Лекція №10. Списки. Стеки. Черги

ПРОГРАМУВАННЯ ТА ПРИКЛАДНІ ІНФОРМАЦІЙНІ СИСТЕМИ

Список

Список - **це скінчений набір даних одного типу**, між якими налагоджено зв'язок. Елемент однонаправленого списку **складається з двох частин**: **самого даного (часто складеного)** та **вказівника на наступний елемент** списку. Для опису списку використовують тип даних структура і вказівник.

Опис списку

struct <назва типу списку>{

    <тип поля 1><назва поля 1>;

    …

    <тип поля n><назва поля n>;

    <назва типу списку> *<назва вказівника>;

};

<назва типу списку> *<назва вказівника 1>, ...,
 *<назва вказівника n>;

Приклад 1

Створимо структуру про річку (rika), яка містить поля: назва, довжина річки у кілометрах та площа басейну у квадратних кілометрах, і поставимо їй у відповідність елементи списку:

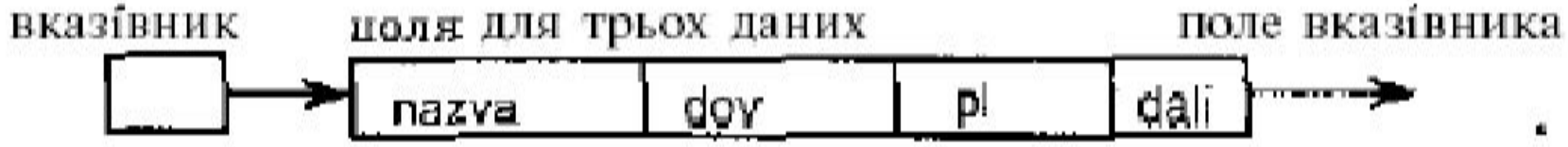
```
struct rika {
    char nazva[20];
    int dov;
    long int pl;
    rika *dali;
};
rika *element;
```

Приклад 1

Тут **element** - вказівник (тип структура rika) на поточний елемент списку, **element -> dov** - динамічна змінна цілого типу (int), яка містить значення довжини річки, а **element->dali** – вказівник на наступний елемент списку. Звідси випливає, що **element->dali -> dov** - це довжина наступної річки, а **element-> dali -> dali** - вказівник на ще наступну річку і т.д.

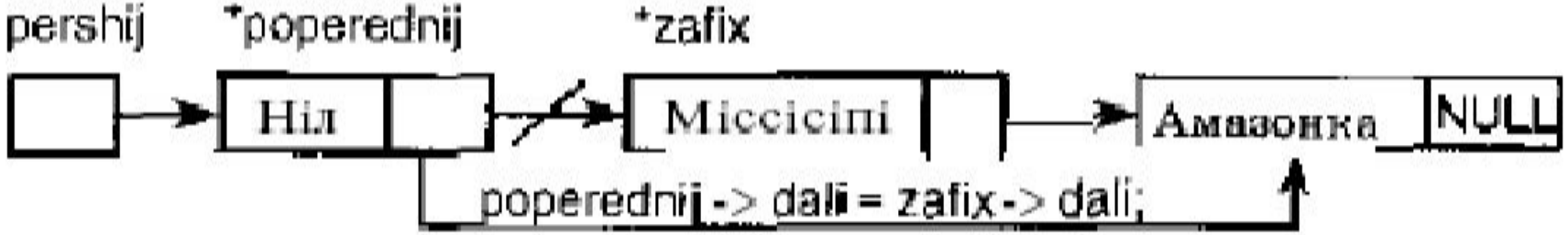
Приклад 2

Задача 1 (про річки). Утворити список, який містить інформацію про річки. Вивести цей список на екран. Додати на початок списку новий запис. Вивести список зі змінами.

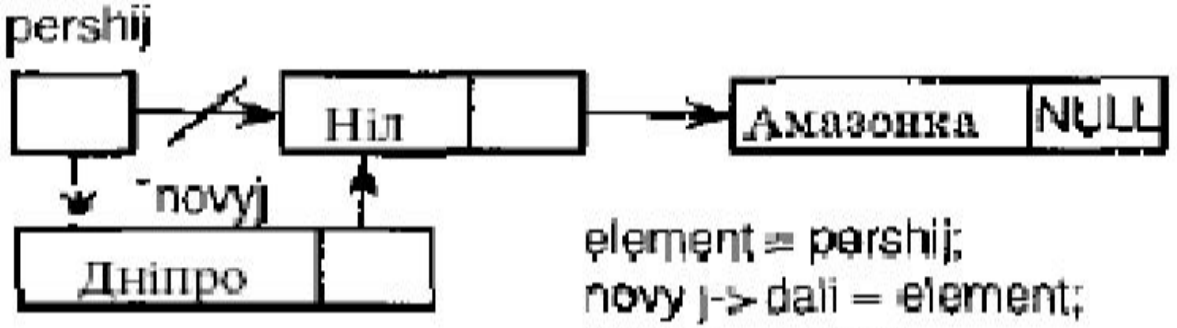


*а) графічне зображення елемента *element списку;*

б) список річок (даних про річки) з трьох елементів;

