Структуры данных

Списки (list)

Связанные списки (linked list)

Линейный – упорядоченное множество, состоящее из переменного числа элементов, отражающий отношения соседства между элементами. Односвязный список имеет в поле связок каждого элемента один указатель: следующий элемент. Кроме того, есть особые указатели: начало списка и конец списка. Двухсвязный список – имеет два указателя: следующий элемент и предыдущий элемент. Можно задать кольцевой список – из последнего элемента указатель ссылается на первый элемент. Многосвязный список (мультисписок) – каждый элемент включает несколько указателей на связи между собой подмножеств данного списка.

Разветвленный – это списки, элементами которых могут быть тоже списки. Выше рассмотрены двухсвязные линейные списки. Если один из указателей каждого элемента списка задает порядок обратный к порядку, устанавливаемому другим указателем, то такой двусвязный список будет линейным. Если же один из указателей задает порядок произвольного вида, не являющийся обратным по отношению к порядку, устанавливаемому другим указателем, то такой список будет нелинейным.

Каждый элемент списка может быть идентифицирован по ключу. Обычно ключ – это число или строка символов, расположенные в поле данных элемента как часть поля записи (структуры) или как отдельное поле. Ключи разных элементов списка могут совпадать.

Основные операции над списками:

- начальное форматирование списка (создание первого элемента);
- добавление элемента в конец списка;
- чтение (выборка) элемента с заданным ключом ;
- вставка элемента в заданное место списка (до или после элемента с заданным ключом);
- удаление элемента с заданным ключом;
- упорядочивание списка по ключу.

Структуры данных

Pascal

Списки (list)

Pascal

Практическое занятие

***Односвязный список* (процедуры)**

```
type Pt=^k;  k=record  d: char; p: Pt   end;
var ph, pk, px, pc: Pt;  c, kl: char; pr: boolean;

Procedure  FormLi (var ph,pk: Pt; var c: char);
(*Формирование первого элемента списка *)
begin
  New (ph);
  ph^.p := nil;
  ph^.d := c;
  pk := ph;
end;

Procedure  AddLi (var pk: Pt; var c: char);
(* Вставка элемента в список*)
    var px: Pt;
Begin
  New (px);
  px^.p:=nil;
  pk^.p:=px;
  pk:=px;
  pk^.d:=c;
end;

Procedure  DelLi (var ph: Pt; var kl: char);
(*Удаление элемента из списка по заданному
ключу*)
  var px, pc: Pt;
begin
  SearchLi (kl, ph, pc, px, pr);
   px^.p := pc^.p;
Dispose (pc);
end;
```

```
Procedure  SearchLi (var kl: char; var ph,pc,px: Pt;
var pr: boolean);
(*Поиск элемента в списке по заданному ключу*)
begin
   pc := ph;
   while (pc<>nil) and (kl<>pc^.d) do
      begin
        px := pc;
        pc := pc^.p;
      end;
   if (pc = nil) and (kl<>pc^.d) then
     pr := False
    else
     pr := True
end;

Procedure  InsLi (var kl, c: char);
(*Вставка элемента в список по заданному ключу*)
   Var   pm := Pt;
begin
   SearchLi (kl, ph, pc, px, pr);
   New (pm);
   pm^.d := c;
   pm^.p := pc^.p ;
   pc^.p := pm;
end;
```

Структуры данных

Pascal

Списки (list)

Pascal

Практическое занятие

Сформировать используя процедуры односвязный список, добавить в него N элементов, выполнить вставку и удаление элемента по ключу, а затем считать и вывести список на экран. Элементы должны быть символьного типа. Ввод элементов с клавиатуры, признак конца ввода – точка.

```
Program ListP;
 type  Pt=^k;    k=record  d:  char;  p:  Pt
end;
 var        ph, pk, px, pc: Pt;  c, kl: char;
            n: integer; pr: boolean;
begin
  n:=1;
  writeln (' Введите первый элемент ');
  readln (c);
  FormLi (ph, pk, c);
 repeat
    n:=n+1;
    writeln (' Введите ', n:3,'-й элемент');
    readln (c);
    AddLi (pk, c);
  until c='.';
 n:=1;
 px := ph;
  repeat
    writeln (n:3, '  ':3, px^.d);
    px := px^.p;
    n:=n+1;
  until  px=nil;
```

