Лабораторна робота 8 (С4)

"Інформаційні динамічні структури"

Ціль:Знайомство з динамічними інформаційними структурами з прикладу одно- і двунаправленных (пов'язаних) списків.

Постановка задачі

Написати програму, в якій створюються динамічні структури та виконати їхню обробку відповідно до свого варіанта.

Для кожного варіанта розробити такі функції (узгоджується з викладачем):

1. Створення списку.
2. Додавання елемента до списку (відповідно до свого варіанта).
3. Видалення елемента зі списку (відповідно до свого варіанта).
4. Друк списку.
5. Запис списку файлу.
6. Знищення списку.
7. Відновлення списку із файлу.

Порядок виконання роботи

1. Написати функцію створення списку. Функція може створювати порожній список, а потім додавати до нього елементи.
2. Написати функцію друку списку. Функція повинна передбачати виведення повідомлення, якщо список є порожнім.
3. Написати функції для видалення та додавання елементів списку відповідно до свого варіанта.
4. Здійснити зміни у списку та друк списку після кожної зміни.
5. Написати функцію для створення списку файлу.
6. Написати функцію для знищення списку.
7. Записати список у файл, знищити його та виконати друк (при друку має бути видане повідомлення "Список порожній").
8. Написати функцію відновлення списку з файла.
9. Відновити список та роздрукувати його.
10. Знищити список.

Навчальні приклади роботи зі списками можна переглянути

тут один зв'язаний список

тут приклад одного зв'язаного списку

тут двох пов'язаний список

Варіанти завдань

1. Записи в списку ліній містять ключове поле типу int. Сформувати однонаправлений перелік. Видалити з нього елемент із заданим номером, додати елемент із заданим номером;

2. Записи в списку ліній містять ключове поле типу int. Сформувати однонаправлений перелік. Видалити з нього елемент із заданим ключем, додати елемент перед елементом із заданим ключем;

3. Записи в списку ліній містять ключове поле типу int. Сформувати однонаправлений перелік. Видалити з нього До елементів, починаючи із заданого номера, додати елемент перед елементом із заданим ключем;

4. Записи в списку ліній містять ключове поле типу int. Сформувати однонаправлений перелік. Видалити з нього елемент із заданим номером, додати До елементів, починаючи із заданого номера;

5. Записи в списку ліній містять ключове поле типу int. Сформувати однонаправлений перелік. Видалити з нього До елементів, починаючи із заданого номера, додати До елементів, починаючи із заданого номера;

6. Записи в списку ліній містять ключове поле типу int. Сформувати двонаправлений перелік. Видалити з нього елемент із заданим номером, додати елемент до початку списку.

7. Сформувати двонаправлений перелік. Видалити з нього перший елемент, додати елемент до кінця списку.

8. Записи в списку ліній містять ключове поле типу int. Сформувати двонаправлений перелік. Видалити з нього елемент після елемента із заданим номером, додати До елементів на початок списку.

9. Записи в списку ліній містять ключове поле типу int. Сформувати двонаправлений перелік. Видалити з нього До елементів перед елементом із заданим номером, додати До елементів до кінця списку.

10. Записи в списку ліній містять ключове поле типу int. Сформувати двонаправлений перелік. Додати до нього елемент із заданим номером, видалити До елементів із кінця списку.

11. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити з нього елемент із заданим ключем, додати елемент із зазначеним номером.

12. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити з нього елементи з однаковими ключовими полями. Додати елемент після елемента із заданим ключовим полем.

13. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити з нього перші елементи. Додати елемент після елемента, що починається із зазначеного символу.

14. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити з нього До елементів із зазначеними номерами. Додати До елементів із зазначеними номерами.

15. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити До елементів із кінця списку. Додати елемент після елемента із заданим ключем.

16. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити елемент із заданим ключем. Додати До елементів у кінець списку.

17. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити елемент із заданим номером. Додати До елементів на початок списку.

18. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити елемент із заданим ключем. Додати До елементів на початок списку.

19. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити До елементів із заданими номерами. Додати До елементів на початок списку.

20. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити елемент із заданим ключем. Додати до елементів у початок і в кінець списку.

21. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити елементи перед та після елемента із заданим ключем. Додати до елементів у початок і в кінець списку.

22. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити елемент із заданим ключем. Додати До елементів перед елементом із заданим ключем.

23. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити елемент із заданим ключем. Додати До елементів після елемента із заданим ключем.

24. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити елемент із заданим номером. Додати до елементів перед і після елемента із заданим ключем.

25. Записи в лінійному списку містять ключове поле типу *char(рядок символів). Сформувати двонаправлений перелік. Видалити елемент із заданим ключем. Додати До елементів перед елементом із заданим номером.

Зміст звіту

1. Постановка задачі.
2. Функції роботи зі списком.
3. Функція main().
4. Результати виконання.

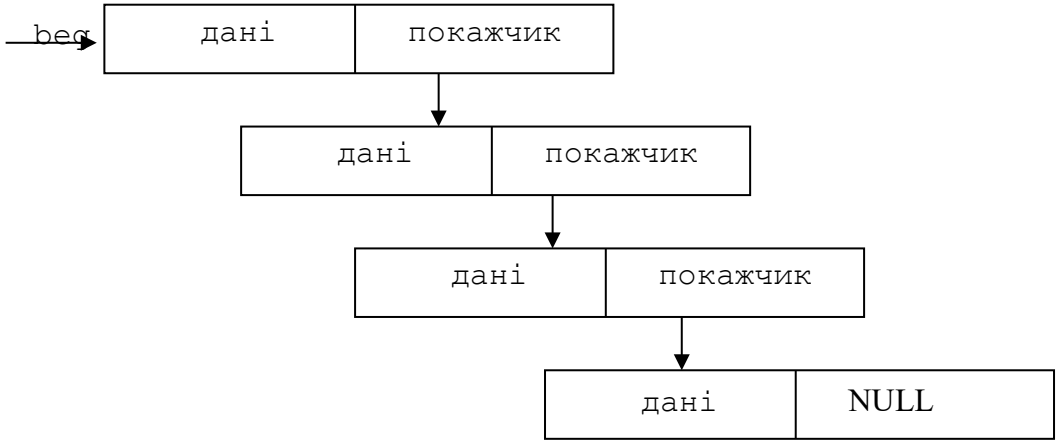
Один зв'язковий список

У багатьох завданнях потрібно використовувати дані, у яких конфігурація, розміри, склад можуть змінюватися в процесі виконання програми. Для їхнього подання використовують динамічні інформаційні структури.

Найпростіша інформаційна структура – це однозв'язковий список, елементами якого є об'єкти структурного типу. Наприклад

```
struct ім'я_структурного_типу
{
   елементи_структури_дані;
   ім'я_структурного_типу *покажчик;
}
```

У кожну структуру такого типу входить покажчик на об'єкт того ж типу, що й структура, що визначається.



Приклади:

1. Опис структури

```
struct point
{
   int key;
   point * next;
};
```

Поле key містить інформаційну частину структури point, а поле next містить адресу наступного списку.

2. Функція на формування односпрямованого списку

```
point* make_point( int n)
{
```

```
   point *first, *p;
   first = NULL;
   for (int i=n; i>0; i--)
    {
       p = new (point);
       p-> key = i;
       p-> next = first;
       first = p;
     }
   return first;
}
```

Як параметр до функції передається кількість елементів у списку, а результатом є покажчик на перший елемент цього списку. Покажчик p вказує на новостворюваний елемент.

Для звернення до полів використовується операція доступу до елемента структури, з якою пов'язаний покажчик ->.

Існує друга можливість звернення до поля динамічної структури: (*p).key або (*p).next. В інформаційному полі key заноситься порядковий номер елемента у списку. Додавання нових елементів здійснюється на початок списку.

3. Функція для друку однонаправленого списку

```
point* print_point(point*first)
{
  if (first==NULL) return NULL;
  point * p = first;
  while(p!=NULL)
{
      cout << p->key << "\n";
      p = p->next;
   }
  return first;
}
```

При друку сформованого списку здійснюється прохід по списку за допомогою допоміжної змінної p до тих пір, поки вона не дорівнюватиме NULL.

