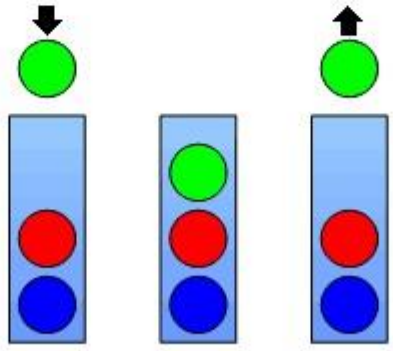
Чулюков В.А.

МЕТОДЫ РАЗРАБОТКИ ПРОГРАММ (АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ)

ЧАСТЬ 1

ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ



Воронеж - 2014

Лабораторная работа № 1

Задание 1.

Наберите и отладьте программу: Дано предложение, оканчивающееся точкой. Из символов предложения постройте простой линейный список и выведите его на экран.

```
program pereupspis;
uses crt;
type
   svjas=^objekt;
   objekt=record
              sled:svjas;
              dann:char
          end;
var
  nach,p:svjas;
  c:char;
begin
    clrscr;
    {заполнение списка}
    nach:=nil;
    read(c);
    while c<>'.' do
         begin
             new(p);
             p^.dann:=c;
             p^.sled:=nach;
             nach:=p;
             read(c)
         end;
   {чтение списка}
   p:=nach;
   while p<>nil do
        begin
            write(p^.dann);
            p:=p^.sled
        end;
end.
```

Содержание отчета:

1. Текст программы.
2. Протокол выполнения программы.

Задание 2.

Наберите и отладьте программу, работающую со следующими пунктами меню:

a. Добавить элемент в очередь
b. Удалить элемент из очереди
c. Просмотреть очередь
d. Выход

Структура программы с меню:

```
program queue;
uses crt;
type
----------
var
----------
procedure DelQueue(var first,oldfront,oldrear:svjas);
begin
 -----------
end;
procedure AddQueue(NewQ:svjas; var newfront,newrear:svjas);
begin
-------------
end;
begin {основная программа}
    clrscr;
    writeln('Демонстрационная программа.');
    writeln;
    writeln('ДИНАМИЧЕСКАЯ СТРУКТУРА "ОЧЕРЕДЬ"');
    front:=nil;
    repeat
         delay(2000);
         clrscr;
         writeln('1 - Добавить элемент в очередь');
         writeln('2 - Убрать элемент из очереди');
         writeln('3 - Показать очередь');
         writeln('4 - Выход');
         writeln('Выберите нужное');
         k:=readkey;
    case k of
    '1' :begin
            write('Кто хочет стать в очередь? ');
            new(p);
            readln(c);
            p^.sled:=nil;
            p^.dann:=c;
            AddQueue(p,front,rear)
        end;
    '2': begin
            write('Обслуживается очередной: ');
            DelQueue(p,front,rear);;
            if p=nil then writeln('Очередь пуста.')
            else writeln(p^.dann);
            if (front=nil) and (rear=nil) then
            writeln('Очередь опустела.')
        end;
    '3': begin
            writeln('ОЧЕРЕДЬ С ГОЛОВЫ:');
            p:=Front;
            if p=nil then writeln('Очередь пуста.')
```

```
          else
          while p<>nil do
            begin
              writeln(p^.dann);
              p:=p^.sled
            end
        end;
    '4': begin write('By!'); delay(1000) end
    end
    until k='4';
end.
```

Указания:

1. При попытке удалить элемент из пустой очереди должно выводиться сообщение «Очередь пуста»
2. При попытке прочитать пустую очередь должно выводиться то же сообщение.
3. При удалении последнего элемента очереди должно выводиться сообщение «Очередь опустела»

Содержание отчета:

1. Текст программы.
2. Протокол выполнения программы.

Задание 3.

Наберите и отладьте программу обработки линейного списка, использующую рекурсию.

Содержание отчета:

1. Текст программы.
2. Протокол выполнения программы.

Задание 4.

