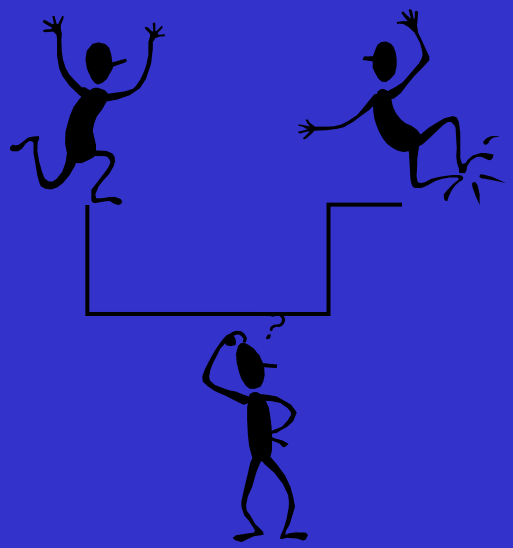
Тема 9. Динамические структуры данных

Статические и динамические структуры данных

Динамические структуры данных

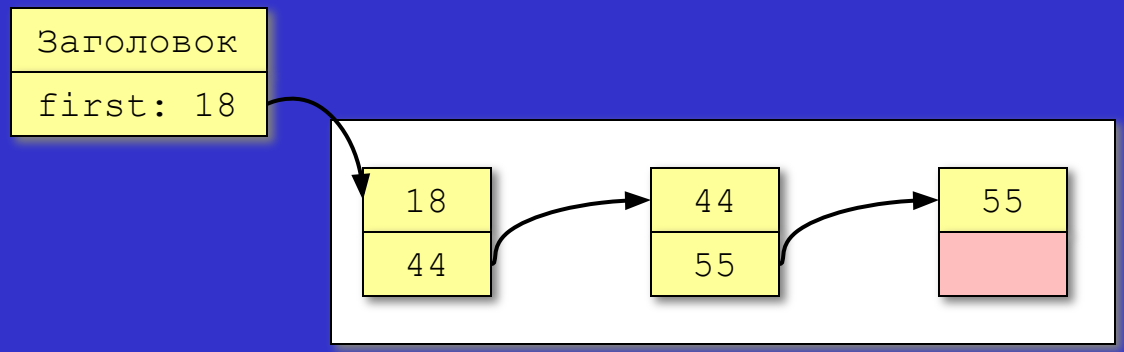


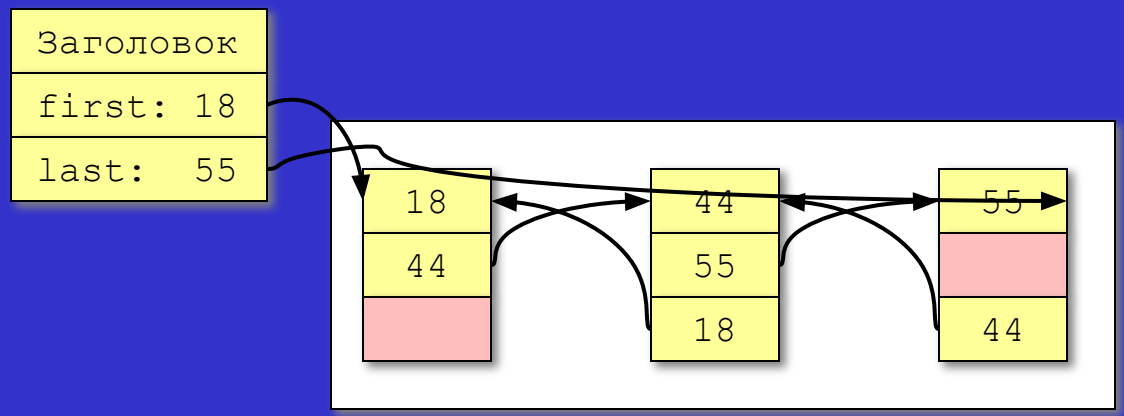
"Я женился на вдове, у которой была взрослая дочь. Мой отец, часто навещавший нас, влюбился в мою падчерицу и женился на ней. Следовательно, мой отец стал моим зятем, а моя падчерица - моей мачехой. Спустя несколько месяцев моя жена родила сына, который стал шурином моего отца и одновременно моим дядей. У жены моего отца, то есть моей падчерицы, тоже родился сын. Таким образом, у меня появился брат и одновременно внук. Моя жена является моей бабушкой, так как она мать моей мачехи. Следовательно, я муж моей жены и одновременно ее внук, другими словами - ***я собственный дедушка****."*

Из одной цюрихской газеты, июль 1922 года.

