Last updated: 2019-05-17
vadimov@i.ua

Практическое занятие 14 (семестр2).

Задача 14.1. Переделайте представленную ниже программу так, чтобы в ней отсутствовала инструкция **using namespace std.**

```
// Convert spaces to |s.
#include <iostream>
#include <fstream>
using namespace std;
int main(int argc, char *argv[]) {
    if (argc !=3) {
        cout << " Usage : CONVERT <input> <output>\n";
        return 1;
    }
    ifstream fin(argv[1]);
    ofstream fout(argv[2]);
    if (!fout) {
        cout << "Cannot open output file.\n";
        return 1;
    }
    if (!fin) {
        cout << "Cannot open input file.\n";
        return 1;
    }
    char ch;
    fin.unsetf(ios::skipws); // do not skip spaces
    while (!fin.eof()) {
        fin >> ch;
        if (ch == ' ') ch = '|';
        if (!fin .eof()) fout << ch;
    }
    fin.close();
    fout.close();
    return 0;
}
```

Задача 14.2. Используя класс **strtype** из Example 14.5, создайте функцию преобразования для превращения строки в целое. Функция преобразования должна возвращать длину строки, хранящейся в str. Покажите, что ваша функция преобразования работает.

Задача 14.3. У вас есть класс:

```
class Pwr {
    int base;
    int exp;
public:
    Pwr(int b, int e) { base = b; exp = e; }
    // create conversion to integer here
};
```

Создайте функцию преобразования для превращения объекта типа **Pwr** в целое. Функция должна возвращать результат возведения в степень **base^exp**.

Задача 14.4. Переделайте Example 14.8 так, чтобы на экране отображался тот объект, который осуществляет вывод символов, и тот объект или те объекты, для которых из-за занятости буфера вывод запрещен.

Задача 14.5. Одним из интересных применений статических переменных-членов является хранение информации о количестве объектов класса, существующих в каждый конкретный момент времени. Для этого необходимо увеличивать на единицу статическую переменную-член каждый раз, когда вызывается конструктор класса, и уменьшать на единицу, когда вызывается деструктор. Реализуйте эту схему и продемонстрируйте ее работу.

Задача 14.6. В следующей программе сделана попытка создать простой таймер для измерения временных интервалов. По истечении каждого такого интервала таймер должен подавать звуковой сигнал. В том виде, в котором эта программа здесь представлена, она компилироваться не будет. Найдите и исправьте ошибку.

```
// This program contains an error:
#include <iostream >
using namespace std;
class CountDown {
    int incr;
    int target;
    int current;
public:
    CountDown(int delay, int i=1) {
        target = delay;
        incr = i;
        current = 0;
    }
    bool counting () const {
        current += incr;
        if (current >= target) {
             cout << "\a";
             return false;
        }
        cout << current << " ";
        return true;
    }
};
int main() {
    CountDown obj(100, 2);
    while (obj.counting());
    return 0;
}
```

Задача 14.7. В примере из Unit 14, в котором:

```
     explicit myclass(int x) { a = x; }
     explicit myclass(const char *str) { a = atoi(str); }
```

если спецификатор **explicit** указать только для конструктора **myclass(int)** , можно ли будет выполнить неявное преобразование также и для конструктора **myclass(const char *)**? Попробуйте и посмотрите, что получится. Попробуйте убрать **const** из **myclass(const char *)** . Посмотрите и проанализируйте то, что вам сообщит компилятор.

Задача 14.8. Попытайтесь на конкретных примерах оправдать введение ключевого слова **explicit.** Другими словами, объясните, почему неявное преобразование конструкторов в некоторых случаях может оказаться нежелательным.

Задача 14.9. Используя массивы в качестве объектов ввода/вывода, напишите программу для копирования содержимого одного массива в другой, хотя мы с вами уже знаем, что это не

самый эффективный способ решения подобной задачи.

Задача 14.10. Используя массивы в качестве объектов ввода/вывода, напишите программу для преобразования строки, содержащей значение с плавающей точкой , в число с плавающей точкой (double).

Задача 14.11. Поскольку для конструктора с одним аргументом преобразование типа этого аргумента в тип класса, в котором определен конструктор, происходит автоматически, исчезает ли в этой ситуации необходимость в использовании перегруженного оператора присваивания? Ответ продемонстрируйте на примере.

Задача 14.12. Можно ли в постоянной функции-члене использовать оператор const_cast, чтобы разрешить этой функции-члену модифицировать вызвавший ее объект? Ответ продемонстрируйте на примере.

Задача 14.13. Вопрос для размышления: поскольку библиотека исходного C++ содержится в глобальном пространстве имен и для старых программ на C++ это уже свершившийся факт, какая польза от размещения указанной библиотеки в пространстве имен std "задним числом"?

Задача 14.14. Вопрос для размышления: вернитесь к примерам первых 13-ти лекций. Подумайте о том, в каких из них функции-члены можно было бы сделать постоянными или статическими. Может быть это примеры, в которых определение пространств имен наиболее предпочтительно? Пары примеров будет достаточно.