Наберите и отладьте программу общего алгоритма обработки списка, работающую со следующими пунктами меню:

1. Добавить элемент в начало списка');
2. Вставить элемент после');
3. Вставить элемент перед 1');
4. Вставить элемент перед 2');
5. Удалить элемент');
6. Показать список');
7. Распечатать ссылку');
8. Выход').

Примерная структура программы:

```
program obchii;{Общий алгоритм добавления и исключения}
uses crt;
type
-----------------
var
 -------------------------

{процедура печати ссылки}
procedure printss(slk:svjas);
  ----------------------
end;

{процедура первоначального создания списка}
procedure AddToBegin(var first:svjas);
var p:svjas;
begin
   ---------------
   writeln('Кто хочет стать в список ?');
   ----------------
   writeln(p^.dann,' помещен в начало списка')
end;

{процедура просмотра списка}
Procedure Look;
var p:svjas;
begin
    p:=nach;
    writeln('НАЧАЛО СПИСКА');
    while p<>nil do
        begin
            writeln(p^.dann);
            p:=p^.sled
        end;
    writeln('КОНЕЦ СПИСКА');
end;

{процедура поиска заданного элемента}
procedure find(what:string; var p:svjas);
begin
    ----------------------
end;

{процедура «добавить после»}
procedure addafter(vanja,vasja:svjas);
begin
    vasja^.sled:=vanja^.sled;
    vanja^.sled:=vasja
end;

{процедура «добавить перед» 1 способ}
procedure addbefore1(vanja,vasja:svjas; var nach:svjas);
```

```
var
  ----------------------
begin
-------------------------
end;

{процедура «добавить перед» 2 способ}
procedure addbefore2(vanja,vasja:svjas);
var
----------------------
begin
----------------------
end;

{процедура «удалить»}
procedure del(vanja:svjas; var nach:svjas);
var
-------------
begin
   --------------------------
end;

begin
   clrscr;
   writeln('Демонстрационная программа.');
   writeln;
   writeln('ОБЩИЙ АЛГОРИТМ ДОБАВЛЕНИЯ И ИСКЛЮЧЕНИЯ');
   nach:=nil;
   repeat
      delay(2000);
      clrscr;
      writeln('1 - Добавить элемент в начало списка');
      writeln('2 - Вставить элемент после');
      writeln('3 - Вставить элемент перед 1');
      writeln('4 - Вставить элемент перед 2');
      writeln('5 - Удалить элемент');
      writeln('6 - Показать список');
      writeln('7 - Распечатать ссылку');
      writeln('8 - Выход');
      writeln('Выберите нужное');
      k:=readkey;
   case k of
   '1' : AddToBegin(nach);
   '2': begin
      writeln('Кого найти ?');
      readln(c);
      find(c,sdes);
      if sdes<>nil then
        begin
          writeln('Ссылка показывает на ',sdes^.dann);
          new(vasja);
          writeln('Кого вставить после ',sdes^.dann,'?');
```

```
          readln(vasja^.dann);
          addafter(sdes,vasja);
          writeln(vasja^.dann,' вставлен после ',sdes^.dann)
        end
      else writeln(c, ' не найден');
     end;
'3': begin
      writeln('Кого найти ?');
           ------------------------
      if sdes<>nil then
        begin
          writeln('Ссылка показывает на ',sdes^.dann);
           ----------------------
           --------------------------
          ----------------------readln(vasja^.dann);
          addbefore1(sdes,vasja,nach);
          writeln(vasja^.dann,' вставлен перед ',sdes^.dann)
        end
      else writeln(c, ' не найден');
     end;
'4': begin
      writeln('Кого найти ?');
      -------------------------
      if sdes<>nil then
        begin
          writeln('Ссылка показывает на ',sdes^.dann);
          ---------------------------------
          addbefore2(sdes,vasja);
          writeln(sdes^.dann,' вставлен перед ',vasja^.dann)
        end
      else writeln(c, ' не найден');
     end;
'5': begin
      writeln('Кого удалить ?');
      ---------------------------------
      if sdes<>nil then
        begin
          writeln('Ссылка показывает на ',sdes^.dann);
          del(sdes,nach);
          writeln(sdes^.dann,' удален')
        end
      else writeln(c, ' не найден');
     end;
'6': Look;
'7': begin
      writeln('Кого найти ?');
      --------------------------
      if sdes<>nil then
        begin
          writeln('Ссылка показывает на ',sdes^.dann);
          write('Значение ссылки ');
          printss(sdes)
```

```
            end
         else writeln(c, ' не найден');
        end;
    '8': begin write('By!'); delay(1000) end;
    end
    until k='8';
end.
```

Задание 5.