См. продолжение

продолжение:

```
  writeln (' Введите kl+ ');
  readln (kl);
  writeln (' Введите c+ ');
  readln (c);
InsLi (kl, c);
   writeln (' Введите kl-');
   readln (kl);
DelLi ( ph, kl);
  n:=1;
  repeat
    writeln (n:3, '  ':5, px^.d);
    px := px^.p;
    n:=n+1;
  until  px=nil;
end.
```

Структуры данных

Списки (list)

C / C++

C / C++

Практическое занятие

Двусвязный список (функции)

```
#include <iostream.h>
#define  bool    int
#define true 1
#define false 0

struct Node {   int d;   Node *next; Node *prev; };


Node * firstLi (int d)
// Начальное формирование списка
{    Node *pv = new Node;
     pv -> d  = d;
     pv -> next  = 0;     pv -> prev = 0;
 return pv;   }

void addLi (Node **pend, int d)
// Добавление элемента  в конец списка
{  Node *pv = new Node;
   pv -> d  = d;
   pv -> next  =0;  pv -> prev = *pend;
   (*pend) -> next  = pv;
   *pend = pv;   }

Node * findLi (Node * const pbeg, int d)
/ Поиск  элемента в списке по ключу
{    Node *pv = pbeg;
  while (pv)
     {   if (pv -> d == d)  break;
         pv = pv -> next;     }
return pv;  }
```

```
bool removeLi (Node **pbeg, Node **pend, int key)
// Удаление  элемента из списка по ключу
{   Node *pkey = findLi (pbeg, key);
    if (pkey)
     {   if (pkey == *pbeg)
           { *pbeg = (*pbeg) -> next;          (*pbeg)-> prev = 0;   }
         else  if (pkey == *pend)
                 { *pend = (*pend) -> prev;    (*pend) -> next = 0;  }
               else   { (pkey -> prev) -> next = pkey –> next;
                        (pkey -> next) -> prev = pkey –> prev;   }
   delete pkey;
    return true;  }
return false;          }

Node * insertLi (Node **pbeg, Node **pend, int key, int d)
// Вставка  элемента в список по ключу
{   Node *pkey = findLi (pbeg, key);
    if (pkey)
     { Node *pv = new Node;
       pv -> d = d;
       pv->next=pkey->next;
       pv->prev= pkey;
       pkey->next=pv;
 if (pkey!=*pend) (pv->next)->prev=pv; else *pend = pv;
return pv;}
return 0;   }
```

Вопрос:

Объяснить работу указателей в этих функциях. В том числе объяснить конструкции: Node **pend и в bool removeLi Node **pbeg и Node **pend

Структуры данных

C / C++

Списки (list)

Практическое занятие

C / C++

Сформировать двусвязный список из N целых чисел, добавить число в список, удалить число из списка и вывести список на экран. Функции разместить в конце программы.

```
#include <iostream.h>
#define bool int
#define true 1
#define false 0

  struct Node { int d; Node *next; Node *prev; };
  int N, ad, kl;

Node * firstLi (int d);
void addLi (Node **pend, int d);
Node * findLi (Node * const pbeg, int i);
bool removeLi  (Node **pbeg, Node **pend,  int key);
Node * insertLi (Node * const pbeg, Node **pend, int key, int d);

int main()
{    // Формирование первого элемента списка
    Node *pbeg = firstLi(1);
    Node *pend = pbeg;
cout << "\nВведите размер списка - "; cin >> N;
// Добавление в конец списка  N элементов
 for (int i=2; i<N+1; i++)
    addLi (&pend, i);
// Вставка элемента d+ после элемента  kl+
cout << "Введите вставляемый элемент ad+  = "; cin >> ad;
cout << "Введите ключевой элемент  kl+  = "; cin >> kl;
   insertLi (pbeg, &pend, kl, ad);
```

