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:

```
struct Train
{
      char name[200]; // название поезда
      int size;       // количество вагонов
      int * wagwts;   // вес вагонов
      int * waglen;   // длина вагонов
      int weight;     // вес поезда
      int length;     // длина поезда
}
```

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

```
int main( )
{
     vector<Train> vcTr;
     const char * data = "d:\\trains.txt";
                              // полное имя файла с данными для обработки.
     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<Train>& vec);`

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. Формат файла следующий (**очень важно хорошо понимать формат файла**):

<название поезда> - <число вагонов> : <длина вагона> <вес вагона> <длина ваг.> <вес ваг.> …

Например:

Small - 3 : 1 10 1 10 1 10

Это поезд **Small** с **3** вагонами, каждый из которых имеет длину **1** и вес **10**. (см. файл *trains.txt*). Читаемые из файла данные помещаются в вектор структур. Для каждого поезда в структуре **Train**:

- Название поезда сохранятеся в поле **name**;
- Число вагонов сохраняется в поле **size**;
- Создаются динамические массивы **waglen** и **wagwts** для хранения длин и весов вагонов;
- Поле **weight** рассчитывается как сумма весов вагонов;
- Поле **length** рассчитывается как сумма длин вагонов;

  В консольное окно выводится сообщение:

```
Входной файл успешно прочитан, и данные размещены в векторе
Имена вагонов: Small, Red Arrow,  ...
Число элементов вектора равняется ____
```

Из main функция readfile вызывается с параметрами: finp, vcTr.

2) Вторая пользовательская функция

прототип: `void masscentre(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:

```
Информация о парке поездов:
(Название поезда, (#,длина,вес) вагонов, длина и вес поезда )
#1: Small      (1,1,10) (2,1,10) (3,1,10) Длина: 3 Вес: 30
#2: Red Arrow  (1,5,37) (2,5,73) (3,7,192) (4,7,80) (5,5,19) Длина: 29 Вес:...
#3: ....
```

Пользователь вводит номер поезда (должна быть проверка, что поезд с таким номером имеется).
Найти вагон, на который приходится упрощенный центр тяжести поезда. Т.е. вагон, у которого разница между весом вагонов слева и весом вагонов справа минимальна.
Вывести название поезда, номер вагона центра тяжести, суммы вагонов слева и справа от этого вагона, разницу этих сумм.

Например (пользователь ввел 6):

```
#6: Поезд: Small   Центр тяжести: Вагон #3  Вес слева: 50 Вес справа: 50
Разница: 0
```

3) Третья пользовательская функция

прототип: `int sort (vector<Primer>& vec, int index);`

параметры: *ссылка на вектор структур; номер элемента*.

Отсортировать вектор поездов по длине поезда (поле length).
Найти поезд у которого разница веса самого тяжелого вагона и самого легкого максимальна. Вывести на экран название поезда, номер и вес самого тяжелого и самого легкого вагонов, и разницу весов.

Поезд: Long long train Тяжелый – (#7,Вес:333) Легкий – (#1,Вес:3)
Разница: 330 тонн

Найти поезд у которого самый маленький номер вагона, на который приходится центр тяжести.

Поезд: Small Центр тяжести: Вагон #2

В функции **main** вывести отсортированный список на экран и в файл.

4) Четвертая пользовательская функция:

прототип: `void FunnyFunction();`

Пользователь вводит N и M. Создается целочисленный массив NxM и его элементы заполняются концентрическими кольцами, как показано на примере:

```
11111111
12222221
```

12333321
12333321
12222221
11111111

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