?
3. В чем отличие закрытых и открытых членов класса?
4. Сформулируйте основополагающую концепцию закрытых полей и открытых методов класса.
5. Где в программе с классами можно создавать объекты?
6. Сколько объектов и методов можно создать в программе с классами?
- В какой момент метод готов для вызова объектом?
8. Какие способы заполнения полей объекта информацией Вы знаете?
9. Зависит ли результат работы метода объекта от объекта?
10. Если в классе одно поле данных и два объекта, сколько полей принадлежит каждому объекту? Совпадает ли имена и значения этих полей для объектов?
11. Чем отличается массив элементов типа **char** от объектов стандартного класса **string**?

Лабораторная № 3

Темы:

1. исследование на практике работы конструктора и деструктора, как методов класса
2. перегрузка конструктора
3. конструктор копирования

Теория

Р. Лафоре «Объектно-ориентированное программирование в C++», 4-е издание, Питер, 2004 г., гл. 6
стр.227 «Конструкторы»,
стр. 232 «Деструкторы»,
стр. 234 «Перегруженный конструктор»,
стр. 237 «Конструктор копирования по умолчанию».

Задание

1.Создайте класс **t_and_d**, которому при его создании передается текущее системное время и дата в виде параметров конструктора. Этот класс должен включать в себя функцию - член, выводящую время и дату на экран. Используйте деструктор.

(Подсказка: для нахождения и вывода на экран этих данных воспользуйтесь стандартной библиотечной функцией времени и даты.)

2.Создайте класс **cbox**, конструктор у которого передаются три значения типа **double**, представляющие собой длины сторон параллелепипеда. Класс **box** должен подсчитывать его объем и хранить результат также в виде значения типа **double**. Включите в класс функцию-член **vol()**, которая будет выводить на экран объем любого объекта типа **box**. Используйте деструктор.

. Разработайте программу с классом **timer**, в которой таймер действует как таймер обратного отсчета. При создании объекта типа **timer** ему присваивается начальное значение времени. В результате вызова функции **run()** таймер начинает отсчет в сторону уменьшающихся значений, пока не достигнет значения 0, после чего зазвонит звонок. **Создайте 3 варианта конструктора, т.е. конструктор перегрузите 3 раза** для того, чтобы можно было указывать время в секундах с помощью целого числа или строки, или в минутах и секундах, если указываются два целых числа.

Эта программа использует библиотечную функцию **clock()**, возвращающую число тиков, прошедших с момента запуска программы. Поделив это значение на макрос **CLK_TCK**, получаем значение в секундах. Прототипы для **clock()** и **CLK_TCK** содержатся в заголовочном файле **time.h**.

4. В задании 1 изменить программу так, чтобы можно было использовать **Конструктор копирования** по умолчанию.

Задание

1. **Описать класс, реализующий стек.** Написать программу, использующую этот класс для моделирования Т-образного сортировочного узла на железной дороге. Программа должна разделять на два направления состав, состоящий из вагонов двух типов (на каждое направление формируется состав из вагонов одного типа). Предусмотреть возможность формирования состава из файла и с клавиатуры.

2. Все выполненные задания объединить с помощью меню, т.е. представить как готовый программный продукт

Контрольные вопросы

11. Что такое конструктор в классах? Для чего его используют?
12. Почему возникла необходимость в использовании конструктора?
13. Какие способы инициализации полей объекта вы знаете?
14. Напишите синтаксис простого конструктора?
15. Можно ли при инициализации полей использовать тело конструктора?
16. Сколько раз конструктор вызывается для инициализации, если количество полей объекта равно четырем?
17. Назовите особенности конструктора, явно отличающие его от других методов.
18. Как работает конструктор с аргументами, чему присваиваются аргументы в качестве значений?
19. Можно ли перегружать конструктор?
20. Что такое конструктор копирования?

Лабораторная работа № 4

Тема: Практическое исследование работы дружественной функции и дружественного класса.

Теория:

Р. Лафоре «Объектно-ориентированное программирование в C++», 4-е издание, Питер, 2004 г., гл. 11 стр.491 «Дружественные функции», стр.499 «Дружественные классы».

