**Линейные двухсвязные списки**

Реализация одного элемента списка

typedef struct node

{

int data;

struct node \*next;

struct node \*previos;

}ITEM;

Также имеет смысл хранить не только начало списка, но и конец. Объединим эти два указателя в одну структуру головного элемента

typedef struct head

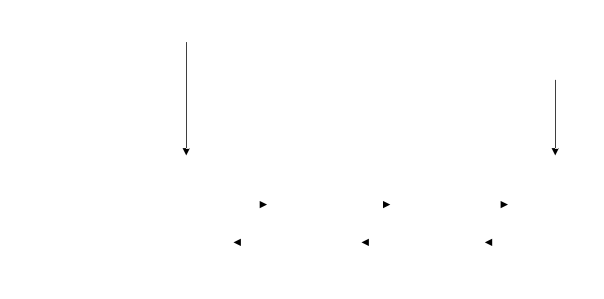
{

struct node \*first;

struct node \*last;

}HEAD;

Пример списка реализованного на базе этих структур



Переделаем функцию добавления элемента, взяв за основу добавление в односвязный список.

**Добавление элемента.**

Добавление элемента в список зависит от способа организации списка (отсортированный или нет), и правил добавления элементов (только в конец, только в начало, в определенное сортировкой место).

В общем виде разделяется на следующие шаги:

1. Выделить память под новый элемент
2. Записать данные в элемент
3. Определить место вставки и вставить
   1. Если список пуст, новый элемент становиться единственным элементом списка. Нужно создать головной элемент. Указателям на первый и последний элемент присвоить указатель на новый элемент.
   2. Если нужно вставить в начало списка
   3. Если нужно вставить в середину, тогда нужно найти элемент, после которого произвести вставку Prev и сделать 4 шага. 1)записать в новый элемент (new\_item) в поле указатель на следующий (поле new\_item–>next) тот же указатель что и в поле next элемента prev (поле prev–>next). 2) в элементе Prev изменить указатель на следующий элемент (записать адрес new\_item). 3) Предыдущим элементом для нового элемента (поле new\_item–>previos) сделать элемент prev. 4) Сделать новый элемент предыдущим для следующего за ним (поле new\_item–>next–>previos).
   4. Если нужно вставить в конец

Реализуем функцию

HEAD\* add\_node(HEAD\* head, int new\_data)

{

ITEM \*new\_item, \*prev;

new\_item=(ITEM \*)malloc(sizeof(ITEM)); /\*шаг 1 Выделить память под новый элемент\*/

if(new\_item==NULL)

{

printf("Error \n");

return head; /\*голова не поменялась\*/

}

new\_item->data=new\_data;/\* шаг 2 Записать данные в элемент\*/

if(head==NULL) /\*список пуст\*/

{

head=(HEAD \*)malloc(sizeof(HEAD)); /\*шаг 1 Выделить память под новый элемент\*/

puts("List created!");

new\_item->next=NULL;/\*следующего элемента нет\*/

new\_item->previos=NULL;/\*предыдущего элемента нет\*/

head->first=head->last=new\_item;/\*новый элемент является и первым и последним\*/

return head; /\*новая голова\*/

}

if(head->first->data>new\_data)

{

/\*список отсортирован по возрастанию и новый элемент нужно вставить в начало\*/

printf("Element %d insert in begin \n",new\_data);

new\_item->next=head->first;/\*следующий элемент после нового - текущий первый элемент списка\*/

head->first->previos=new\_item;/\*новый элемент становиться предыдущим для текущего первого элемента\*/

head->first=new\_item;/\*новый элемент становиться первым в списке\*/

new\_item->previos=NULL;/\*предыдущего элемента для него нет\*/

return head; /\*голова\*/

}

prev=head->first;

while(prev->next!=NULL) /\*пока не конец списка\*/

{

if(prev->next->data>new\_data)