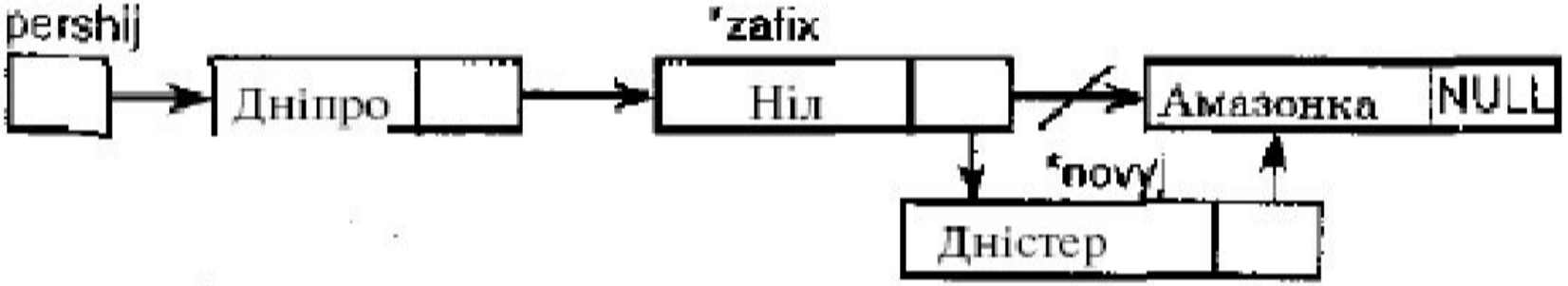


*в) вилучення даних про Miccicini (елемент *zafix) зі списку;*



element = pershij;
novyj -> dali = element;
pershij = novyj;

г) перехід до початку списку і долучення даних про річку Дніпро;



novyj -> riali = zafix -> dali;
zafix -> dali = novyj;

*д) долучення даних про річку Дністер (елемент *novyj) у список*

```
struct rika {           //Створюємо тип rika
    char nazva[20];
    int dov;
    long int pl;
    rika *dali;  //Створюємо поле вказівник типу rika

};
rika *element, *pershij, *poperednij, *novyj;
void StvorytySpisok();
void VyvestiNaEkran();
void StvorytyNovijElement();
```

```
void main(){
    cout << "Stvorennja spisku\n";
    cout << "vvedit nuli dlja zakin4ennja\n";
    StvorytySpisok();
    VyvestiNaEkran();
    StvorytyNovijElement();
    element = pershij;
    novyj->dali = element;
    pershij = novyj;
    cout << "Novij Spisok\n";
    VyvestiNaEkran();
    system("pause");
}
```

```
void StvorytySpisok(void){
    element = new(rika);
    pershij = element;
    do {
        poperednij = element;
        cout << "Vvedite nazvu, dovzinu, plowad\n";
        cin >> element->nazva;
        cin >> element->dov;
        cin >> element->pl;
        element->dali = new(rika);
        element = element->dali;
    } while (poperednij->dov != 0 || poperednij->pl != 0);
    poperednij->dali = NULL;
}
```

```
void VyvestiNaEkran(void){
    cout << "Stvorenij spisok\n";
    element = pershij;
    while (element != NULL){
        cout << element->nazva << "\t" << element->dov << "\t"
        << element->pl << "\n";
        element = element->dali;
    }
}

void StvorytyNovijElement(void){
    novyj = new(rika);
    cout << "Uvedit nazvu, dovzinu ta plowu novoi ri4ki\n";
    cin >> novyj->nazva >> novyj->dov >> novyj->pl;
}
```

Стек

Стек - це структура даних, у якій елемент, записаний останнім, зчитують (він доступний для опрацювання) першим. Принцип "**останній прийшов - перший пішов**" використовується в багатьох технічних пристроях і в побуті: ріжок від автомата; посадка пасажирів у вагон, який має лише одні двері тощо. Стек використовують у програмуванні, зокрема, для реалізації рекурсії. Рекурсія виконується так: спочатку всі виклики нагромаджуються (аналогія така: пружина стискається), а потім виконуються вкладені функції (пружина розпрямляється).

Створення стеку

Стек описують і створюють у пам'яті за допомогою **типу даних структура**. Над елементами стека визначені лише **дві операції**: **занесення елемента у стек** та **вилучення елемента зі стека**. У стеку завжди доступним є лише верхній елемент, який називають ***вершиною стека***. Розглянемо типову задачу роботи зі стеком.

Задача 2

Увести послідовність символів, де крапка (".") є ознакою закінчення введення. Вивести введені символи на екран у зворотному порядку. Розв'яжемо цю задачу із застосуванням стека (stack), який містить такі поля: символ (ch), вказівник на наступний елемент стека (dali).

```
struct stack {
    char ch;
    stack *dali;
};
```

```
stack *st, *element;
void StvorytyStek(stack *st);
void VyluchenniaZiSteku(stack *st);

void main(){
    st = NULL;
    cout << "Vvedit stroku\n";
    StvorytyStek(st);
    cout << "Naoborot\n";
    VyluchenniaZiSteku(st);
    cout << endl;
    system("pause");
}
```

```
void StvorytyStek(stack *st){
    char a;
    do{
        cin >> a;
        element = new(stack);
        element->dali = st;
        st = element;
        element->ch = a;
    } while (a != '.');
}
void VyluchenniaZiSteku(stack *st){
    do{
        st = element->dali; element = st;
        cout << st->ch;
    } while (st->dali != NULL);
}
```

Черга

Черга - це структура даних, у якій елемент, записаний першим, зчитують першим. Тут діє принцип **"перший прийшов - перший пішов"**, добре відомий з побуту: черга у магазині тощо.

Чергу, як і стек, **описують з використанням структури**.

Над елементами черги визначені операції: **занесення елемента у чергу** та **забираня з черги**. У черзі доступним є лише нижній елемент.