Лабораторна робота №2.

Стеки. Реалізація стеків за допомогою масивів та покажчиків.

Мета роботи: Закріпити техніку роботи з напівстатичними структурами даних на прикладі стеку. Розглянути основні операції зі стеком і познайомитись з типовими прикладами застосування стеків.

Теоретичні відомості.

Стек – це спеціальний тип списку, в якому всі операції включення і виключення виконуються на одному кінці, що називається *вершиною*, тому стек називають списком типу LIFO (last-in-first-out – останнім включається – першим виключається). Елементи додаються у вершину (голову) стеку; елементи видаляються з вершини (голови) стеку; покажчик в останньому елементі дорівнює nil; неможливо вилучити елемент із середини стеку, не вилучивши всі елементи, що йдуть попереду.

Стек можна організувати на базі будь-якої структури даних, де можливе зберігання декількох однотипних елементів і де можна реалізувати визначення стека: лінійний масив, типізований файл, однонаправлений або двонаправлений список. Над стеком і його елементами виконуються такі типові операції: додавання елемента в стек; видалення елемента зі стеку; перевірка, чи порожній стек; перегляд елемента у вершині стека без видалення; очищення стека.

Реалізація стеку за допомогою масиву.

Існує декілька фізичних представлень стеків за допомогою масивів (рис.1, рис.2, рис.3).

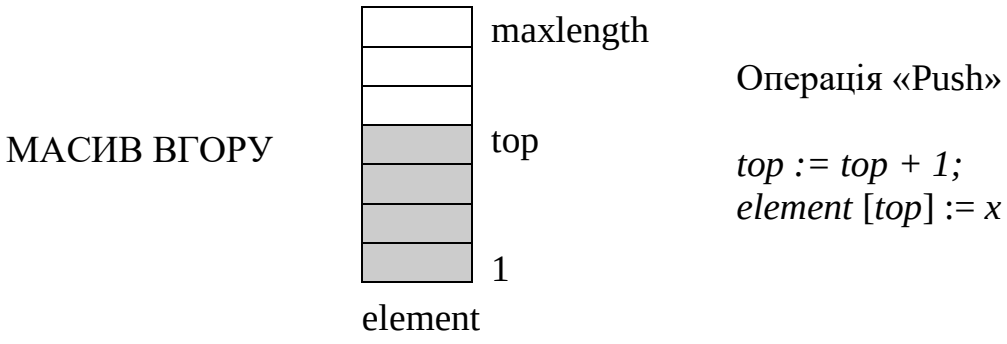


Рис.1. ≠

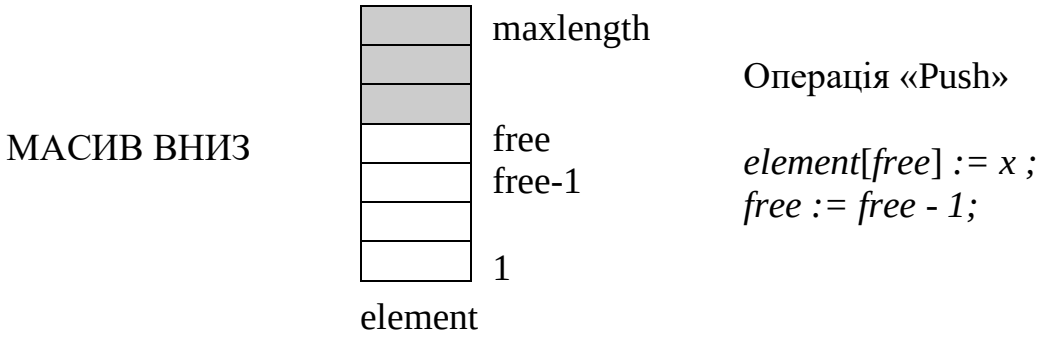


Рис. 2. Представлення стеків за допомогою масиву вниз

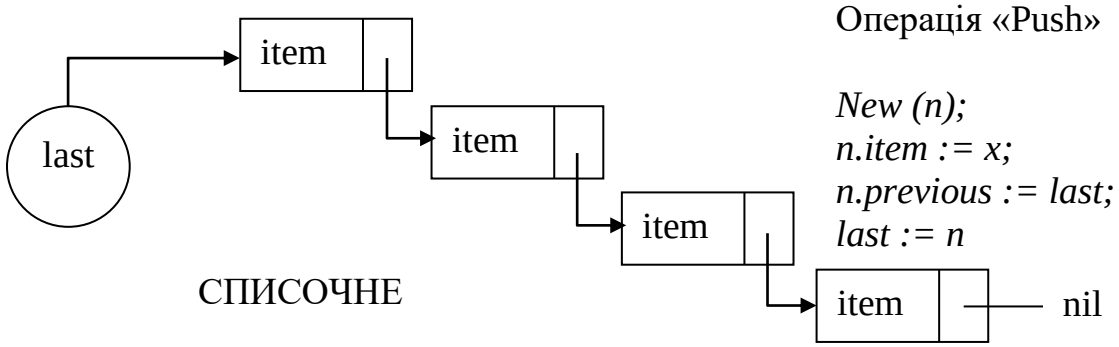


Рис. 3. Списочне представлення стеків

1. МАСИВ ВГОРУ – представлення стеку за допомогою масиву *element* і цілого числа *top* з діапазоном значень від 0 (для пустого стеку) до *maxlength* (для заповненого стеку). Елементи стеку зберігаються в масиві та індексуються від 1 до *top*.

