```
1       /*
          MSP, 30.10.07, 13:20

          Вопрос.
            Что такое стрелка: -> ?

          Объяснение.
          Part 1.

          Ниже читаем ответ.

        */

9       # include <stdio.h>
10      # include <conio.h>

11      // Ты знаешь, что существуют базовые ПРОСТЫЕ типы данных, такие как:
12      //   int,
13      //   int *,
14      //   char,
15      //   char *,
16      //   float,
17      //   float *
18      // и так далее, что есть
19      //   unsigned char,
20      // что есть
21      //   long,
22      //   unsigned long
23      // то есть существуют БАЗОВЫЕ ТИПЫ языка от его рождения,
24      // они зашиты в самом языке, это такие ПРОСТЫЕ типы, из которых
25      // могут собираться по кирпичикам СЛОЖНЫЕ типа данных.

26      // Как собрать СЛОЖНЫЙ тип?
27      // Очень просто, надо написать служебное слово class и в теле класса
28      // прописать набор из простых типов данных.

29      class spisok
30      {
31        public:
32          spisok * L; // указатель на элемент списка слева
33          int data; // данные элемента списка
34          spisok * R; // указатель на элемент списка справа
35      }; // это правильно, здесь НУЖНО ставить ;

36      // Здесь произошло описание НОВОГО СЛОЖНОГО типа данных.
37      // Теперь main () может работать не только с ПРОСТЫМИ типами данных,
38      // которые ты уже знаешь, но и со СЛОЖНЫМ типом данных -
39      // со структурой, или с классом, и ИМЯ у этого нового СЛОЖНОГО типа
40      // данных
41      //   spisok
```

```
42      // Итак, что же произошло в коде выше?
43      // А вот что.
44      // Объявлен НОВЫЙ тип данных - spisok - он СЛОЖНЫЙ, структурный,
45      // классовый тип.

46      // Что же это за тип данных spisok?
47      // Его придумал САМ программимст.
48      // Зачем?
49      // чтобы решаить задачу на КОЛЬЦО-ДЕК.

50      // Новая, сложная структура данных, придуманная самим программмстом,
51      // имеет имя-идентификатор spisok. Этот тип данных объдиняет в себе
52      // три типа данных:
53      //   1. указатель
54      //   2. int
55      //   3. указатель

56      // Если появился НОВЫЙ тип данных, значит, можно порождать
57      // ПЕРЕМЕННЫЕ такого нового типа.
58      // Как это сделать, а точно также, как и порождать переменные
59      // простых типов, вот так:
60      //
61      //   spisok element;

62      int main ()
63      {
64        spisok element; // создана новая переменная element

65        return 0;
66      }

67      // Теперь в памяти PC создана новая переменная с
68      // идентификатором-именем element.
69      //
70      // Это значит, что OS разрешила по какому-то адресу A1 хранить
71      // один за другим три переменных: Указатель, int, указатель.

72      // Давай, так, указатель я буду обозначать как U

73      // Тогда, начиная с адреса A1 хранятся ТРИ переменных:
74      //   U1, int, U2,
75      // на указатель U1 израсходуется 4 байта,
76      // на int -                      2 байта,
77      // на U2 -                       4 байта.
78      // Таким образом на хранение переменной
79      //   element
80      // OS отдаст 10 байтов.

81      // Итак, с адреса A1 находится информация о составе переменной
82      // element. И длина этой переменной 10 bytes.
```

```
83      // U1 расположен A1..(A1 + 3)
84      // int:          (A1 + 4)..(A1 + 6)
85      // U2:           (A1 + 7)..(A1 + 9)

86      // Но что важно понимать, так это то, что по этим адресам находится
87      // еще неопределенная программистov информация, так называемый
88      // "МУСОР".
89      // Почему?
90      // Потому что еще не было ИНИЦИАЛИЗАЦИИ переменной element.
91      // То есть не было назначено первоначальное значение этой
92      // переменной.

93      // element - это объединение TPEX переменных.
94      // Что является значением U1?
95      //   -- Адрес
96      // переменной int?
97      //   -- целое число
98      // переменной U2?
99      //   -- Адрес

100     // U1 расположен A1..(A1 + 3), но никто еще эти ячейки
101     // памяти не заполнил, но по этим адресам уже работали другие
102     // программы и там "намусорили". В этих 4-х ячейках записана
103     // какая-то информация, но не мы ее туда записали, а
104     // другие программы,
105     // в этих 4-х ячейках записан какой-то МУСОРНЫЙ АДРЕС - значение
106     // переменной
107     //   spisok * L
108     // L имеет значение, например, A867

109     // int:          (A1 + 4)..(A1 + 6), в этих ячейках тоже мусор,
110     // там находится значение переменной
111     //   int data,
112     // например, там может быть записано число -23456.
113     //
114     // U2:           (A1 + 7)..(A1 + 9), но никто еще эти ячейки
115     // памяти не заполнил, но по этим адресам уже работали другие
116     // программы и там намусорили. В этих 4-х ячейках записана какая-то
117     // информация, но не мы ее туда записали, а другие программы,
118     // в этих 4-х ячейках записан какой-то МУСОРНЫЙ АДРЕС - значение
119     // переменной
120     //   spisok * R
121     // L имеет значение, например, A209

122     // Тогда как же ПРОИНИЦАЛИЗИРОВАТЬ переменную element?

123     // 1. Попросить OS дать адрес для хранения ЗНАЧЕНИЯ указателя
124     //      spisok * L,
125     // 2. Задать с клавиатуры значение переменной
126     //      int data,
```

```
127     // 3. Посросить OS дать адрес для хранения ЗНАЧЕНИЯ указателя
128     //      spisok * R.

129     // Значения указателей дает операция
130     //   new

131     // Но у класса spisok есть очень трудно понимаемая ообенность,
132     // у него указатели ссылаются на саму структуру-класс, это
133     // так называемый рекурсивный тип данных. Но ничего сложного в этом
134     // нет. Потому что указатели на структуру просто будут просить
135     // у OS 10 bytes на расположение новой переменной типа spisok.

136     // Теперь смотрите дальнейшее объяснение, part 2.
```

Listing данной задачи опубликован в сети Internet по адресу
http://www.Best-Listing.ru/color-3-task-18.html

Sergey Mitrofanov, 30.10.13, 21:23
E-mail: infostar @ mail.ru