Задача 10. Безумная линковка

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

Код задачи: `2/A_crazylink`

Эта задача стремится показать, что линкер — это слепое чудовище! В закрытой для вас единице трансляции есть N символов с известными именами, но с неизвестными типами. Тип этих символов сообщается только во время запуска программы (задан во входном файле). Вам нужно научиться оперировать с этими символами в таких условиях.

На каждом тесте ваша программа будет собираться заново. К списку отправленных вами исходных файлов будет добавляться файл `symbols.c`, в котором определено ровно 10 символов с именами от `symbol0` до `symbol9`. Далее эта программа должна скомпилироваться и слинковаться в исполняемый файл. Затем этот исполняемый файл запускается, читает из входного файла типы символов и запросы, и выводит ответы в выходной файл.

Формат входных данных

В первой строке входного файла записано целых числа: N — сколько символов используется в тесте и M — количество запросов ($1 \leqslant N \leqslant 10$, $1 \leqslant M \leqslant 100$). Среди определённых в `symbols.c` десяти символов в тесте используются только первые N штук, остальные не упоминаются.

В следующих N строках задаются типы символов, по одному символу в строке. Символы описываются строго в порядке от 0-ого до $(N-1)$-ого. Типы описываются в формате языка C. Если символ является переменной, то записано определение переменной, а если функцией — то прототип функции.

Возможны следующие типы:

- переменная типа `int` или `double`;
- переменная-указатель типа `int*` или `double*`;
- функция с количеством аргументов от нуля до двух; в этом случае каждый аргумент имеет тип `int` или `double`;
- переменная типа «указатель на функцию» с ограничениями как в предыдущем пункте;

Для вашего удобства при форматировании типов жёстко соблюдается единый стиль. У параметров функции и указателя на функцию отсутствуют имена. Если функция или указатель на функцию не принимает параметров, то в скобках нет ничего (в частности, не пишется `void`). Все переменные/функции имеют имя, совпадающее с именем соответствующего символа. В обычном указателе звёздочка ставится вплотную к имени переменной. Одиночный пробел ставится после запятой, а также после основной части типа переменной или возвращаемого типа функции. Других пробелов нет.

В последних M строках описываются запросы, по одному в строке. Каждый запрос — это последовательноть целых чисел длиной от одного до трёх. Все числа последовательности лежат в диапазоне от 0 до $N-1$ включительно.

Бывает два типа запросов:

1. Найти значение переменной: в этом случае задаётся номер символа-переменной, значение которого надо вывести.

2. Вычислить значение функции: в этом случае сначала задаётся номер символа-функции, которую надо вызвать, а потом номера символов-переменных, которые надо передать в функцию в качестве аргументов.

Если символ является указателем на переменную или указателем на функцию, то надо предварительно разыменовать эту переменную. Гарантируется, что все запросы сформированы корректно, количество аргументов и их типы подходят (без преобразований).

Формат выходных данных

Нужно вывести M строк, в каждой строке ответ на соответствующий запрос. Ответом считается результат вычисления, который получается или типа `int`, или типа `double`. Значения типа `double` рекомендуется распечатывать с максимальной точностью.

Запросы нужно вычислять в порядке их описания: некоторые функции могут иметь побочные эффекты.

Пример

Содержимое файла `symbols.c` для примера:

```
int rnd(double x) { return (int)x; }
int symbol0 = 42;
double symbol1 = 5.7;
int* symbol2 = &symbol0;
int symbol3() { return -1; }
int symbol4(int a) { return a * 4; }
int symbol5(int a, int b) { return a - b + 7; }
double symbol6(double a) { return a * a - 3.5; }
int (*symbol7)(double a) = &rnd;
int symbol8 = 23;
int symbol9;
```

input.txt	output.txt

9 10
int symbol0;
double symbol1;
int *symbol2;
int symbol3();
int symbol4(int);
int symbol5(int, int);
double symbol6(double);
int (*symbol7)(double);
int symbol8;
0
1
2
3
4 0
8
5 0 8
6 1
7 1
5 0 0
	42
	5.700000000000000
	42
	-1
	168
	23
	26
	28.990000000000002
	5
	7

Комментарий

Данная задача крайне далека от реальности. В реальности если кто-то полагается на несовпадающие типы при линковке, надо сразу сослать его в Сибирь!

В зависимости от способа реализации в решении может получиться очень много кода. Рекомендуется подумать, как минимизировать объём кода и вероятность облажаться. У меня получилось 163 строки.

Учтите, что к этой задаче подключены компиляторы GCC, Clang и TCC — решение должно работать на них всех.