Навчальний приклад програми

```
#include <string.h>
#include <stdio.h>

  struct ss // структура ланки списку
```

```
   {int n;
     char z [80];
     ss *next;
};
     int main(int argc, char* argv[])
{
         ss*top; // покажчик початку
         ss*tek; // покажчик поточний
         top=new ss();
         top->n=1;
         strcpy(top->z,"zap-01");
         top->next = NULL;
         printf("top=%4x\n", top);
            for (int i=1;i<=10;i++) // створюємо 10 ланок
{
              tek=top;
              while(tek->next!=NULL) // демонструємо пошук ОСТАННЬОЇ ланки
                tek=tek->next;
              // додаємо ланку
              tek->next=new ss();
tek=tek->next;
              tek->n=i;
              sprintf(tek->z,"zap-%02d",i);
              tek-> next = NULL;
};
/////друк
         printf("після створення списку\n");
         tek=top;
         int k=1;
            while(tek->next! = NULL) // обхід всіх ланок списку
              { printf("tek=%4x zap=%2d {n=%2d z=<%s> next=%4x}\n",
                        tek, k, tek->n, tek->z, tek->next);
                tek=tek->next;
                k++;
};
/////пошук
k=1;
          int find=5;
          ss * zfind = NULL;
```

```
          tek=top;
            while(tek->next!=NULL) // пошук ланки у списку
              { if(tek->n == find)
                 {zfind=tek;
                   printf("find=%2d tek=%4x tek->n=%2d tek->z=<%s> tek->next=%4x\n",
find, tek, tek->n, tek->z, tek->next);
                   break;
};
                tek=tek->next;
                k++;
};
          if(zfind==NULL)
            printf("not found.....find=%d\n", find);
          else
printf("!!! found find=%d\n", find);
            printf("zfind=%4x\n", zfind);

/////видалення ланки

     ss * last;
     ss * zdel = NULL;
     tek=top;
     last=tek;
       while (tek->next! = NULL)
         { if (tek->n==5) // що шукаємо ......
{
              zdel = tek;
              break;
};
          last=tek;
          tek=tek->next;
};
            if(zdel!=NULL)
             { if (top==tek)
{delete top;
                top = NULL;
printf("delete 1-st zap --- spis is lost......\n");
                return 555;
};
```

```
                if(zdel->next==NULL)
                  { last->next = NULL;
                    delete zdel;
}
                else
{ last->next=zdel->next;
                    delete zdel;
};
};
/////друк

         printf("після видалення 5-го запису\n");
         tek=top;
k=1;
            while(tek->next! = NULL)
              { printf("tek=%4x zap=%2d {n=%2d z=<%s> next=%4x}\n",
                        tek, k, tek->n, tek->z, tek->next);
                tek=tek->next;
                k++;
};

   /// вставка нового запису.
   // ss * last;
     zfind=NULL;
     tek=top;
     last=tek;
     ss*newadd;
       while (tek->next! = NULL)
         { if (tek->n==6) // що шукаємо
{
              zfind=tek;