Линейные списки

Код	Фамилия
01	Александров
02	Алексеев
03	Антонов
...	...
18	Иванов
19	Михайлов
...	...
44	Петров
...	...
55	Сидоров
56	Тимофеев



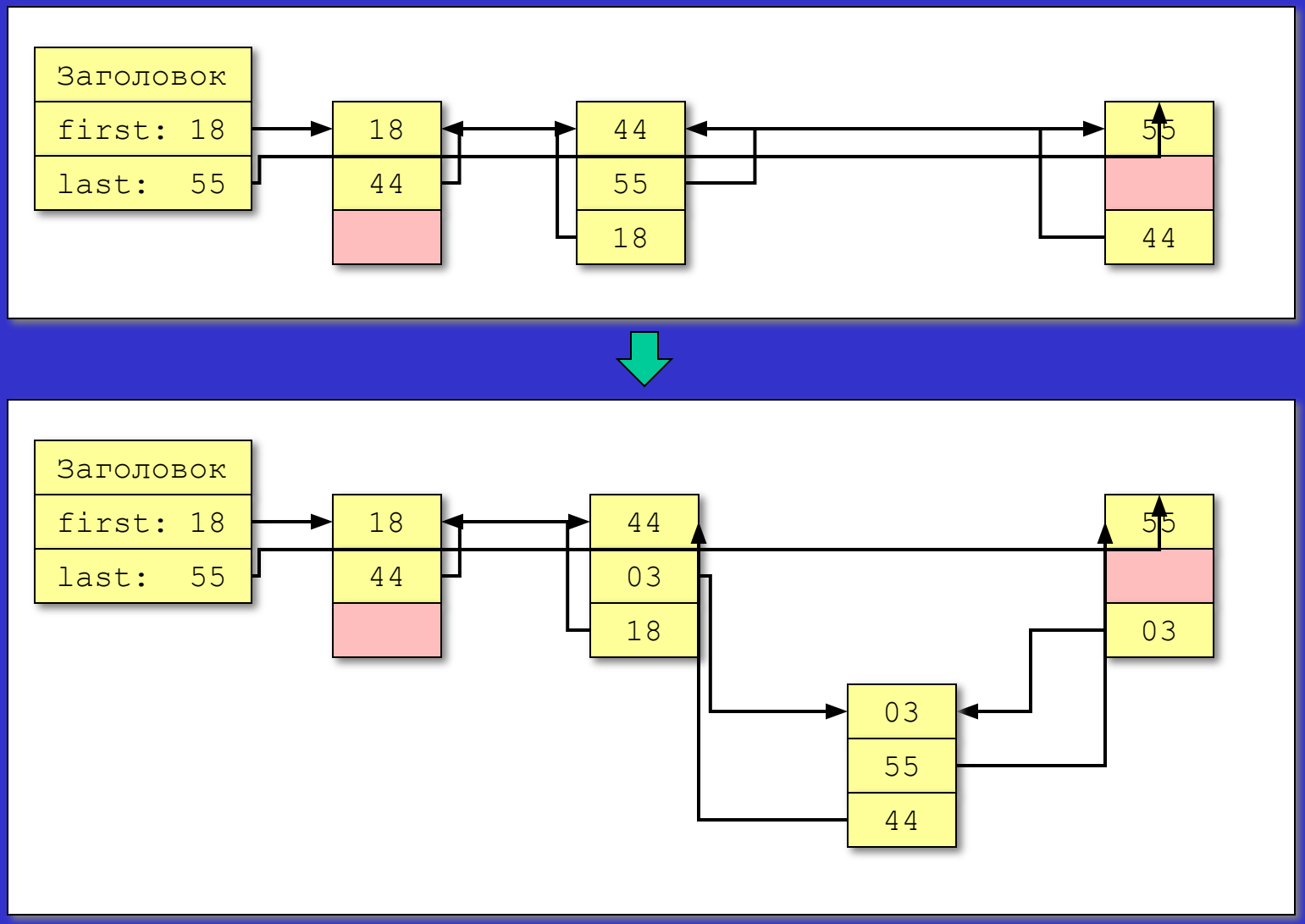


Иванов
Петров
Сидоров

Пример обхода списка

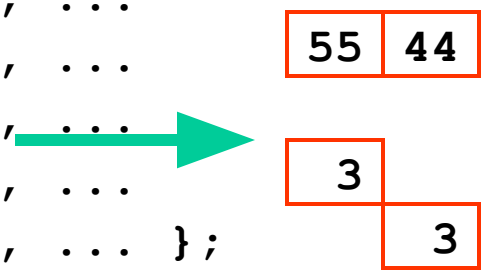
```
typedef struct {short code; char name[24]; short next;} PERS;

PERS pers[] = {{ 1, "Александров",  0}, ...
               {18, "Иванов",      44}, ...
               {44, "Петров",      55}, ...
               {55, "Сидоров",      0}, ... };

   int first = 18;

   while(first)
   {
      int i = first-1;
      printf(pers[i].name);
      first = pers[i].next;
   }
```