Задание:

1. Составить программу, в которой создаются классы **car** (легковая машина) и **truck** (грузовик), причем оба содержат в закрытой переменной скорость соответствующего транспортного средства. Создайте функцию **sp_greater()**, дружественную для обоих классов. Эта функция возвращает положительное число, если объект **car** движется быстрее объекта **truck**, нуль, если их скорости одинаковы, и отрицательное число, если скорость объекта **truck** больше, чем скорость объекта **car**. Разработать два варианта программы:

- 1) С использованием конструктора с одним аргументом;
 - 2) С использованием метода установки значения полю «скорость».
2. Функция может быть дружественной одному классу. Напишите программу, в которой функция **show()** является дружественной классу **truck**.
3. Класс может быть дружественным другому классу. Перепишите задание 2 так, чтобы класс **car** подружился с классом **truck**.

Задание

В классе Distance, как показано в примере ниже, создайте перегруженную операцию умножения *, чтобы можно было умножить два расстояния. Сделайте эту функцию дружественной, тогда можно будет использовать выражение типа `Wdist=7.5 *dist`. Вам понадобится конструктор с одним аргументом для перевода величин из формата чисел с плавающей запятой в формат Distance. Напишите какой-либо `main()` на свое усмотрение для того, чтобы несколькими способами проверить работу этой перегруженной операции.

```
// frengl.cpp
// Дружественная перегружаемая операция +
#include <iostream>
using namespace std;
////////////////////////////////////
class Distance                //Класс английских расстояний
{
private:
    int feet;
    float inches;
public:
    Distance()                //конструктор без аргументов
    { feet = 0; inches = 0.0; }
    Distance( float fltfeet ) //конструктор (1 арг.)
    { //Переводит float в Distance
      feet = int(fltfeet);    //feet - целая часть
      inches = 12*(fltfeet-feet); //слева - дюймы
    }
    Distance(int ft, float in) //конструктор (2 арг.)
    { feet = ft; inches = in; }
    void showdist()            //Вывести длину
    { cout << feet << "'-" << inches << "'"; }
    friend Distance operator + (Distance, Distance): //дружественный
};
//-----

Distance operator + (Distance d1, Distance d2) // d1 + d2
{
    int f = d1.feet + d2.feet;    //+ футы
    float i = d1.inches + d2.inches; //+ дюймы
    if(i >= 12.0)                //если больше 12 дюймов,
    { i -= 12.0; f++; }          //уменьшить на 12 дюймов,
                                //прибавить 1 фут
    return Distance(f,i);        //Новая длина с суммой
}
//-----

int main()
{
    Distance d1 = 2.5;           //конструктор переводит
    Distance d2 = 1.25;          //float-feet в Distance
    Distance d3;
    cout << "\nd1 = ": d1.showdist();
    cout << "\nd2 = ": d2.showdist();

    d3 = d1 + 10.0;              //distance + float: OK
    cout << "\nd3 = ": d3.showdist();
    d3 = 10.0 + d1;              //float + Distance: OK
    cout << "\nd3 = ": d3.showdist();
    cout << endl;
    return 0;
}
```

Контрольные вопросы

1. Что такое дружественная функция, для чего она нужна?
2. Как она объявляется, каков её синтаксис?
3. Через что дружественная функция может получить доступ к закрытому полю класса?
4. Какой вариант вызова ДФ правильный:
a.show() или **show(a)**?
5. Наследуется ли дружественная функция?
6. Может ли дружественная функция быть дружественной более чем к одному классу?

7. Для решения каких задач есть смысл использовать функцию, дружественную двум и более классам?
8. Когда нужно использовать дружественные классы?
9. Какому принципу ООП не соответствуют дружественные функции?
10. Как можно ограничить использование ДФ?
11. В каких крайних случаях нужно использовать ДФ?