              break;
};
          last=tek;
          tek=tek->next;
};
       // zfind add next zap
       if (zfind! = NULL)
{
```

```
            printf("!!! found zfind=%d\n", zfind);
            printf("zfind->n=%2d zfind->z=<%s> zfind->next=%4x\n",
                           zfind->n, zfind->z, zfind->next);
//getchar();// return 6;


           newadd = new ss();
           strcpy(newadd->z,"новий запис після 6 zap ****");
           newadd->next=zfind->next;
           zfind->next = newadd;
}
       else printf("zap not found,,,,,\n");

/////друк
         printf("після вставки нового запису списку\n");

         tek=top;
k=1;
            while(tek->next! = NULL)
              { printf("tek=%4x zap=%2d {n=%2d z=<%s> next=%4x}\n",
                        tek, k, tek->n, tek->z, tek->next);
                tek=tek->next;
                k++;
};

/*****************************************************/
///// видалення списку з пам'яті
       ss*temp;
       tek=top;
       while(tek->next! = NULL)
{temp=tek;
         tek=tek->next;
         delete temp;
};

     return 0;
}


C:\Test\cpp\!! приклад програми по структурах> spis-1
```

```
top=842d60
після створення списку
tek=842d60 zap=1 {n=1 z=<zap-01> next=842dbc}
tek=842dbc zap=2 {n=1 z=<zap-01> next=842e18}
tek = 842e18 zap = 3 {n = 2 z = <zap-02> next = 842e74}
tek=842e74 zap=4 {n=3 z=<zap-03> next=842ed0}
tek = 842ed0 zap = 5 {n = 4 z = <zap-04> next = 842f2c}
tek = 842f2c zap = 6 {n = 5 z = <zap-05> next = 842f88}
tek = 842f88 zap = 7 {n = 6 z = <zap-06> next = 842fe4}
tek = 842fe4 zap = 8 {n = 7 z = <zap-07> next = 843040}
tek=843040 zap=9 {n=8 z=<zap-08> next=84309c}
tek=84309c zap=10 {n= 9 z=<zap-09> next=8430f8}
find= 5 tek=842f2c tek->n= 5 tek->z=<zap-05> tek->next=842f88
!!!found find=5
zfind=842f2c
після видалення 5-го запису
tek=842d60 zap=1 {n=1 z=<zap-01> next=842dbc}
tek=842dbc zap=2 {n=1 z=<zap-01> next=842e18}
tek = 842e18 zap = 3 {n = 2 z = <zap-02> next = 842e74}
tek=842e74 zap=4 {n=3 z=<zap-03> next=842ed0}
tek = 842ed0 zap = 5 {n = 4 z = <zap-04> next = 842f88}
tek = 842f88 zap = 6 {n = 6 z = <zap-06> next = 842fe4}
tek = 842fe4 zap = 7 {n = 7 z = <zap-07> next = 843040}
tek = 843040 zap = 8 {n = 8 z = <zap-08> next = 84309c}
tek=84309c zap=9 {n=9 z=<zap-09> next=8430f8}
!!!found zfind=8662920
zfind->n=6 zfind->z=<zap-06> zfind->next=842fe4
після вставки нового запису списку
tek=842d60 zap=1 {n=1 z=<zap-01> next=842dbc}
tek=842dbc zap=2 {n=1 z=<zap-01> next=842e18}
tek = 842e18 zap = 3 {n = 2 z = <zap-02> next = 842e74}
tek=842e74 zap=4 {n=3 z=<zap-03> next=842ed0}
tek = 842ed0 zap = 5 {n = 4 z = <zap-04> next = 842f88}
tek=842f88 zap=6 {n=6 z=<zap-06> next=842f2c}
tek=842f2c zap= 7 {n=8651072 z=<новий запис після 6 zap ****> next=842fe4}
tek = 842fe4 zap = 8 {n = 7 z = <zap-07> next = 843040}
tek=843040 zap=9 {n=8 z=<zap-08> next=84309c}
tek=84309c zap=10 {n= 9 z=<zap-09> next=8430f8}
```