Включение в список



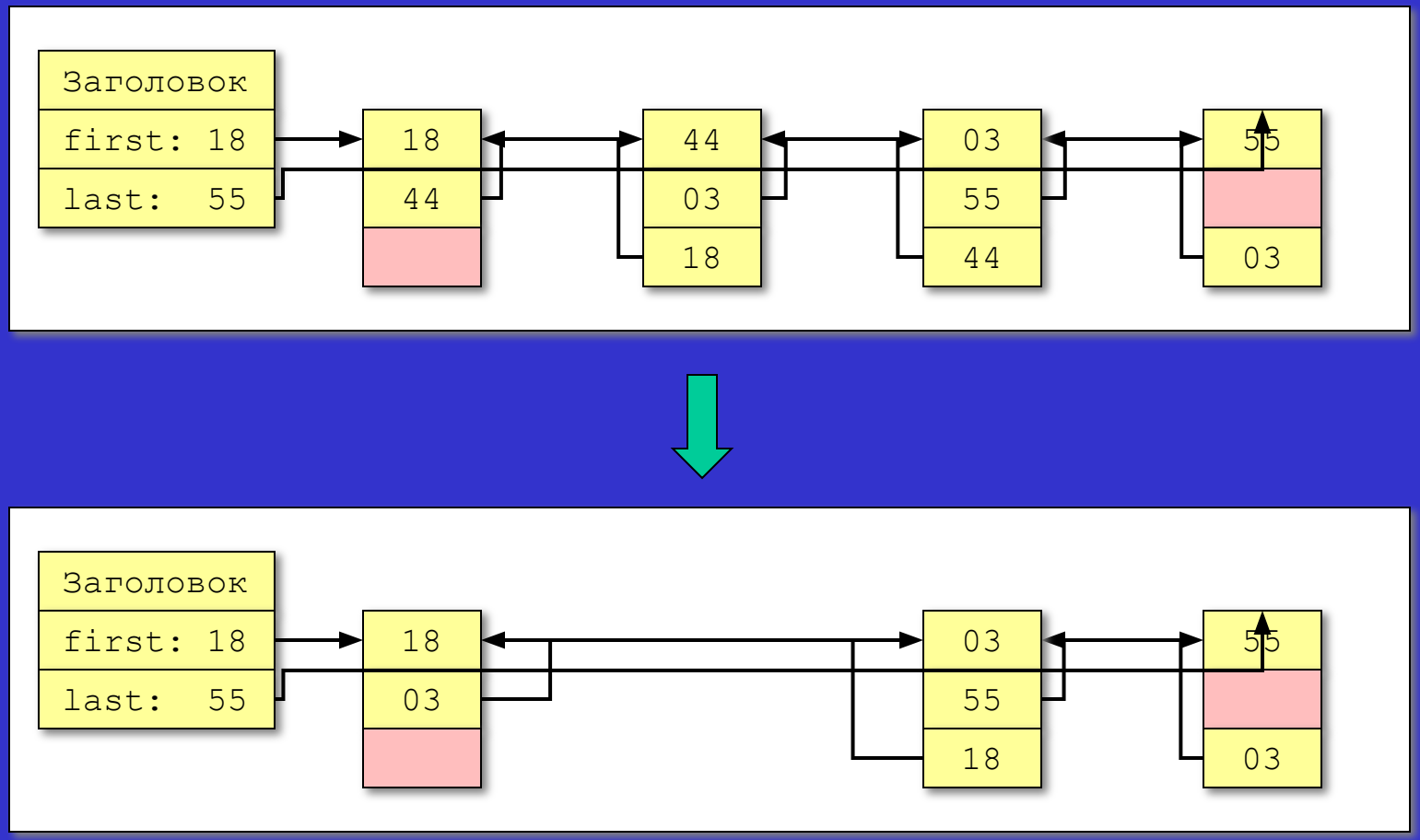
Пример включения в список

```
typedef struct {short code; char name[24]; short next;
                short prev;} PERS;

PERS pers[] = {{ 1, "Александров",  0,  0}, ...
               {03, "Антонов",      0,  0}, ...
               {18, "Иванов",      44,  0}, ...
               {44, "Петров",      55, 18}, ...
               {55, "Сидоров",      0, 44}, ... };
```



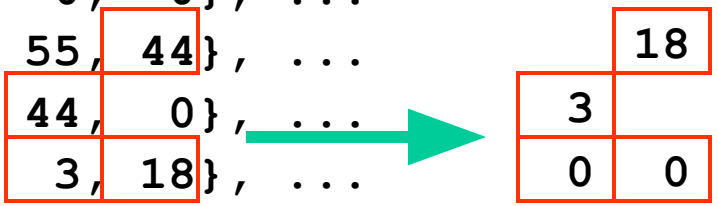

```
   int current = 44;
   int insert  = 3;
   int i = current-1;
   int j = insert-1;
   pers[pers[i].next-1].prev = insert;
   pers[j].next = pers[i].next;
   pers[i].next = insert;
   pers[j].prev = current;
```