ЛАБОРАТОРНАЯ РАБОТА №5

Тема: исследование на практике перегрузки операторов и функций

Задания:

1. Опишите класс **fraction**, у которого поля *x* и *y* задают числитель и знаменатель обыкновенной дроби. Перегрузите для этого класса арифметические операции сложения, вычитания, умножения и деления так, чтобы они могли оперировать как с объектами класса, так и с числами (то есть выполнять, например, не только действие $3/4 + 2/5$, но и $1/2 + 4$ или $2 * 5/6$). Также перегрузите унарную операцию инкремента в префиксной или постфиксной форме увеличения дроби. Протестируйте работу класса.
2. Создать программу, в которой перегружается функция **rect_area()**. Эта функция возвращает площадь прямоугольника. В этой программе функция **rect_area()** перегружается двумя способами. В первом—функции передаются оба размера фигуры. Эта версия используется для прямоугольника. Однако, в случае квадрата необходимо задавать только один аргумент, поэтому вызывается вторая версия функции **rect_area()**.

Задание

1. Создайте класс **Int**, основанный на классе в нижеприведенной программе. Перегрузите четыре целочисленных арифметических операции (+, -, *, /) так, чтобы их можно было использовать для операций с объектами класса **Int**. Если результат какой-либо из них выходит за границы типа **int** (в 32-битной системе), имеющие значения от -2 147 483 648 до 2 147 483 648, то операция должна послать сообщение об ошибке и завершить программу. Такие типы данных полезны там, где ошибки могут быть вызваны арифметическим переполнением, которое не допустимо.

Для облегчения проверки переполнения выполняйте вычисления с использованием типа `long double`. Напишите программу для проверки этого класса.

Контрольные вопросы

1. Что такое перегрузка операторов?
2. Что такое переопределение операторов?
3. Каков синтаксис операторной функции?
4. Какие операторы можно перегружать, а какие нельзя?
5. Какова зависимость между количеством аргументов в операторной функции и количеством операндов? Объясните эту зависимость.
6. Каким образом перегруженные операции позволяют вид программного кода сделать более читабельным?
7. Что такое перегрузка методов?

Лабораторная работа № 6

Цель работы: Освоить на практике правила построения класса исключений на языке программирования C++ в среде Visual Studio 2010.

Теория:

Синтаксис исключений

Представим себе приложение, создающее объекты какого-либо класса и работающее с ними. В нормальной ситуации вызовы методов не приводят ни к каким ошибкам. Но иногда в программе возникает ошибка, которую обнаруживает метод. Он информирует приложение о случившемся. При использовании исключений это событие называется генерацией исключительной ситуации. В приложении при этом имеется отдельная секция кода, в которой содержатся операции по обработке ошибок. Этот код называют *обработчиком исключительных ситуаций* или *улавливающим блоком*. В нем отлавливаются исключения, сгенерированные методами. Любой код приложения, использующий объекты класса, заключается в *блок повторных попыток*. Соответственно, ошибки, возникшие в последнем, будут пойманы улавливающим блоком. Код, который не имеет никакого отношения к классу, не включается в блок повторных попыток. На рис. 14.4 показана схема, описанная выше.