C:\Test\cpp\!! приклад програми по структурам>

Тут - майже закінчений приклад

```
#include <iostream>
using namespace std;
/*************
C++: Однозв'язний список в консольному додатку
Тут - майже закінчений приклад, що робить наступне:
описати структуру,
сформувати динамічний однозв'язний список структур;
реалізуе перегляд списку,
додавання елементів на початок і кінець,
видалення (по ключу або номеру запису),
сортування
я що треба додати записи з клавіатури.
************************/

struct person {// опис структури
char f [30]; //прізвище
char d [30]; //Посада
float zp,pr; // зарплата, премія
char prof; //профспілка
person *next; //покажчик на наступного у списку};

person vvesti() { //введення елемента з клавіатури
person p;
printf ("\nПІБ?"); fflush (stdin); scanf ("%s", pf);
printf ("Посада?"); fflush (stdin); scanf ("%s", pd);
printf ("Зарплата?"); fflush (stdin); scanf ("%f",&p.zp);
printf ("Премія"); fflush (stdin); scanf ("%f",&p.pr);
printf ("Член профоюзу (+/-)?"); fflush (stdin); scanf ("%c", p.prof);
return p;}
```

```
int show (person *head) { // показ всього списку з визначенням кількості елементів
int count = 0;
if (head) while (1) {
printf ("\n%s %s %.2f+%.2f %c", head->f,head->d,head->zp,head->pr,head->prof);
count++;
if (head->next == NULL) break;
head = head->next;}
printf ("\nУ списку всього %d зап. (лан.)\n",count);
return count;}
void search (person *head, char *st) { //Пошук у списку рядка st
if (head==NULL) return;
person *next = head;
do {
char * find = strstr (next-> f, st);
if (find!=NULL) printf ("\n%s",next->f);
if (next->next==NULL) break;
next = next->next;} while (1);}

void copy1 (person *to, person *from) { //копіювання одного запису
strcpy (to->f, from->f);
strcpy (to->d, from->d);
to->zp = from->zp; to->pr = from->pr; to->prof = from->prof;}

person *add1 (person *head, person *st) { //додавання елемента st на початок списку//*st вже заповнений
даними
person * current = new person; copy1 (current, st); //Копіює 1 запис
if (head==NULL) current->next=NULL;
else {
current->next = head;
head = current;}
return current;}

person *add2 (person *head, person *st) { //додавання елемента st в кінець списку
```

```
людина *last = NULL;
if (head!=NULL) {
last=head;
while (last->next!=NULL) last=last->next;}
person * current = new person; copy1 (current, st);
current->next = NULL;
if (last) last->next = current;
return current;}

person *delete1 (person *head0, int n) { //видалення елемента за номером 1..N// видалити елемент номер n
і поверне указ.на поч.
person *head = head0;
if (head==NULL) return NULL;
if (n==1) { // видаляємо перший
person *ptr = head->next;
delete head;
return ptr;}
person *prev=NULL, *start=head; int i=1;
while (i<n) { //Шукаємо n-ий елемент
prev = head; head = head->next;
if (head==NULL) return start;i++;}
person *ptr = head->next;
delete head;

prev->next = ptr;
return start;}

person * sort (person * ph) { // Сортування списку методом вставок
person *q, *p, *pr, *out=NULL;
while (ph! = NULL) {q = ph; ph = ph->next; // виключити ел-т // шукаємо, куди його включити:
for (p=out,pr=NULL ; p!=NULL &&
strcmp(q->f,p->f)>0; pr=p,p=p->next) ;//або ваш критерій, коли переставляти!
if (pr==NULL) { q->next=out;out=q; } //в поч.
```

```
else {q->next = p; pr-> next = q; } //після перед.}
return out;}

int main() { // Меню та головна програма
person *head = NULL;
system("chcp 1251");
while (1) {
printf ( "\n 1. Показати все (Show all)""\n 2. Додати на початок (Add in head)""\n 3. Додати в кінець
(Add in tail)""\n 4. Видалити за номером (Delete by number)""\n 5. Сортувати за назвою (Sort by
name)""\n;
fflush (stdin);
char c;
scanf ("%c",&c);
person p;
person *cur;
int n, all;
switch (c) {
case '1': show (head);
break;
case '2': p = vvesti(); head=add1(head,&p);
break;
case '3': p = vvesti();
cur = add2 (head, & p);
if (head==NULL) head=cur;
break;
case '4':
all = show (head);
while (1) {
printf ("\nВведіть номер запису, що видаляється (1-%d): ",all);
fflush (stdin); scanf("%d",&n);
if (n>=1 && n<=all) break;}
head = delete1 (head, n);
break;
```

```
case '5': head = sort (head);
break;
case '0': exit (0); break;}}}
```