2. МАСИВ ВНИЗ схожий на МАСИВ ВВЕРХ, але елементи розміщуються в кінець стеку, а не в початок. Число *free* є індексом верхньої вільної позиції в стеку або 0, якщо всі позиції в масиві зайняті, і змінюється в діапазоні від *maxlength* (для порожнього стеку) до 0 (для заповненого). Елементи стеку зберігаються в масиві й індексуються від *maxlength* до *free*+1.

3. При СПИСОЧНОМУ представленні кожний елемент стеку зберігається у вузлі з двома полями: *item*, що вміщує сам елемент, і *previous*, що вміщує покажчик на вузол з попереднім елементом. Для такого представлення потрібен покажчик *last* на вузол, що вміщує вершину стека.

Поряд з кожним представленням на рисунку наведено фрагмент програми з відповідною реалізацією основної стекової операції *push* включення елемента *x* у вершину стеку).

Для представлення за допомогою масивів команди збільшують або зменшують покажчик на вершину (*top* або *free*) і записують *x* у відповідний елемент масиву. Так як ці представлення підтримують стеки з не більше чим *maxlength* елементами, то коректні реалізації повинні містити тести, які захищають від переповнення, виду:

If *top* = maxlength **then**...

if*free* = 0 **then** ...

СПИСОЧНЕ представлення для включення елемента потребує чотирьох дій: створення нового вузла *n*; присвоювання *x* полю *item* нового вузла; приєднання нового вузла до вершини стеку шляхом надання його полю *previous* поточного значення покажчика *last*; змінювання *last* так, щоб він посилався на щойно створений вузол.

Хоча ці представлення зустрічаються найчастіше, існує і багато інших представлень стеків. Наприклад, якщо потрібні два стеки з однотипними елементами і пам'ять для них обмежена, то можна використовувати один массив з двома мітками вершин *top* як в представленні МАСИВ ВГОРУ і *free* як в МАСИВ ВНИЗ. При цьому один стек буде рости вгору, а інший вниз (рис.4). Умовою повного заповнення цього представлення є рівність *top* = *free*.

Перевага такого представлення полягає у зменшенні ризику переповнити пам'ять: при двох масивах розміру *n*, що представляють стеки способом МАСИВ ВГОРУ або МАСИВ ВНИЗ, пам'ять вичерпується, коли лише будь який зі стеків досягне *n* елементів. А у випадку одного масиву розміру 2*n*, що містить два стеки один напроти другого, робота продовжується до тих пір, доки їх спільна довжина не перевищить 2*n*, що є малоймовірним, якщо стеки ростуть незалежно один від одного.

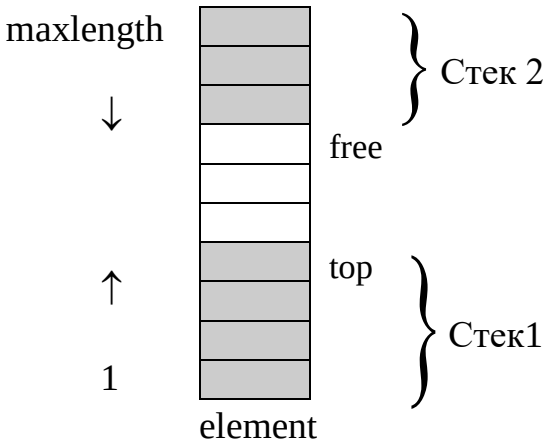


Рис.4. Представлення двох стеків один до одного

Абстрактні типи даних сімейства Stack (Стек) зазвичай використовують наступні п’ять операторів:

1. MAKENULL (S). Робить стек S порожнім.
2. TOP (S). Повертає елемент з вершини стека S.
3. POP (S). Видаляє елемент з вершини стека (виштовхує зі стеку).
4. PUSH(*x*,S) Включає елемент *x* у вершину стека S (заштовхує елемент в стек). Елемент, що раніше знаходився в вершині стека, стає елементом наступним за вершиною.
5. EMPTY(S). Ця функція повертає значення *true* (істина), якщо стек S порожній і значення *false* в протилежному випадку.

