Задача 8. Выравнивание структур

Источник:	повышенной сложности
Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по времени:	1 секунда
Ограничение по памяти:	разумное

Петя изучает язык C, и только что дошёл до структур. Его особенно заинтересовал тот факт, что поля структуры не всегда размещаются в памяти подряд: иногда между соседними полями появляются дополнительные «пустоты» (padding). Из-за этого получается, что размер структуры может зависеть от того, в каком порядке программист перечислит поля структуры! Очень интересна в этом плане, например, вот эта статья.

Дано описание структуры на языке C. В первой строке написано ключевое слово `struct` и открывающая фигурная скобка. В последней строке записана закрывающая фигурная скобка и точка с запятой. Каждая строка между ними описывает ровно одно поле структуры.

В описании поля сначала стоит тип поля, затем имя поля, и наконец точка с запятой. Тип поля отделён от имени поля хотя бы одним пробелом. Имя поля — это непустая строка, состоящая из латинских букв любого регистра, цифр и символов подчёркивания, причём имя точно **не** начинается с цифры.

В качестве типа поля может быть указан либо примитивный тип, либо указатель на примитивный тип. Во входных данных могут быть только следующие примитивные типы: `char`, `short`, `int`, `long`, `float`, `int64_t` и `double`. Указатель может быть любого порядка (т.е. двойной указатель, тройной указатель и т.п.).

В данной задаче будем считать, что структура размещается в памяти следующим образом (в реальности правила размещения определяются компилятором). Первое поле структуры помещается по адресу с максимальным выравниванием: будем считать, что его адрес делится нацело на 8. Каждое следующее поле размещается по такому адресу, что:

1. оно расположено после предыдущего,
2. оно корректно выровнено, то есть его адрес делится на его размер, и
3. его адрес минимален при выполнении первых двух условий.

При этом между соседними полями может появиться пустое место (padding) размером от 1 до 7 байт.

После последнего поля также может быть добавлено несколько байт пустоты. Это необходимо, чтобы можно было хранить массив таких структур, то есть располагать в памяти много одинаковых структур друг за другом. В конце структуры добавляется минимальное количество байт пустоты, так чтобы в массиве любой длины все поля всех структур были корректно выровнены. При этом размер всей структуры равен сумме размеров всех полей и размеров всех имеющихся в структуре пустот.

Петя может переставлять местами поля структуры. Он хочет узнать, какой при этом может получиться минимальный размер структуры, и какой максимальный размер. Как известно, размер также зависит от модели данных. Петя хочет узнать ответ для всех популярных моделей.

Формат входных данных

Во входном файле описана одна структура согласно указанным в условии правилам. В любом месте описания может быть добавлено любое количество пробелов (ASCII 32), если их добавление не разрывает на части название примитивного типа, имя поля или ключевое слово.

Гарантируется, что в структуре от 1 до 100 полей, а общее количество символов в описании не превышает 5 000. Имена всех полей структуры отличаются друг от друга и не совпадают с ключевыми словами языка C.

Формат выходных данных

В выходном файле должно быть четыре строки, по одной строке на модель данных. В каждой строке нужно записать через пробел два целых числа: минимально возможный размер структуры и максимально возможный размер.

В первой строке должен быть ответ для модели данных `LP32`, во второй — для модели `ILP32`, в третьей — для модели `LLP64` и в четвёртой — для модели `LP64`. Информацию о том, сколько байтов занимает каждый тип на каждой модели данных, можно найти на странице: `https://ru.cppreference.com/w/cpp/language/types`

Пример

input.txt	output.txt

`struct {`
`  int x;`
`  int64_t* * Y;`
`  char  _temp1;`
`} ;`
	8 12
	12 12
	16 24
	16 24

Пояснение к примеру

Рассмотрим структуру в модели принятой на Win64 модели данных `LLP64`.

Поле `x` имеет размер 4 байта, поле `Y` размера 8 байт, а поле `_temp1` занимает 1 байт.

Если не менять порядок полей, тогда поле `x` будет занимать байты 0-3. Придётся добавить 4 байта пустоты, чтобы поле `Y` было выровнено по 8 байтам, заняв байты 8-15. Поле `_temp1` будет иметь адрес 16 и занимать один байт, но после него нужно добавить ещё 7 байтов пустоты. Без этого поле `Y` второй структуры в массиве не может быть корректно выровнено. Получается общий размер 24 байта.

Если выбирать порядок полей, то размер 24 байта получается только когда поле `Y` находится между другими двумя полями. Так получается в 2 способах среди всех 6 спопобов выбора порядка. В остальных случаях размер структуры равен 16 байтам.