Двох пов'язаний список

Нижче наведено текст навчальної програми роботи зі зв'язаним списком. Приймаючи ім'я файлу як параметр командного рядка, програма читає вміст зазначеного файлу і з урахуванням записів формує двох пов'язаний список. Ланка списку задається структурою sp. Список роздруковується у прямому та зворотному "напрямку" і "знищується".

```
#include <conio.h>#include <stdio.h>#include <string.h>

   struct sp
   {sp*a_up; // покажчик вище варте ланка
    sp * a_down; // покажчик на нижче ланка
    char info [81]; // інформаційна частина ланки};

   int main (int argv, char * * argc) {int k_zap = 0, k;
     char z; char nfile [80], buf [80];
     FILE * h; sp *first, *a_tek, *p, *a_next;
     if (argv > 1) strcpy(nfile,argc[1]);
     else { printf("При виклику програми не вказано ім'я файлу для читання.....\n"); return -2;};

   // ініціалізація списку.....
   first=NULL;//======
   h=fopen(nfile,"r"); if(h==NULL) { printf( "Cannot open input file <%s> \n", nfile); return -1; };//======

   next:
    if ( fgets(buf, 80, h) == NULL )
      {
        printf( "Кінець файлу <%s> \n", nfile); goto endd; };
        k = strlen (buf) -1;
        if (k>=0) buf[k]='\0';
        k_zap++;
        printf("прочитали %d <%s>\n", k_zap, buf);

       // помістити новий запис з buf до списку
        if (first == NULL ) { p = new sp; // адреса нової ланки
```

```
     p-> a_up = NULL; // у новій ланці покажчики вгору й униз - порожні
     p->a_down=NULL;
     p->info[0]='\0'; // У новій ланці інф. рядок порожній
     strcpy(p->info, buf); / / Запис з буфера в інф. рядок списку
     first = p; //Голова списку
     printf("first=\n",first);
     printf("<%p><%s>\n",p->a_up,p->a_down,p->info);
 }
  else {// пошук останньої ланки (показано для прикладу, на практиці в цьому немає необхідності)
     printf("*** пошук останньої ланки**\n");
     a_tek = first; // адреса поточної (першої) ланки
     p = a_tek-> a_down; // адресу "вниз" з поточної ланки в покажчику p
     printf("***поточна ланка <%p><%s>\n", a_tek->a_up, a_tek->a_down, a_tek->info);
     printf("***ук. на слід. \n", p);
     while (p! = NULL) {a_tek = p; // адреса наступної ланки
                        p = a_tek-> a_down; // адресу "вниз" з поточної ланки в покажчику p
                        };
    // в a_tek - адреса останньої ланки
    p = new sp; // адреса нової ланки
    printf("tek=<%p> new=<%p>\n", a_tek, p);
    a_tek->a_down = p; //посилання вниз у передостанньому ланці
    p->a_up= a_tek; //посилання нагору в останній (новій) ланці
    p->a_down=NULL;
    strcpy(p->info, buf); printf("<%p><%s>\n",p->a_up,p->a_down,p->info);
};
goto next;
endd:
 fclose(h);
 getchar();// роздрук списку в прямому напрямку ......
 clrscr();
 a_tek = first;
 printf("first=\n",first);p = a_tek; // -> a_down;
 while (p! = NULL) {
    printf("ADD=<%p> ZVENO=<%p><%s>\n", p, p->a_up, p->a_down, p->info);
    getchar();
   a_tek = p; p = a_tek-> a_down; };
getchar();
```

```
// Пошук останньої ланки(Показано для прикладу, на практиці в цьому немає потреби)
printf("***пошук останньої ланки**\n");
a_tek = first; // адреса поточної (першої) ланки
p = a_tek-> a_down; // адресу "вниз" з поточної ланки в покажчику p
printf("***поточна ланка <%p><%s>\n",a_tek->a_up, a_tek->a_down, a_tek->info);
printf("***ук. на слід. \n", p);
while (p! = NULL) {a_tek = p; // адреса наступної ланки =
   a_tek-> a_down; // адресу "вниз" з поточної ланки в покажчику p
                 }; // в a_tek - адресу останньої ланки
// Роздрук списку у зворотному напрямку
p=a_tek->a_up; // адреса "вгору"
printf("*** останнє зв.<%p><%s>\n",a_tek->a_up, a_tek->a_down, a_tek->info);
printf("***ук. на слід. вгору \n", p);

while (p! = NULL) {
  printf("ADD=<%p> ZVENO=<%p><%s>\n", p, p->a_up, p->a_down, p->info);
  getchar();
  a_tek = p; p = a_tek-> a_up;
};
  getchar();

// видалити список із пам'яті // пошук останньої ланки
a_tek = first; p = a_tek-> a_down;
while (p! = NULL) {
  a_tek = p; p = a_tek-> a_down;
};
// в a_tek - адреса останньої ланки
do{// адреса попередньої ланки
  a_next=a_tek->a_up;
  delete a_tek;
  a_tek = a_next; }
while (a_tek! = NULL);
printf("усієї\n");
return 0;}
```