Исключение из списка



Пример исключения из списка

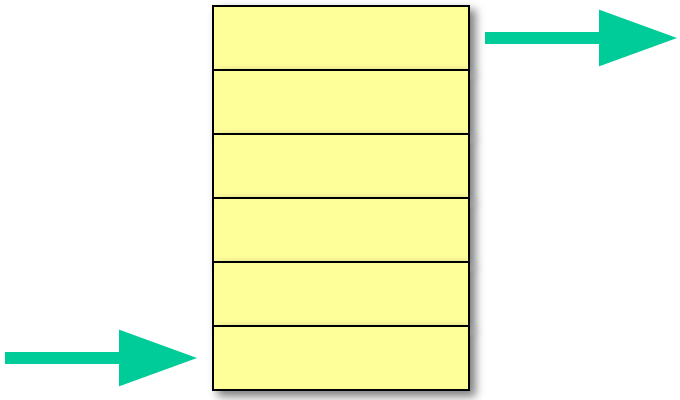
```
typedef struct {short code; char name[24]; short next;
                short prev;} PERS;

PERS pers[] = {{ 1, "Александров",  0,  0}, ...
               {03, "Антонов",     55, 44}, ...
               {18, "Иванов",      44,  0}, ...
               {44, "Петров",       3, 18}, ...
               {55, "Сидоров",      0,  3}, ... };
```




```
   int current = 44;
   int i = current-1;
   pers[pers[i].next-1].prev = pers[i].prev;
   pers[pers[i].prev-1].next = pers[i].next;
   pers[i].next = 0;
   pers[i].prev = 0;
```

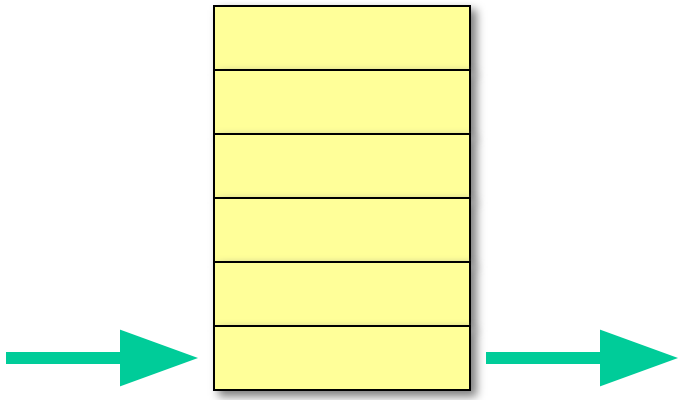
Очередь, стек

Очередь (FIFO)



Дисциплина очереди - первым вошел - первым вышел

Стек (LIFO)



Дисциплина стека - последним вошел - первым вышел

Пример реализации очереди

```
int queue[100];
int n = 0;

void QueueIn(int Value)
{
   queue[n++] = Value;
}

int QueueOut()
{
   int value = queue[0];
   n--;

   for(int i = 0; i < n; i++)
      queue[i] = queue[i+1];

   return(value);
}
```

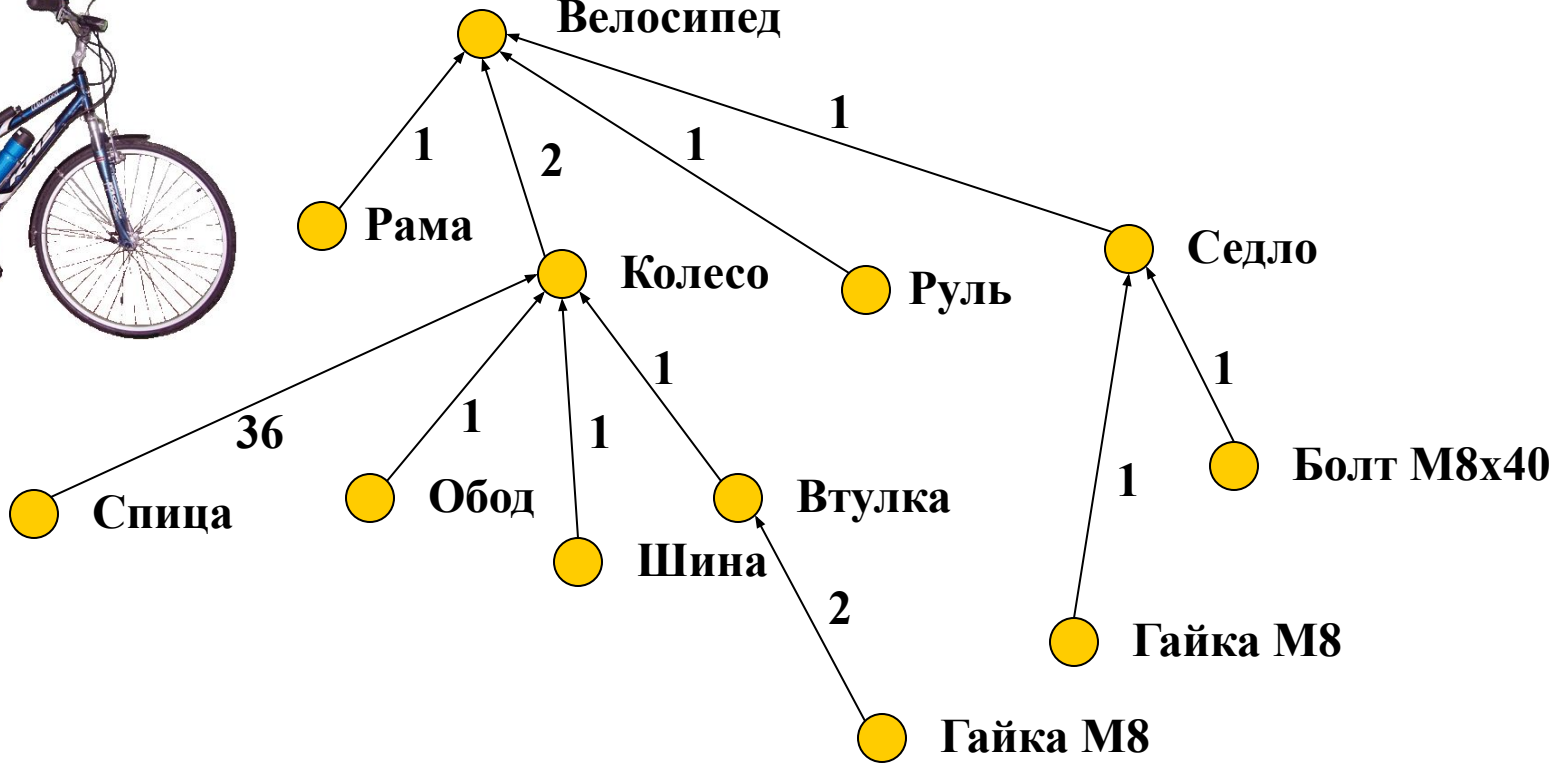
```
void main()
{
   QueueIn(18);
   QueueIn(44);
   QueueIn(3);
   QueueIn(55);

   while(n)
   {
      int next = QueueOut();
      ...
   }
}
```