См. продолжение

продолжение:

```
// Удаление элемента с значением  kl-
cout << "Введите удаляемый элемент
kl-  = ";      cin >> kl;
  if (!removeLi (&pbeg, &pend, kl))
       cout << " не найден";
  Node *pv = pbeg;
// Вывод списка на экран
 while (pv)
   {
   cout << pv -> d << "   ";
   pv = pv -> next;
   }
 return 0;
}


//Функции работы со списком
……….
```

Структуры данных

Списки (list)

C / C++

C / C++

Практическое занятие

Двусвязный список (функции)

```
#include <iostream.h>
#define  bool    int
#define true 1
#define false 0
     struct Node {   int d;   Node *next; Node *prev; };
Node * firstLi (int d)
                    // Начальное формирование списка
{    Node *pv = new Node;
    pv -> d  = d;    pv -> next  = 0;     pv -> prev = 0;    return pv;   }
void addLi (Node **pend, int d)
                  // Добавление элемента  в конец списка
{  Node *pv = new Node;
   pv -> d  = d;  pv -> next=0;  pv->prev = *pend;  (*pend) -> next  = pv;
   *pend = pv;   }
Node * findLi (Node * const pbeg, int d)
                    // Поиск  элемента в списке по ключу
{    Node *pv = pbeg;
  while (pv)          {   if (pv -> d == d)  break;  pv = pv -> next;     }
       return pv;  }
bool removeLi (Node **pbeg, Node **pend, int key)
                  // Удаление  элемента из списка по ключу
{   Node *pkey = findLi (pbeg, key);
    if (pkey)
     {   if (pkey == *pbeg)
           { *pbeg = (*pbeg) -> next;  (*pbeg)-> prev = 0;   }
         else  if (pkey == *pend)
                 { *pend = (*pend) -> prev;    (*pend) -> next = 0;  }
               else   { (pkey -> prev) -> next = pkey –> next;
                        (pkey -> next) -> prev = pkey –> prev;   }
   delete pkey;  return true;  }
  return false;           }
Node * insertLi (Node **pbeg, Node **pend, int key, int d)
                  // Вставка  элемента в список по ключу
{   Node *pkey = findLi (pbeg, key);
    if (pkey)
     { Node *pv = new Node;
       pv -> d = d;  pv->next=pkey->next;  pv->prev= pkey;
       pkey->next=pv;
 if (pkey!=*pend) (pv->next)->prev=pv; else *pend = pv;
return pv;}
return 0;    }
```

Сформировать двусвязный список из N целых чисел, добавить число в список, удалить число из списка и вывести список на экран. Функции разместить в конце программы.

```
#include <iostream.h>
#define bool int
#define true 1
#define false 0
  struct Node { int d; Node *next; Node *prev; };        int N, ad, kl;
Node * firstLi (int d);
void addLi (Node **pend, int d);
Node * findLi (Node * const pbeg, int i);
bool removeLi  (Node **pbeg, Node **pend,  int key);
Node * insertLi (Node * const pbeg, Node **pend, int key, int d);
int main()
{    // Формирование первого элемента списка
    Node *pbeg = firstLi(1);
    Node *pend = pbeg;
cout << "\nВведите размер списка - "; cin >> N;
// Добавление в конец списка  N элементов
 for (int i=2; i<N+1; i++)
    addLi (&pend, i);
// Вставка элемента d+ после элемента  kl+
cout << "Введите вставляемый элемент ad+  = "; cin >> ad;
cout << "Введите ключевой элемент  kl+  = "; cin >> kl;
   insertLi (pbeg, &pend, kl, ad);
// Удаление элемента с значением  kl-
cout << "Введите удаляемый элемент kl-  = ";       cin >> kl;
  if (!removeLi (&pbeg, &pend, kl))
        cout << " не найден";
  Node *pv = pbeg;
// Вывод списка на экран
 while (pv)   { cout << pv -> d << "  ";    pv = pv -> next;    }
 return 0;
}

//Функции работы со списком
……..……
```

Вопрос:

Объяснить работу указателей в этих функциях. В том числе объяснить конструкции: Node **pend и в bool removeLi Node **pbeg и Node **pend