

```

1      /*
      File:      dv_spis1.cpp

      Compiler:  VC++ 3.1

      Задача.   Сформировать двусвязный список из ТРЕХ элементов.

      Решение.  MSP,
                CNIT "North Star",
                02.11.07, 10:15-12:16 (1h)

      Learn:    Сначала идеально разобрать уроки
                strelka1.cpp,
                strelka2.cpp,
                strelka3.cpp.
      */

13     # include <stdio.h>
14     # include <conio.h>

15     // список состоит из элементов
16     // структура одного элемента двусвязного списка
17     class element
18     {
19     public:
20         element * L; // указатель на предыдущий элемент списка
21         int data; // ключ элемента списка
22         element * R; // указатель на следующий элемент списка
23     };

24     int main ()
25     {
26         clrscr ();

27         element * element_L; // предыдущий элемент списка
28         element * center; // центральный элемент списка
29         element * element_R; // следующий элемент списка

30         // выделим память под элементы списка
31         element_L = new element; // A1
32         center = new element; // A2
33         element_R = new element; // A3
34         // в element_L записан адрес A1, который определила OS
35         // в center записан адрес A2, который определила OS
36         // в element_R записан адрес A3, который определила OS

37         // Обозначим поля для element_L так:
38         // element * L; - UL
39         // int data; - D

```

```

40 // element * R; - UR

41 // Тогда element_L это:
42 // UL D UR

43 // | A1 - такой трехэтажный символ означает, что
44 // | указатель имеет значение A1 и стрелочка
45 // | на начало элемента списка
46 // V

47 // В этой точке кода в памяти (схематически) будет вот что:

48 // | A1 | A2 | A3
49 // | element_L | center | element_R
50 // | | |
51 // v v V
52 // UL D UR UL D UR UL D UR

53 // причем все значения UL, D и UR "мусорные", в них записаны
54 // неопределенные программистом значения, в них остатки работы
55 // других, чужих программа, работавших на PC до запуска этой
56 // программы.

57 // мусор будем обозначать как M

58 // | A1 | A2 | A3
59 // | element_L | center | element_R
60 // | | |
61 // v v V
62 // UL D UR UL D UR UL D UR
63 // M M M M M M M M M

64 // Нужно понимать, что OS выделяет память динамически, и совсем
65 // необязательно, чтобы адреса A1, A2, A3 шли по-порядку,
66 // один за другим, в памяти PC они могут идти в любом порядке,
67 // но на каждый элемент списка будет тратиться 10 байт.

68 // Теперь определим все поля элементов, запрограммируем их
69 // значения.

70 // Заполняем значения полей СЛЕВА НАПРАВО.
71 // Заполним element_L.
72 // element_L самый левый, левее него ничего нет, нет ни одного
73 // элемента, раз так, то его поле UL выставим в NULL.
74 // В поле D запишем 10, как раньше.
75 // В поле UR запишем A2, так как после element_L идет center,
76 // у которого адрес A2

77 // Получим в памяти такую картинку:

78 // | A1 | A2 | A3

```

```

79     // | element_L           | center           | element_R
80     // |                     |                   |
81     // v                     v                       V
82     // UL  D  UR           UL  D  UR           UL  D  UR
83     // NULL 10  A2        M   M   M           M   M   M

84     // Вот код этого действия:
85     element_L -> L = NULL;
86     element_L -> data = 10;
87     element_L -> R = center;

88     // Ниже картинка, которая нам нужна, к чему мы стремимся, чтобы
89     // на самом деле был список элементов и, чтобы он был, связным.
90     // Так как указателей ДВА, то у нас должен получиться двусвязный
91     // список из ТРЕХ элементов:

92     // | A1                 | A2                 | A3
93     // | element_L         | center             | element_R
94     // |                     |                   |
95     // v                     v                       V
96     // UL  D  UR           UL  D  UR           UL  D  UR
97     // NULL 10  A2        A1  20  A3           A2  30  NULL

98     // вот код организации связки элементов списка по его указателям
99     center -> L = element_L;
100    center -> data = 20;
101    center -> R = element_R;

102    element_R -> L = center;
103    element_R -> data = 30;
104    element_R -> R = NULL;

105    // Мы программируем на C++, который разрешает объявлять новые
106    // переменные в любом месте кода.

107    // А теперь все три адреса элементов списка поместим в массив:
108    element * A [3];

109    // заполним массив адресами
110    A [0] = element_L;
111    A [1] = center;
112    A [2] = element_R;

113    // Пробежимся по списку и напечатаем значения ключей
114    int i; // номер элемента двусвязного списка

115    printf ("Двусвязный список через массив A []:\n");
116    for (i = 0; i < 3; i ++)
117        printf ("%i ", A [i] -> data);
118    printf ("\n");

```

```

119 // а теперь пробежимся по левым указателям, и напечатаем список
120 // СПРАВА НАЛЕВО
121 element * tek; // текущий указатель
122 int key; // ключ-данные

123 printf ("\nДвусвязный список СПРАВА-НАЛЕВО через tek:\n");
124 tek = element_R;
125 while (tek != NULL)
126 {
127     key = tek -> data;
128     printf ("%i ", key);
129     tek = tek -> L;
130 }
131 printf ("\n");

132 // печать списка СЛЕВА-НАПРАВО без использования массива A []
133 printf ("\nДвусвязный список СЛЕВА-НАПРАВО через tek:\n");
134 tek = element_L;
135 while (tek != NULL)
136 {
137     key = tek -> data;
138     printf ("%i ", key);
139     tek = tek -> R;
140 }
141 printf ("\n");

142 // удалим из списка центральный элемент

143 // Нам дано:

144 // | A1                | A2                | A3
145 // | element_L        | center            | element_R
146 // |                  |                   |
147 // | v                | v                 | V
148 // | UL  D  UR        | UL  D  UR        | UL  D  UR
149 // | NULL 10  A2      | A1  20  A3       | A2  30  NULL

150 // Надо получить:

151 // | A1                | A3
152 // | element_L        | element_R
153 // |                  |
154 // | v                | v
155 // | UL  D  UR        | UL  D  UR
156 // | NULL 10  A3      | A1  30  NULL

157 // Вот код удаления элемента

158 element_L -> R = element_R;

```

```

159     element_R -> L = element_L;
160     delete center;

161     // печать списка СЛЕВА-НАПРАВО после удаления центрального
162     // элемента:
163     printf (
164         "\nДвусвязный список СЛЕВА-НАПРАВО после удаления "
165         "центрального элемента:\n"
166     );
167     tek = element_L;
168     while (tek != NULL)
169     {
170         key = tek -> data;
171         printf ("%i ", key);
172         tek = tek -> R;
173     }
174     printf ("\n");

175     // нельзя забывать о возврате занятой памяти
176     // в распоряжение OS
177     delete element_L;
178     delete element_R;
179     printf ("\nПамять OS возвращена.");

180     getch ();

181     return 0;
182 }

```

Listing данной задачи опубликован в сети Internet по адресу  
<http://www.Best-Listing.ru/color-3-task-12.html>

Sergey Mitrofanov, 30.10.13, 20:52

E-mail: [infostar@mail.ru](mailto:infostar@mail.ru)

© <http://www.Best-Listing.ru/>, 2006–2013