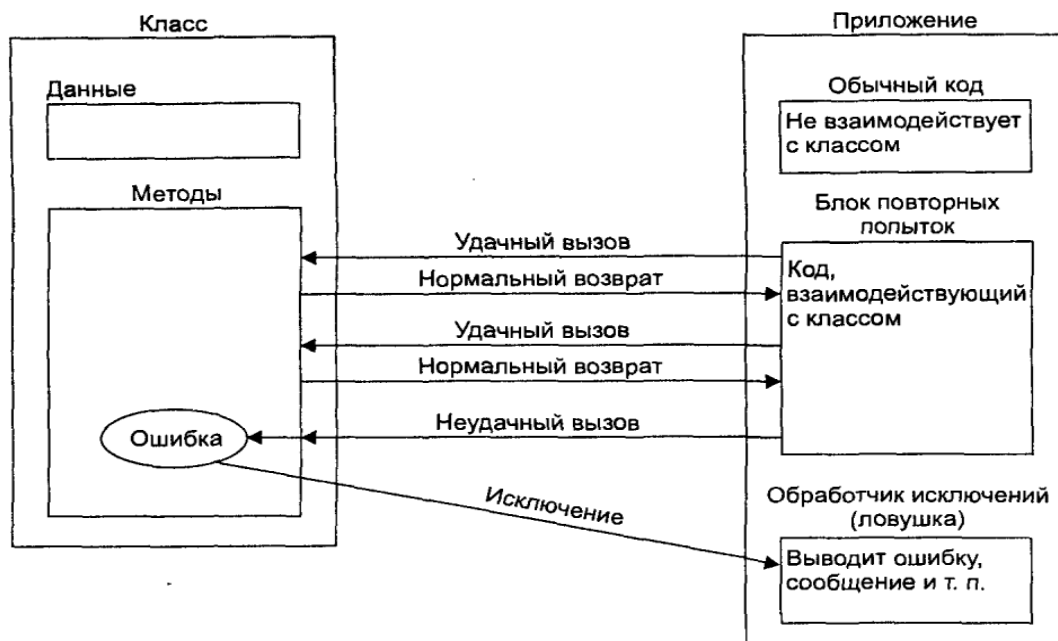


Рис. 14.4. Механизм исключений

Механизм исключений использует три новых слова C++: `catch`, `throw` и `try`. Кроме того, нужно создать новый тип сущности, называемый классом `exception`. Программа XSYNTAX — это еще не нормальная программа, а только скелет, приводимый для того, чтобы показать синтаксис.

Листинг 14.7. Скелет программы XSYNTAX

```
// xsyntax.cpp
// эта программа не работает!
////////////////////////////////////
class AClass          // просто класс
{
public:
    class AnError      // класс exception
    {
    };
    void Func()        //какой-то метод
    {
        if( /* условие ошибки */ )
            throw AnError(); // генерировать исключение
    }
};
////////////////////////////////////
int main()             // приложение как бы
{
    try                // блок повторных попыток
    {
        AClass obj1;   // взаимодействие с объектами AClass
        obj1.Func();   // тут может возникнуть ошибка
    }
    catch(AClass::AnError) //обработчик ошибок
    {                    // (улавливающий блок)
    }
    return 0;
}
```

Мы начинаем работу с создания класса AClass, представляющего собой некий класс, в котором могут возникнуть ошибки. Класс исключений, AnError, определен в общедоступной части класса AClass. Методы последнего подвергаются тщательному обследованию на предмет выявления ошибок. Если таковые обнаруживаются, мы с помощью ключевого слова throw и конструктора класса исключений производим генерацию исключительной ситуации:

```
throw AnError();      //'throw' и конструктор класса AnError
```

В main() мы заключаем все выражения, каким-либо образом относящиеся к AClass, в блок повторных попыток. Если какие-то из них приводят к тому, что возникают подозрения на наличие ошибки в методе AClass, то генерируется исключение, и управление программой переходит к улавливающему блоку, который следует сразу же за блоком повторных попыток.

Задание общее:

Добавьте класс исключений к программе **ARROVER1**, чтобы индексы, выходящие за пределы массива, вызвали генерацию исключения. Блок-ловушка может выводить пользователю сообщение об ошибке. Листинг программы **ARROVER1**:

```
// arrover1.cpp
// демонстрация создания безопасного массива, проверяющего
// свои индексы при использовании
// используются отдельные функции для установки и получения значения
#include <iostream>
using namespace std;
#include <process.h>    // для функции exit
const int LIMIT = 100; // размер массива
////////////////////////////////////
class safearray
{
private:
    int arr [ LIMIT ];
public:
    // установка значения элемента массива
    void putel ( int n, int elvalue )
    {
        if ( n < 0 || n >= LIMIT )
            { cout << "\nОшибочный индекс!"; exit ( 1 ); }
        arr [ n ] = elvalue;
    }
    // получение значения элемента массива
    int getel ( int n ) const
    {
        if ( n < 0 || n >= LIMIT )
            { cout << "\nОшибочный индекс!"; exit ( 1 ); }
        return arr [ n ];
    }
};
////////////////////////////////////
int main ( )
{
    safearray sal;

    // задаем значения элементов
    for ( int j = 0; j < LIMIT; j++ )
        sal.putel ( j, j * 10 );

    // показываем элементы
    for ( j = 0; j < LIMIT; j++ )
    {
        int temp = sal.getel ( j );
        cout << "Элемент " << j << " равен " << temp << endl;
    }

    return 0;
}
```

