Last updated: 2019-02-24
vadimov@i.ua

Практическое занятие 3 (семестр2).

Задача 3.1. Программа (см. Unit 3 "An example of class inheritance"), которая определяет общий базовый класс **Fruit**, описывающий некоторые характеристики фруктов. Этот класс наследуется двумя производными классами **Apple** и **Orange**. Эти классы содержат специальную информацию о конкретном фрукте (яблоке или апельсине). Требуется расширить программу, перегрузив методы **seto** и **seta**, таким образом, чтобы их вызовы выглядели вот так:

```
int main() {
    Apple a2;
    Orange o2;
    a2.seta("Jonathan", red, yes, no, yes);
    o2.seto("Valencia", orange, yes, yes, no);
    ...
```

Задача 3.2. Дан следующий базовый класс:

```
class Area {
public:
    double height;
    double width;
}
```

Создайте два производных класса **Rectangle** и **Isosceles**, которые наследуют базовый класс **Area**. Каждый класс должен включать в себя функцию **area()**, которая возвращает площадь соответственно прямоугольника (rectangle) и равнобедренного треугольника (isosceles). Для инициализации переменных **height** и **width** (высота и длина основания, соответственно) используйте конструктор с параметрами. Добавьте производный класс, который наследует класс **Area**, назовите этот класс **Cylinder** и пусть он вычисляет площадь поверхности цилиндра. Эта площадь задается так: 2*pi*R*R + pi*D*height.

Задача 3.3. Используйте объединение (union), чтобы поменять местами старший и младший байты **int** (Намек: наверное ваш компьютер использует 64-битное целое).

Задача 3.4. Компиляторы бывают разные. В зависимости от типа вашего компилятора возможны некоторые ограничения на использование встраиваемых функций. Например, некоторые компиляторы не воспринимают функцию как встраиваемую, если функция является рекурсивной или если она содержит либо статическую **static** переменную, либо любую инструкцию выполнения цикла, либо инструкцию **switch**. Вам необходимо написать программу, при помощи которой выяснить ограничения на использование встраиваемых функций.
Подсказка: целесообразно просмотреть руководство по вашему компилятору, чтобы точно определить ограничения на использование встраиваемых функций.

Задача 3.5. В Unit 2 вы перегружали функцию **abs()** так, чтобы она находила абсолютные значения типа **int**, **long** и **double**. Модифицируйте программу, чтобы эти функции стали

встраиваемыми.

Задача 3.6. Переделайте класс **stack** (см. Unit 2 "A more practical example"), так, чтобы в классе, где это возможно, использовались встраиваемые функции.

Задача 3.7. Создайте класс **Dice**, который содержит закрытую целую переменную. Создайте функцию **roll()**, использующую стандартный генератор случайных чисел **rand()**, для получения чисел от 1 до 6. Сделайте 5 бросков четырех игральных костей. Функция **roll()** должна вывести эти значения на экран.

Задача 3.8. Используя класс **stack** (см. Unit 3 Example 3.7) добавьте в программу функцию **showstack(),** которой в качестве аргумента передается объект типа **stack**. Эта функция должна выводить содержимое стека на экран.

Задача 3.9. Создайте класс **who**. Конструктор **who** должен иметь один символьный аргумент, который будет использоваться для идентификации объекта. При создании объекта конструктор должен выводить на экран сообщение:

```
Constructing who #x
```

где x — идентифицирующий символ, свой для каждого объекта. При удалении объекта на экран должно выводиться сообщение:

```
Destroying who #x
```

где x — снова идентифицирующий символ. Наконец, создайте функцию **make_who()**, которая возвращает объект **who**. Присвойте каждому объекту уникальное имя. Проанализируйте и объясните выводимый на экран результат работы программы.

Задача 3.10. Придумайте пример программы, в которой, как и при неправильном освобождении динамической памяти, возвращать объект из функции было бы также ошибочно.

Задача 3.11. Представьте себе ситуацию, в которой показанные ниже два класса **pr1** и **pr2** используют общий принтер. Для оставшейся части программы необходимо знать, когда принтер занят объектом одного из этих классов. Создайте функцию **inuse()**, которая возвращает **true,** когда принтер занят объектом одного из классов и **false** - в противном случае. Сделайте эту функцию дружественной как классу **pr1,** так и классу **pr2.**

```
class pr1 {
    int printing;
    // ...
public:
    pr1() { printing = 0; }
    void set_print(int status) { printing = status; }
    // ...
};
class pr2 {
    int printing;
    // ...
public:
    void set_print(int status) { printing = status; }
```

```
    // ...
};
```

Задача 3.12. У вас есть следующий класс:

```
class planet {
    int moons;
    double dist_from_sun; // in miles
    double diameter;
    double mass;
public:
    // ....
    double get_miles() { return dist_from_sun; }
};
```

Создайте функцию **light()**, получающую в качестве аргумента объект типа **planet** и возвращающую число секунд, за которые свет достигает планеты. (Предположим, что скорость света равна 186000 миль в секунду и что значение **dist_from_sun,** т. е. расстояние от Солнца, задано в милях.)

Задача 3.13. Используя класс **stack** (см. Unit 2 "A more practical example"), напишите функцию **loadstack()**, которая бы возвращала стек, заполненный буквами алфавита (a-z). В вызывающей программе присвойте этот стек другому объекту и докажите, что и в этом объекте находится алфавит. *(Замечание.* Удостоверьтесь, что длина стека достаточна для хранения алфавита.)

Перегрузите функцию **loadstack()** так, чтобы она получала в качестве аргумента целое число **upper**. В перегруженной версии, если переменная **upper** будет равной 1, загрузите стек символами алфавита в верхнем регистре. В противном случае загрузите его символами алфавита в нижнем регистре.

Задача 3.14. Используя класс **strtype** (см. Unit 3 Example 3.2a), добавьте дружественную функцию, которая получает в качестве аргумента указатель на объект типа **strtype** и возвращает указатель на строку. (Таким образом, функция должна возвращать указатель p). Назовите эту функцию **get_string()**.

Задача 3.15. Проведите эксперимент: если объект производного класса присваивается другому объекту того же производного класса, будут ли также копироваться данные, связанные с базовым классом? Чтобы ответить, воспользуйтесь следующими двумя классами и напишите программу.

```
class base {
    int a;
public:
    void load_a(int n) { a = n; }
    int get_a() { return a; }
};
class derived: public base {
    int b;
public:
    void load_b(int n) { b = n; }
    int get_b() { return b; }
};
```