Наберите и отладьте программу обработки двусвязного кольца, работающую со следующими пунктами меню:

1. Вставить элемент после заданного
2. Вставить элемент перед заданным
3. Удалить элемент
4. Просмотреть кольцо по часовой стрелке
5. Просмотреть кольцо против часовой стрелки
6. Выход

Для поиска ссылки p:svjas на заданный элемент what используйте процедуру

```
procedure find(what:string; var p:svjas);
begin
    p:=nach^.vss;
    while (p^.dann<>what) and (p<>nach) do
        p:=p^.vss;
end;
```

примерная структура программы:

```
program circle;{Двусвязное кольцо}
uses crt;
type
 --------------------
var
----------------------
Procedure Look(forv:boolean);
var kolco:svjas;
begin
    if forv then
      begin
        writeln('Просмотр вперед');
        ---------------------------------
      end
    else
       begin
        writeln('Просмотр назад');
        ---------------------------------
      end
end;
```

```
procedure find(what:string; var p:svjas);
begin
    --------------------
end;

procedure addafter(vanja,vasja:svjas);
begin
    ----------------------
end;

procedure addbefore(vanja,vasja:svjas);
begin
    -------------------------
end;

procedure del(vanja:svjas);
begin
    ------------------------
end;

begin
    clrscr;
    writeln('Демонстрационная программа.');
    writeln;
    writeln('ДВУСВЯЗНОЕ КОЛЬЦО');
    {Инициирование пустого кольца}
    ----------------------------------------
    repeat
        delay(2000);
        clrscr;
        writeln('1 - Вставить элемент после');
        writeln('2 - Вставить элемент перед');
        writeln('3 - Удалить элемент');
        writeln('4 - Показать кольцо вперед');
        writeln('5 - Показать кольцо назад');
        writeln('7 - Выход');
        writeln('Выберите нужное');
        k:=readkey;

    case k of
    '1': begin
if nach^.vss=nach then
            begin
            writeln('Кольцо пустое');
            new(vasja);
            writeln('Кого вставить?');
            readln(vasja^.dann);
            addafter(nach,vasja)
            end
        else
        begin
            writeln('Кого найти ?');
```

```
        readln(c);
        find(c,sdes);
        if sdes<>nach then
          begin
            writeln('Ссылка показывает на ',sdes^.dann);
            new(vasja);
            writeln('Кого вставить после ',sdes^.dann,'?');
            readln(vasja^.dann);
            addafter(sdes,vasja);
            writeln(vasja^.dann,' вставлен после ',sdes^.dann)
          end
        else writeln(c, ' не найден');
      end
     end;
'2': begin
      if nach^.nss=nach then
        begin
        writeln('Кольцо пустое');
        new(vasja);
        writeln('Кого вставить?');
        readln(vasja^.dann);
        addbefore(nach,vasja)
        end
      else
      begin
      writeln('Кого найти ?');
      readln(c);
      find(c,sdes);
      if sdes<>nach then
        begin
          writeln('Ссылка показывает на ',sdes^.dann);
          new(vasja);
          writeln('Кого вставить перед ',sdes^.dann,'?');
          readln(vasja^.dann);
          addbefore(sdes,vasja);
          writeln(vasja^.dann,' вставлен перед ',sdes^.dann)
        end
      else writeln(c, ' не найден');
     end
     end;
'3': begin
      writeln('Кого удалить ?');
      readln(c);
      find(c,sdes);
      if sdes<>nach then
        begin
          writeln('Ссылка показывает на ',sdes^.dann);
          del(sdes);
          writeln(sdes^.dann,' удален')
        end
      else writeln(c, ' не найден');
     end;
```

```
    '4': Look(true);
    '5': Look(False);
    '7': begin write('By!'); delay(1000) end;
    end
    until k='7';
end.
```

Содержание отчета:

1. Текст программы.
2. Протокол выполнения программы.

