Last updated: 2019-04-20
vadimov@i.ua

Практическое занятие 11 (семестр2).

Задача 11.1. Создайте манипулятор для вывода чисел в научной нотации с символом **E** в верхнем регистре.

Задача 11.2. Напишите программу для копирования текстового файла. В процессе копирования преобразуйте каждый символ табуляции в соответствующее число пробелов.

Задача 11.3. Напишите программу для поиска в текстовом файле слова, заданного в командной строке. После выполнения программы на экране должно появиться число, обозначающее, сколько раз данное слово найдено в файле. Для простоты считайте следующее: все, что с обеих сторон окружено пробелами, есть слово.

Задача 11.4. Напишите инструкцию, которая устанавливает указатель записи на 81-й байт в файле, связанном с потоком **out**.

Задача 11.5. Напишите программу создания базового класса **Num**. В этом классе должно храниться целое и определяться виртуальная функция **shownum().** Создайте два производных класса **outhex** и **outoct,** которые наследуют класс **Num**. Функция **shownum()** должна быть переопределена в производных классах так, чтобы осуществлять вывод на экран значений, в шестнадцатеричной и восьмеричной системах счисления соответственно.

Задача 11.6. Напишите программу, в которой базовый класс **Dist** используется для хранения в переменной типа **double** расстояния между двумя точками. В классе **Dist** создайте виртуальную функцию **trav_time()**, которая выводит на экран время, необходимое для прохождения этого расстояния с учетом того, что расстояние задано в милях, а скорость равна 60 миль в час. В производном классе **Metric** переопределите функцию **trav_time()** так, чтобы она выводила на экран время, необходимое для прохождения этого расстояния, считая теперь, что расстояние задано в километрах, а скорость равна 100 километров в час.

Задача 11.7. Проведите эксперимент с двумя программами из Example 11.4 и Example 11.5. Попытайтесь создать объект, используя класс **area** из Example 11.4, и проанализируйте сообщение об ошибке. В Example 11.5 попытайтесь удалить переопределение функции **func()** внутри класса **derived2**. Убедитесь, что тогда действительно будет использоваться та версия функции **func(),** переопределение которой находится в классе **derived1**. Что произойдет в Example 11.5 при удалении переопределения функции **func()** из класса **derived1**? Будет ли при этом программа компилироваться и запускаться? Если да, то почему?

Задача 11.8. Программа из Example 11.6 при компиляции при помощи команды **g++ -Wall ex116.cpp** выдает предупреждения. Объясните причину появления этих предупреждений и для их устранения внесите необходимые правки в код данного примера. Совет 1: Функция main() в программе со списками (см. Example 11.6) только иллюстрирует работу классов. Для изучения динамического полиморфизма попробуйте использовать в этой программе следующую функцию main():

```
int main() {
    list *p;
    queue q_ob;
    stack s_ob;
    char ch;
    for (int i=0; i<10; i++) {
        cout << "Stack or Queue? (S/Q):";
        cin >> ch;
        ch = tolower(ch);
        if (ch == 'q')
            p = &q_ob;
        else
            p = &s_ob;
        p->store(i);
    }
    cout << "Enter T to terminate\n";
    for (;;) {
        cout << "Remove from Stack or Queue? (S/Q):";
        cin >> ch;
        ch = tolower(ch);
        if (ch == 't')
            break;
        if (ch == 'q')
            p = &q_ob;
        else
            p = &s_ob;
        cout << p->retrieve() << '\n';
    }
    cout << '\n';
    return 0;
}
```

Добавьте список другого типа к программе из Example 11.6 . Эта версия должна поддерживать отсортированный (в порядке возрастания) список. Назовите список sorted. Совет 2: Обдумайте случаи, в которых следует использовать динамический полиморфизм, чтобы упростить решение разного рода проблем.

Задача 11.9. Расширьте пример со списком, (см. Example 11.6) так, чтобы в нем перегружались операторы + и -. Используйте оператор + для внесения элемента в список, а оператор - для выборки элемента из списка.