Пример реализации стека

```
int stack[100];
int n = 0;

void StackIn(int Value)
{
   stack[n++] = Value;
}

int StackOut()
{
   return(stack[--n]);
}
```

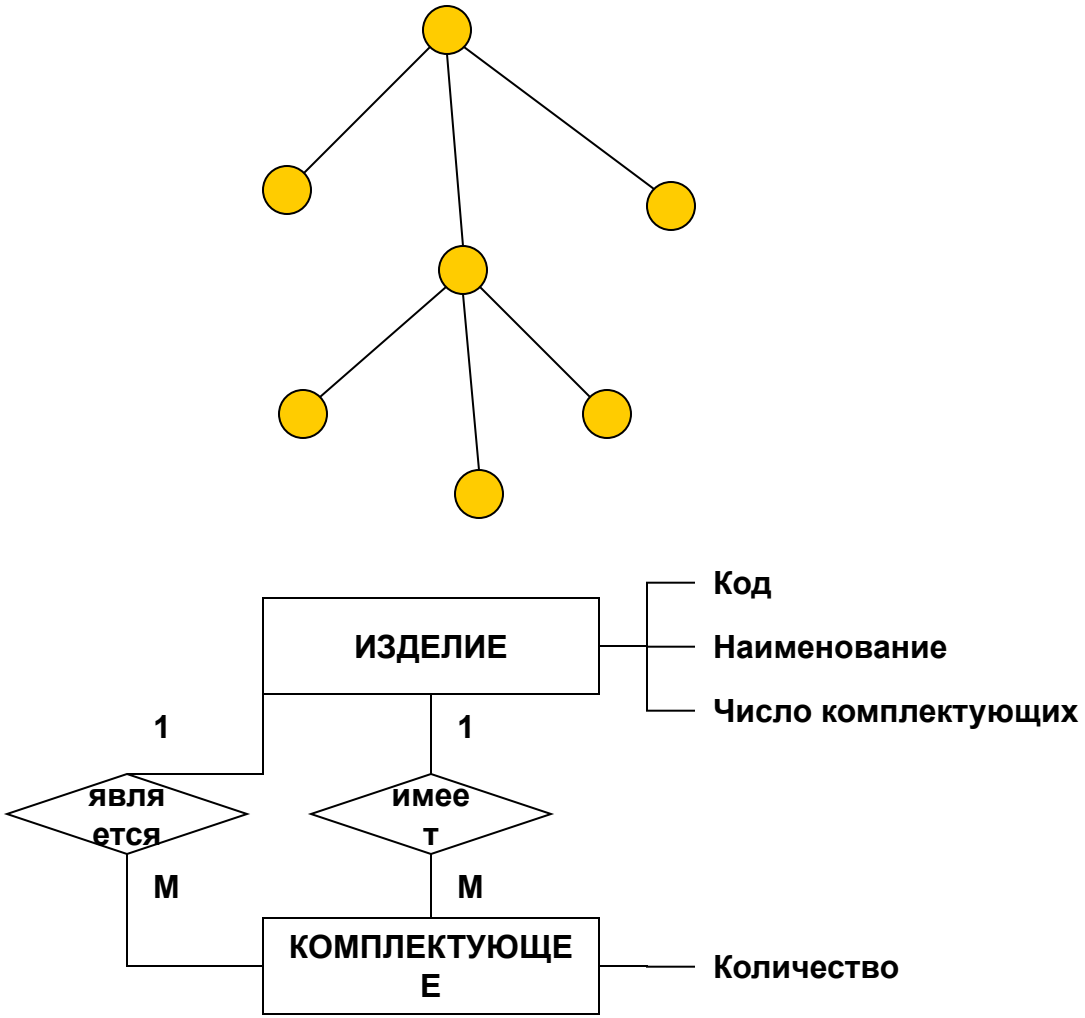
```
void main()
{
   StackIn(18);
   StackIn(44);
   StackIn(3);
   StackIn(55);

   while(n)
   {
      int next = StackOut();
      ...
   }
}
```

