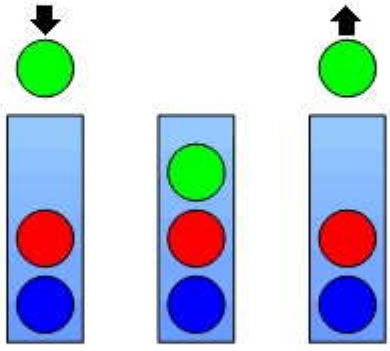
Чулюков В.А.

МЕТОДЫ РАЗРАБОТКИ ПРОГРАММ (АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ)

ЧАСТЬ 1

ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ



Воронеж - 2014

Лабораторная работа № 1

ТЕМА: **Динамические структуры данных. Списковые структуры**

Продолжительность 4 часа

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

```
program queue;
uses crt;
type
   svjas=^objekt;
   objekt=record
              sled:svjas;
              dann:string
          end;
var
  front,rear,p:svjas;
  k:char;
  c:string;
procedure DelQueue(var first,oldfront,oldrear:svjas);
begin
    first:=oldfront;
    if oldfront<>nil then
      begin
          oldfront:=oldfront^.sled;
          if oldfront=nil then
            oldrear:=nil
      end
end;
procedure AddQueue(NewQ:svjas; var newfront,newrear:svjas);
begin
    if newfront=nil then
      newfront:=NewQ
    else
       newrear^.sled:=NewQ;
    newrear:=NewQ
end;
begin
    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;
            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 стала пустой»

Содержание отчета:

1. Текст программы.
2. Протокол выполнения программы.

Задание 3.

Наберите и отладьте программу обработки линейного списка, использующую рекурсию.

```
program spisrecur;
uses crt;
type
   svjas=^objekt;
   objekt=record
              sled:svjas;
              dann:char
         end;
var
  nach:svjas;
  c:char;
procedure addsp(var ss:svjas);
   begin
       if ss=nil then
          begin
              new(ss);
              ss^.sled:=nil;
              read(c);
              ss^.dann:=c
          end
       else
          addsp(ss^.sled)
end;

procedure printsp(ss:svjas);
begin
     if ss<>nil then
        begin
            write(ss^.dann);
            printsp(ss^.sled)
        end
end;

begin
    clrscr;
    writeln('Введите строку, оканчивающуюся точкой:');
    nach:=nil;
    while c<>'.' do
        addsp(nach);
    writeln('Ввывод списка:');
        printsp(nach);
    readln
end.
```

Содержание отчета:

1. Текст программы.
2. Протокол выполнения программы.

Задание 4.

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

a. Вставить элемент после заданного
b. Вставить элемент перед заданным
c. Удалить элемент
d. Просмотреть кольцо по часовой стрелке
e. Просмотреть кольцо против часовой стрелки
f. Выход

```
program circle;{Двусвязное кольцо}
uses crt;
type
   svjas=^objekt;
   objekt=record
             vss,nss:svjas;
             dann:string
         end;
var
  nach,vanja,vasja,start,kolco,sdes:svjas;
  k:char;
  c:string;

Procedure Look(forv:boolean);
var kolco:svjas;
begin
    if forv then
      begin
         writeln('Просмотр вперед');
         kolco:=nach^.vss;
         while kolco<>nach do
              begin
                  writeln(kolco^.dann);
                  kolco:=kolco^.vss
              end
      end
    else
       begin
         writeln('Просмотр назад');
         kolco:=nach^.nss;
         while kolco<>nach do
              begin
                  writeln(kolco^.dann);
                  kolco:=kolco^.nss
              end
      end
end;

procedure find(what:string; var p:svjas);
begin
    p:=nach^.vss;
```

```
    while (p^.dann<>what) and (p<>nach) do
        p:=p^.vss;
end;

procedure addafter(vanja,vasja:svjas);
begin
    vasja^.vss:=vanja^.vss;
    vasja^.nss:=vanja;
    vanja^.vss^.nss:=vasja;
    vanja^.vss:=vasja
end;

procedure addbefore(vanja,vasja:svjas);
begin
    vasja^.vss:=vanja;
    vasja^.nss:=vanja^.nss;
    vanja^.nss^.vss:=vasja;
    vanja^.nss:=vasja
end;

procedure del(vanja:svjas);
begin
    vanja^.vss^.nss:=vanja^.nss;
    vanja^.nss^.vss:=vanja^.vss
end;

begin
    clrscr;
    writeln('Демонстрационная программа.');
    writeln;
    writeln('ДВУСВЯЗНОЕ КОЛЬЦО');
    {Инициирование пустого кольца}
    new(nach);
    with nach^ do
        begin
            vss:=nach;
            nss:=nach;
            dann:='начало'
        end;
    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 5.

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

a. Вставить в дерево узел
b. Прямой просмотр
c. Обратный просмотр
d. Концевой просмотр
e. Найти заданный узел
f. 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('By!'); delay(1000) end;
    end
    until k='7';
end.
```

Содержание отчета:

3. Текст программы.
4. Протокол выполнения программы.