Данные помещаются в массив с помощью метода `putel()` и выводятся на экран метода `getel()` ,
Безопасность массива реализована с помощью вывода сообщения об ошибке при попытке
использования индекса, не входящего в границы массива.

Контрольные вопросы:

1. Исключение в большинстве случаев возникает из-за:
Программиста, написавшего исходный код приложения;
 - Создателя класса, написавшего его метод;
 - Ошибка выполнения;
 - Сбоя в операционной системе.
2. При работе с механизмом исключений в C++ используются следующие ключевые слова:
_____, _____, _____.
3. Исключения передаются:
 - Из блока-ловушки в блок повторных попыток;
 - Из выражения, создавшего исключительную ситуацию, в блок повторных попыток;
 - Из точки, где возникла ошибка, в блок-ловушку.
 - Из выражения, в котором возникла ошибка, в блок-ловушку.
4. Для следующих ошибок обычно генерируются исключения:
 - Чрезмерное количество данных грозит переполнить массив;
 - Пользователь нажал Ctrl+C для закрытия программы;
 - Скачок напряжения в сети привел к перезагрузке системы;
 - «New» не может зарезервировать необходимый объем памяти.
5. Истинно ли утверждение о том, что программа может продолжить свое выполнение после возникновения исключительной ситуации?
6. Поясните механизм исключений.

Лабораторная работа №7

Тема: Исследование на практике работы шаблонов функций и классов.

Задания

1. Создать шаблон функции, которая находит значение максимального элемента массива. Аргументами функции должны быть имя и размер массива (типа `int`). Создать шаблон функции, осуществляющей сортировку данных массива. Аргументами функции должны быть имя и размер массива (типа `int`). Продемонстрировать работу шаблонов на данных различных типов.
2. В программе определена обобщенная функция `AddOne ()`, у которой один аргумент обобщенного типа, а в качестве результата значение аргумента на единицу. Поэтому тип возвращаемого функцией результата совпадает с типом аргумента. Если аргументом передается число типа `int` или `double`, результатом также является число типа `int` или `double` соответственно, на единицу больше аргумента. Если аргумент функции при вызове – символ (тип `char`), в качестве значения возвращается следующий символ кодовой таблицы.
3. Создать шаблон класса, у которого есть 2 поля в виде одномерных массивов одного размера. Заполнить массивы с помощью генератора случайных чисел. Найти сумму элементов этих массивов. Продемонстрировать работу шаблонов на данных различных типов.
4. Создать шаблон класса, у которого есть 2 поля в виде двумерных массивов. Описать методы, с помощью которых в первом массиве меняются местами две строки массива, а во втором – два столбца массива.

Контрольные вопросы

1. Что такое обобщенная функция?
2. Шабоновые функции позволяют удобным способом создать семейство
а) переменных; б) функций; в) классов; г) программ.
3. Истинно ли утверждение, что шабоновая функция может иметь несколько аргументов?
4. Реальный код шабоновой функции генерируется при:

- а) объявлении функции в исходном коде;
 - б) создании объекта;
 - в) запуске функции из её исходного кода.
5. Что такое обобщенный (шаблонный) класс?
6. В каком случае его имеет смысл применять?
7. Шаблонный класс:
- а) работает с разными типами данных;
 - б) генерирует идентичные объекты;
 - в) генерирует классы с различным числом методов.

Лабораторная работа №8

На основе дисциплины «Алгоритмы и структуры данных» разработать программный продукт с выбором меню для построения структуры «**Очередь**». Программный продукт должен содержать меню для выбора действий