Деревья



Задача расчета потребности в ресурсах для партии в 100 шт.
Вариант 1: дерево

Алгоритм обхода дерева. Структура данных

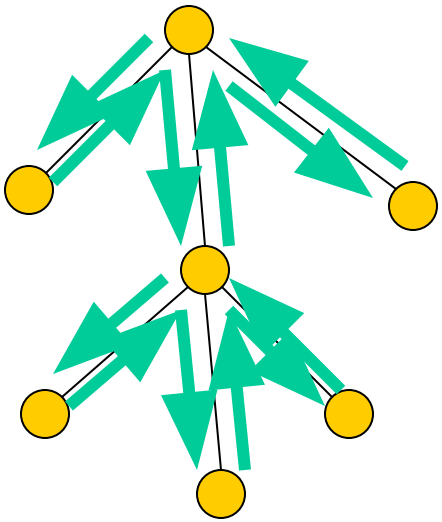



```
typedef struct
{
    short code;
    float quant;
} CHILD;

typedef struct
{
    short code;
    char  name[32];
    CHILD childs[8];
    int   nchild;
} PROD;
```

Алгоритм обхода дерева. Данные

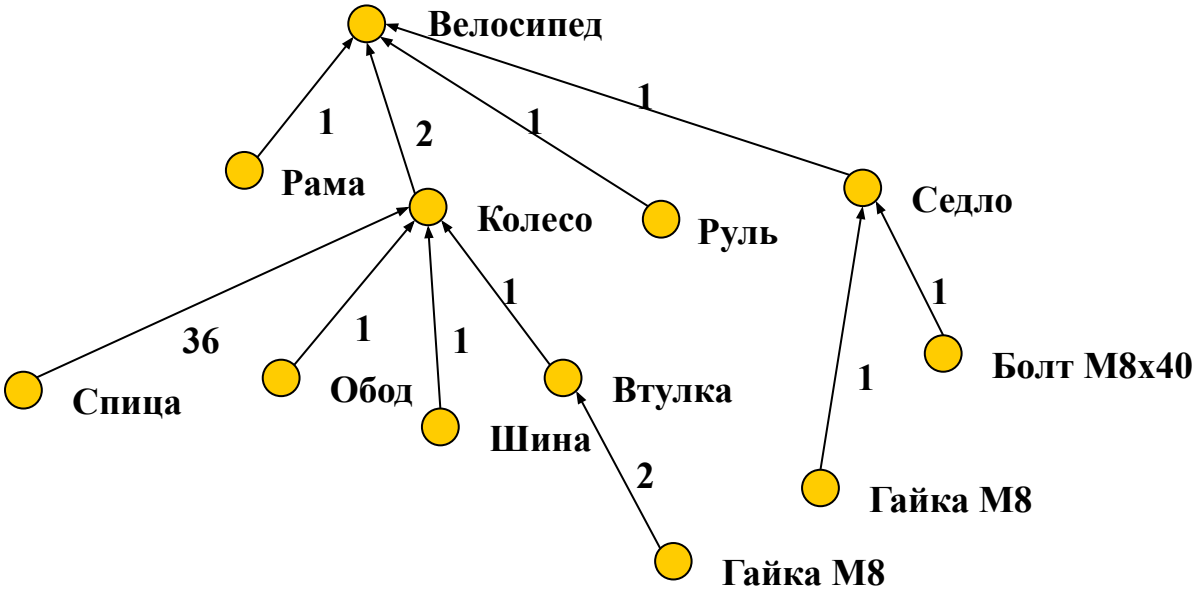
```
PROD prod[] =
{
   { 1, "Велосипед",  {{2, 1}, {3, 2}, {4, 1}, {5, 1}}, 4},
   { 2, "Рама",       {}, 0},
   { 3, "Колесо",     {{6, 36}, {7, 1}, {8, 1}, {9, 1}}, 4},
   { 4, "Руль",       {}, 0},
   { 5, "Седло",      {{10, 1}, {11, 1}}, 2},
   { 6, "Спица",      {}, 0},
   { 7, "Обод",       {}, 0},
   { 8, "Шина",       {}, 0},
   { 9, "Втулка",     {{10, 2}}, 1},
   {10, "Гайка М8",   {}, 0},
   {11, "Болт М8х40", {}, 0}
};
```

