Контрольная работа №2.

НИУ ВШЭ, Отделение прикладной математики и информатики, курс «Основы информатики и программирования»

2014 г., Группа 17х/х

Требования.

Нарушение общих требований ведет к снижению оценки. Нарушение критичных требований ведет к серьезному снижению оценки или отклонению решения. Отклонение от указаний (формат вывода, число и тип переменных, требуемый алгоритм) в задании ведет к отклонению решения, либо снижению оценки на усмотрение преподавателя.

Общие требования:

- аккуратная шапка;
- комментарии для каждого цикла, проверки и относительно сложного участка кода;
- правильное выравнивание кода;
- разумные имена переменных и функций;
- отсутствие магических чисел и глобальных переменных;
- **имена функций, параметров, глобальных констант должны быть осмылены**;
- аккуратный вывод (правильно выровненный) и ввод с приглашением на русском;
- **проверка корректности вводимых значений**;
- все другие требования к оформлению, о которых говорилось на лекциях и семинарах!

Критичные требования:

- программа не компилируется;
- программа завершается аварийно (даже если часть или все результаты выведены в консоль);
- программа зацикливается;

- при запуске нет приглашения для ввода (пользователь должен сам догадаться что нужно вводить).

Пусть дана фиксированная часть программы, которую предстоит написать, а именно:

```
const int len = 20;
const int numel = 200;
struct Primer {
     int num1, num2;
     char * slovo;
};
```

В функции **main( )** используется следующее объявление объектов (обязательное):

```
int main( )
{
     vector<Primer> vect;
     const char * search = "at";     // строка для поиска (задание 3).
     const char * data = "d:\\kontr.txt";
                              // полное имя файла с данными для обработки.
     char input_str[numel];   // сюда последовательно помещаются считываемые из
файла строки.
     ifstream finp(data);     // открытие входного потокового объекта.
     ... ... ... ... ...      // другие объявления переменных...
}
```

Написать полную C++ программу, реализующую следующие действия:

1) Первая пользовательская функция

прототип: `void readfile(ifstream& fin, vector<Primer>& vec, char * pch);`
параметры: *ссылка на входной поток, ссылка на вектор структур, строка* (см. `input_str[ ]` в функции `main( )`).

Из внешнего текстового файла данные читаются до достижения конца файла. Строки текстового файла содержат два целочисленных значения, которые разделены одним пробелом, и слова (чередование строк – см. файл *kontr.txt*). Читаемые из файла данные помещаются в вектор:

- целые числа сохраняются в полях `num1` и `num2;`
- слова хранятся в поле `slovo`. (После прочтения из файла слова в буфер, вычисляется ее длина и далее динамически выделяется С-строка под это слово и указатель на строку сохраняется в поле `slovo.` Размер динамически создаваемой С-строки должен быть равен длине считанного слова, а не просто большой константе.)
  В консольное окно выводится сообщение:

```
Входной файл успешно прочитан, и данные размещены в векторе
Слова вектора: act, matrix ...

Число элементов вектора равняется ____
```

Из main функция readfile вызывается с параметрами: finp, vect и input_str. После выхода из функции `readf( )` вызывается функция `calcpr( ).`

2) Вторая пользовательская функция

прототип: `void calcpr(vector<Primer>& vec);`

параметры: *ссылка на вектор структур*.

- *см. выше по тексту*, выполняется печать (в консольное окно) значений всех элементов вектора, т.е. всех строк и всех числовых данных в следующем формате (аккуратное выравнивание обязательно!):

```
Полученные из внешнего файла данные помещены в вектор:
#1: num1 = ___       num2 = ___    slovo ---> operation       [ w.a. = ____ ]
#2: num1 = ___       num2 = ___    slovo ---> matrix          [ w.a. = ____ ]
#3: ....
```

В последнем столбце выводятся значения взвешенных средних (**w.a.**), которые вычисляются следующим образом: $\left[\sum_{k=0}^{L_i-1} w_k \cdot C(s_k)\right] \Big/ \sum_{k=0}^{L_i-1} w_k$, где L_i - длина i-той строки, $C(s_k)$ - ASCII-код k-того символа строки, а w_k – вес k-того символа (буквы) строки. *Все строки состоят только из прописных букв латинского алфавита*. Каждой букве присваивается вес w_k следующим образом: `'a'` имеет вес **1**, `'b'` - **2**, `'c'` - **3** и т.д. Вычисленные вещественные значения выводятся с 2-мя знаками после запятой.

Например:

```
#1: num1 = 12        num2 = 46     slovo ---> act             [ w.a. = ____ ]
```

Т.е. рассматриваемое слово "act". $L_i = 3$. Для первого символа $s_1 =$ "a", $C(s_1) =$ 97, $w_1 = 1$. $s_2 =$ "c", $C(s_2) =$ 99, $w_2 = 3$. $s_3 = t$, $C(s_3) = 116$, $w_3 = 20$.
(1*97+3*99+20*116)/(1+3+20)=113.08.

3) Третья пользовательская функция

прототип: `int count(vector<Primer>& vec, char * pstr);`

параметры: *ссылка на вектор структур; искомая подстрока*.

Для каждого элемента вектора, у которых значение `num1+num2` нечетно, определяется общее число вхождений подстроки параметра `pstr` в поле `slovo`. На экран выводится текущее слово и число вхождений:

```
слово: matrix       подстрока: at       число вхождений: 1
слово: attenuate    подстрока: at       число вхождений: 2
...
```

Функция возвращает число (общее) таких вхождений во всем векторе.

В функции `main()`, уже после выхода из функции `count()`, выводится следующее:

```
общее число вхождений строки "at" в векторе равняется ___
```

4) Четвертая пользовательская функция:

прототип: `void FunnyFunction();`

Пользователь вводит N и M. Функция создает динамический двухмерный массив int NxM и заполняет его случайными числами от 1 до 99. Аккуратно выводит на экран. После этого числа обнуляются в шахматном порядке и массив сново выводится на экран.

Например:

```
 1  5 21  3 77
55  4  3 23 21
 5 24 32 23 21
15  4  3  3 21

 0  5  0  3  0
55  0  3  0 21
 0 24  0 23  0
15  0  3  0 21
```

В программе освобождается вся память, выделенная ранее под динамические массивы.