{

/\*вставка\*/

printf("Element %d inserted in middle \n",new\_data);

new\_item->next=prev->next;/\*следующий элемент prev->next\*/

prev->next=new\_item;/\*теперь следующим после prev будет new\_item\*/

new\_item->previos=prev;/\*предыдущим для нового будет prev\*/

new\_item->next->previos=new\_item;/\*новый станет предыдущим для следующего за ним\*/

return head; /\*голова не поменялась\*/

}

else

{

prev=prev->next;/\*переводим указатель prev на следующий за ним элемент\*/

}

}

/\*если вышли из цикла, значит дошли до конца списка и производим вставку в конец\*/

printf("Element %d inserted in the end \n",new\_data);

head->last->next=new\_item;/\*следующим после последнего элемента станет новый\*/

new\_item->previos=head->last;/\*предыдущим для нового станет последний элемент списка\*/

head->last=new\_item;/\*новый элемент стал последним в списке\*/

new\_item->next=NULL;/\*следующего элемента после нового нет\*/

return head; /\*голова не поменялась\*/

}

Реализуем вывод списка на печать

Первая функция выводит список от первого элемента до последнего

void print\_begin(HEAD\* head)

{

ITEM \*cur;/\*curr - указаетль на текущий узел\*/

if(head==NULL || head->first==NULL)

{

puts("List empty!");

return;

}

cur=head->first;/\*начинаем с первого элемента\*/

while(cur!=NULL)/\*пока не конец списка \*/

{

printf("%d ",cur->data);

/\*переходим на следующий элемент\*/

cur=cur->next;

}

puts("");

}

Вторая функция выводит список от последнего элемента до первого

void print\_end(HEAD\* head)

{

ITEM \*cur;/\*curr - указаетль на текущий узел\*/

if(head==NULL || head->last==NULL)

{

puts("List empty!");

return;

}

cur=head->last;/\*начинаем с последнего элемента\*/

while(cur!=NULL)/\*пока не конец списка \*/

{

printf("%d ",cur->data);

/\*переходим на элемент перед ним\*/

cur=cur->previos;

}

puts("");

}

**Удаление элемента из списка.**

При удалении существуют два варианта:

* Удаление первого элемента
* Удаление последнего элемента
* Удаление любого другого элемента

Реализация функции

HEAD\* delete\_node(HEAD\* head, int delete\_data)

{

ITEM \*cur, \*prev;/\*curr - указаетль на текущий узел, prev - указатель на предыдущий узел\*/

prev=NULL;

cur=head->first;/\*начинаем с головного элемента\*/

while(cur!=NULL && cur->data<=delete\_data)/\*пока не конец списка и элемент еще может встретиться в списке\*/

{

if(cur->data==delete\_data)/\*проверяем текущий элемент\*/

{

/\*удаление\*/

if(prev==NULL)

{

printf("Delete first %d \n",cur->data);

/\*удаление первого\*/

head->first=cur->next;/\*первым становиться следующий элемент после первого\*/

if(head->first!=NULL)

head->first->previos=NULL;

if(head->first==NULL)

head->last=NULL;

}

else if(cur==head->last)

{

printf("Delete last %d \n",cur->data);

/\*удаление последнего\*/

head->last=prev;/\*последним становиться элемент перед удаляемым\*/

prev->next=NULL;/\*следующего не будет\*/

}

else

{

/\*удаление любого другого элемента\*/

prev->next=cur->next;

cur->next->previos=prev;

}

free(cur);/\*освобождаем память\*/

return head;

}

/\*иначе переходим на следующий элемент\*/

prev=cur;

cur=cur->next;

}

printf("Element %d not found \n",delete\_data);

return head;/\*голова не поменялась\*/

}

Основная функция

int main()

{

HEAD \*Head=NULL;

char key;

int chislo;

while(1)

{

clrscr();

puts("1 - insert element");

puts("2 - delete element");

puts("3 - print");

puts("ESC - exit");

key=getch();

switch(key)

{

case '1':

puts("Enter number");

scanf("%d",&chislo);

Head=add\_node(Head,chislo);

getch();

break;

case '2':

puts("Enter number");

scanf("%d",&chislo);

Head=delete\_node(Head,chislo);

getch();

break;

case '3':

print\_begin(Head);

print\_end(Head);

getch();

break;

case 27:

puts("Exit");

getch();

return 0;

}

}

return 0;

}