Алгоритм обхода дерева. Пример реализации



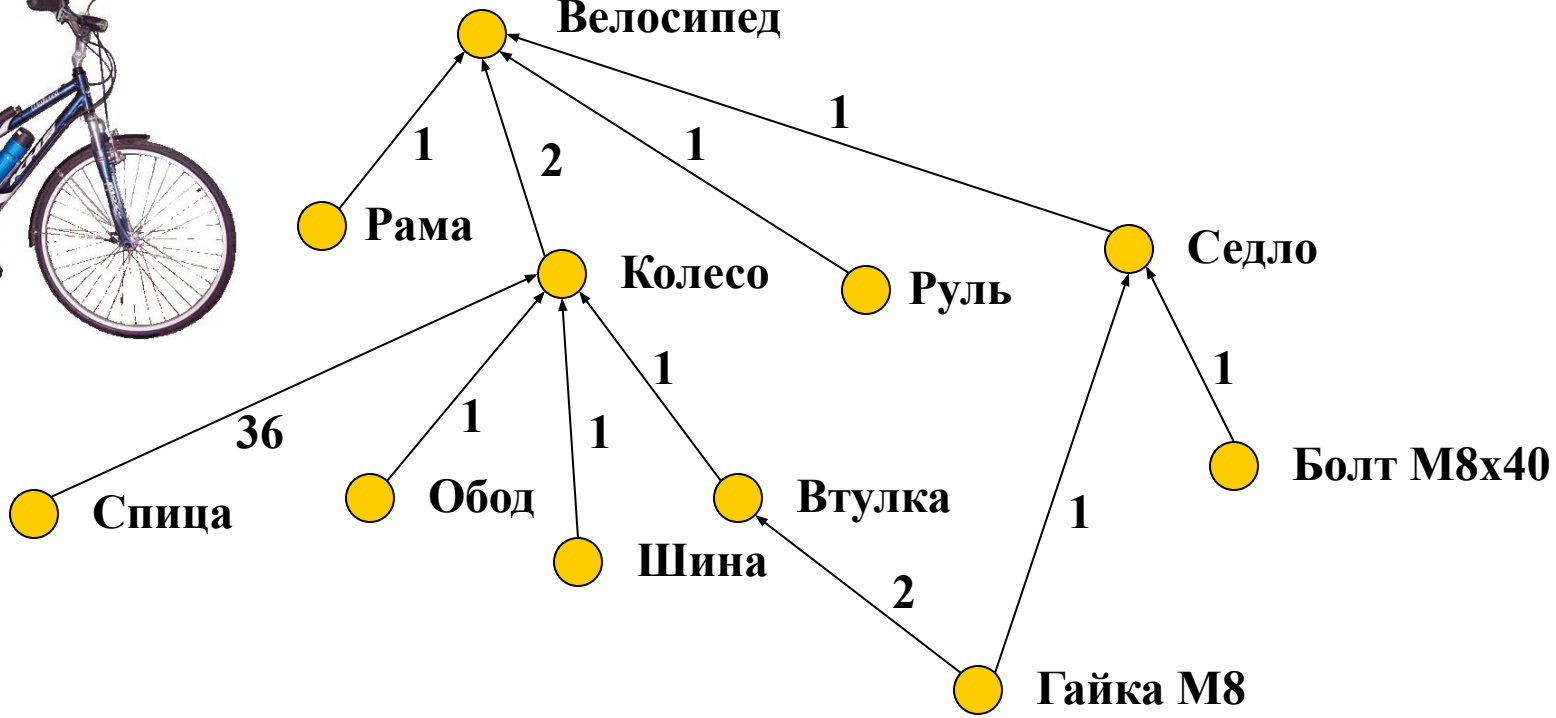
```
void treenode(short code, int Quant)
{
   PROD* p = &prod[code-1];
   printf("%s : %d\n", p->name, Quant);

   for(int i = 0; i < p->nchild; i++)
      treenode(p->childs[i].code,
               p->childs[i].quant*Quant);
}

void main()
{
   treenode(1, 100);
}
```

Алгоритм обхода дерева. Результат



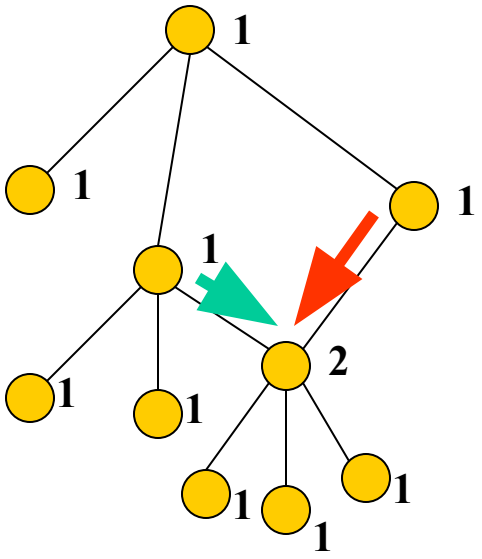

```
Велосипед  :  100
Рама       :  100
Колесо     :  200
Спица      : 7200
Обод       :  200
Шина       :  200
Втулка     :  200
Гайка М8   :  400
Руль       :  100
Седло      :  100
Гайка М8   :  100
Болт М8х40 :  100
```

Графы



Задача расчета потребности в ресурсах для партии в 100 шт.
Вариант 2: ациклический граф