Задание 6.

Наберите и отладьте программу обработки двоичного дерева поиска, работающую со следующими пунктами меню:

1. Вставить в дерево узел
2. Прямой просмотр
3. Обратный просмотр
4. Концевой просмотр
5. Найти заданный узел
6. Выход

```
program mytree;
uses crt;
type
   svjas=^usel;
   usel=record
            left,right:svjas;
            dann:char;
        end;
var
  treechar,treefind:svjas;
  k,mydann:char;
procedure look(tree:svjas;napr:byte);{1-прямой,2-обратный,3-концевой}
  begin
      if tree<>nil then
        begin
          case napr of
              1: begin
                    write(tree^.dann);
                    look(tree^.left,1);
                    look(tree^.right,1)
                 end;
              2: begin
                    look(tree^.left,2);
                    write(tree^.dann);
                    look(tree^.right,2)
                 end;
              3: begin
                    look(tree^.left,3);
```

```
                    look(tree^.right,3);
                    write(tree^.dann);
                end
            end
        end
end;

procedure insert(var tree:svjas; newdann:char);
  begin
      if tree=nil then
        begin
            new(tree);
            with tree^ do
                begin
                    left:=nil;
                    right:=nil;
                    dann:=newdann
                end
         end
      else
          with tree^ do
              if newdann<dann
                  then insert(left,newdann)
              else if newdann>dann
                  then insert(right,newdann)
              else writeln('Дублирование информации !')
  end;

function find1(tree:svjas; key:char):svjas;
    begin
        if tree=nil then
          find1:=nil
        else
            with tree^ do
                if key<dann then
                  find1:=find1(left,key)
                else if key>dann then
                  find1:=find1(right,key)
                else find1:=tree
    end;

function find2(tree:svjas; key:char):svjas;
  var finish:boolean;
  begin
      finish:=false;
      repeat
          if tree=nil then
            finish:=true
          else
              with tree^ do
                  if key<dann then
                    tree:=left
```

```
                else if key>dann
                    then tree:=right
                else finish:=true
      until finish;
      find2:=tree
  end;

begin
    clrscr;
    writeln('Демонстрационная программа.');
    writeln;
    writeln('РАБОТА С ДВОИЧНЫМ ДЕРЕВОМ');
    { nach:=nil;}
    repeat
        delay(2000);
        clrscr;
        writeln('1 - Вставить символ в дерево');
        writeln('2 - Прямой просмотр дерева');
        writeln('3 - Обратный просмотр дерева');
        writeln('4 - Концевой просмотр дерева');
        writeln('5 - Найти символ 1');
        writeln('6 - Найти символ 2');
        writeln('7 - Выход');
        writeln('Выберите нужное');
        k:=readkey;
    case k of
    '1' : Begin
            write('Какой символ вставить ? ');
            readln(mydann);
            insert(treechar,mydann)
        end;
    '2': look(treechar,1);
    '3': look(treechar,2);
    '4': look(treechar,3);
    '5': begin
          write('Какой символ найти ? ');
          readln(mydann);
          treefind:=find1(treechar,mydann);
          if treefind<>nil then
            write('Найден символ ',treefind^.dann)
          else write('Такого символа нет !')
        end;
    '6': begin
          write('Какой символ найти ? ');
          readln(mydann);
          treefind:=find2(treechar,mydann);
          if treefind<>nil then
            write('Найден символ ',treefind^.dann)
          else write('Такого символа нет !')
        end;
    '7': begin write('Ву!'); delay(1000) end;
    end
```

```
    until k='7';
end.
```

Содержание отчета:

3. Текст программы.
4. Протокол выполнения программы.