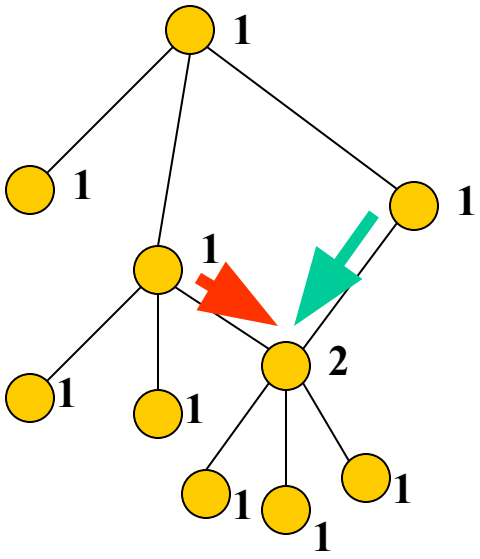
Алгоритм обхода графа. Разметка




```
typedef struct {
   short code;
   char  name[32];
   CHILD childs[8];
   int   nchild;
   int   count;
   int   quant;
} PROD;
```

```
void graphmark(short code)
{
   PROD* p = &prod[code-1];

   if(++p->count > 1) return;

   for(int i = 0; i < p->nchild; i++)
      graphmark(p->childs[i].code);
}

void main()
{
   for(int i = 0; i < NPROD; i++)
   {
      prod[i].count = 0;
      prod[i].quant = 0;
   }

   graphmark(1);
   ...
}
```

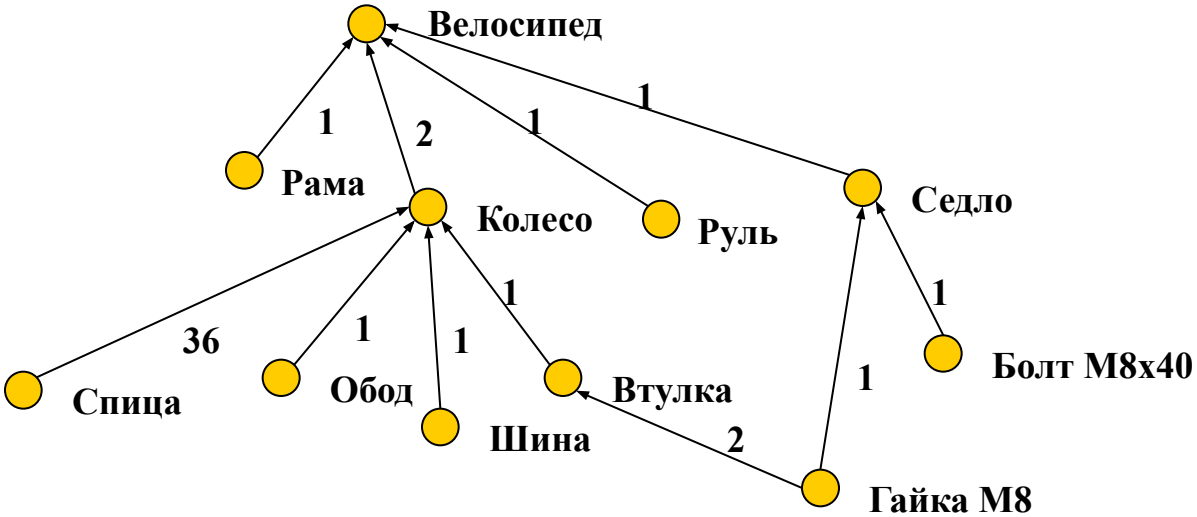
Алгоритм обхода графа. Расчет




```
typedef struct {
   short code;
   char  name[32];
   CHILD childs[8];
   int   nchild;
   int   count;
   int   quant;
} PROD;
```

```
void graphcalc(short code, int Quant)
{
   PROD* p = &prod[code-1];
   p->quant += Quant;

   if(--p->count > 0) return;

   printf("%s %d\n", p->name, Quant);

   for(int i = 0; i < p->nchild; i++)
      graphcalc(p->childs[i].code,
         p->childs[i].quant*p->quant);
}

void main()
{
   ...
   graphcalc(1, 100);
}
```

Алгоритм обхода графа. Результат




```
Велосипед   :   100
Рама        :   100
Колесо      :   200
Спица       :  7200
Обод        :   200
Шина        :   200
Втулка      :   200
Руль        :   100
Седло       :   100
Гайка М8    :   500
Болт М8х40  :   100
```

Соответствие между операторами и структурами данных

"Образ"	Операторы	C++	Данные	C++
Атомарный объект	Присваивание	`=`	Простой тип	`char, short`
Перечисление	Составной оператор	`{ ; ; }`	Запись	`struct`
Выбор	Условный оператор	`if`	Объединение	`union`
Предопределенное повторение	Цикл с параметром	`for`	Массив	`[]`
Повторение	Цикл с условием	`while`	Список	`*p`
Рекурсия	Процедура	`f()`	Дерево	`*p`
Граф	Оператор перехода	`goto`	Сеть	`*p`