Для реалізації стеків за допомогою масивів абстрактний тип Stack можна визначити наступним чином:

```
Type
    Stack=Record
        top: integer;
        element: array [1..maxlength] ofelementtype
    end;
```

Наведемо реалізацію стеку представленням МАСИВ ВНИЗ. В цій реалізації стек складається з послідовності елементів element[*free*+1], element[*free*+2], …, element[*maxlength*]. Розглянемо п’ять типових операторів: MAKENULL, EMPTY, TOP, POP та PUSH.

```
procedure MAKENULL (var S: Stack);
begin
        S.free:= maxlength
end;
```

```
procedure POP (var S : Stack);
begin
    ifEMPTY (S) then
    error (`Стекпорожній`)
    else  S.free:= S.free +1
end;
```

```
function TOP (var S: Stack): elementtype;
begin
        ifEMPTY (S) then
                error (`Стекпорожній`)
        elseTOP := S.element[S.free +1]
end;
```

```
function EMPTY (S: Stack): boolean;
begin
         ifS.free>maxlengththen
                EMPTY := true
         else
                EMPTY := false
end;
```

```
procedure PUSH (x : elementtype; var S : Stack);
begin
        ifS.free = 0 then
        error (`Стекповний`)
        else
beginS.element[S.free] := x;
S.free:=S.free -1;
end
end;
```

Реалізація стеку за допомогою покажчиків.

У цій реалізації стек складається з вузлів, кожен з яких містить елемент стеку і покажчик на наступний вузол. Вузол, що містить останній елемент, має покажчик *nil* (нуль). Вузол *header*(заголовок) не містить елементів стеку, а тільки показує на перший елемент списку. Якщо стек порожній, заголовок містить покажчик *nil*. При цьому потрібна додаткова пам'ять для зберігання покажчиків. На рис. 5наведено представлення стеку за допомогою зв'язного списку описаного виду.

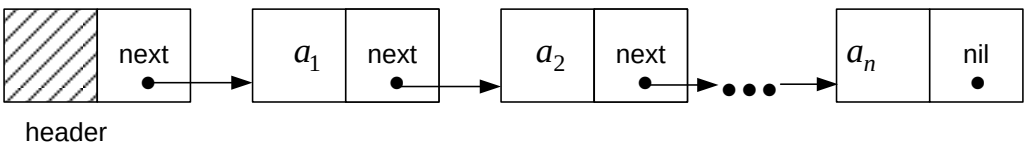


Рис.5. Представлення стеку за допомогою зв'язного списку

Таким чином, описати стек, який містить цілі числа, можна наступним чином:

```
Type
        PStack = ^TStack;
        TStack = Record
```

```
        Data : Integer;
                Next :PStack;
        end;
Var     Stack :PStack;
```

Елементидодаються у вершину (голову) стеку;елементивидаляютьсязвершини (голови) стеку;покажчик в останньомуелементідорівнюєnil.

Неможливо вилучити елемент і з середини стеку, не вилучивши всі елементи, що йдуть попереду.

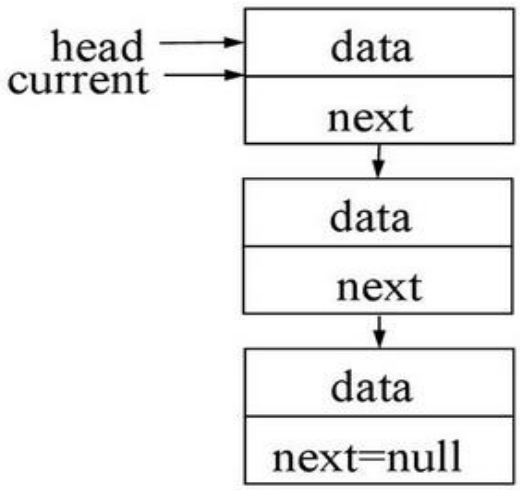
Для роботи зі стеком достатньо мати покажчик head на його вершину та допоміжний покажчик curren tна елемент стеку.

Якщо стек порожній, то значення змінної Stack дорівнює nil.

Включення елемента в стек відбувається аналогічно до вставки нового елемента в початок списку. При виключенні елемента зі стеку деякій змінній N присвоюється значення першого елементу стека і змінюється значення покажчика на початок стеку.

Алгоритм вставки елементадо стеку

1. Виділити пам'ять для нового елемента стеку: new(current);
2. Ввести дані до нового елемента: readln(current^.data);
3. Зв'язати допоміжний елемент із вершиною: current^.next:=head;
4. Встановити вершину стеку на новостворений елемент: head: =current;



Значенням покажчика headна вершину порожнього стеку є nil. Тому для створення стеку слід виконати оператор head:=nilта повторити щойно наведений алгоритм потрібну кількість разів.

Алгоритм видалення елемента зі стеку

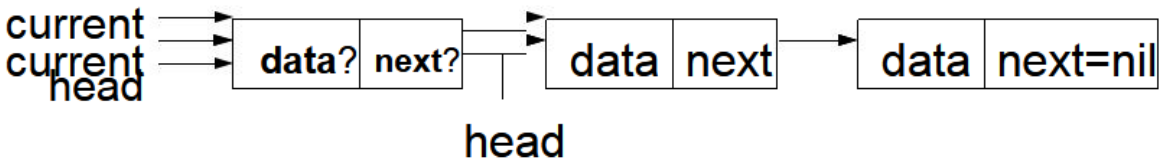
1. Створити копію покажчика на вершину

стеку: current :=head;

2. Перемістити покажчик на вершину стеку

на наступний елемент:head :=current^.next;

3. Звільнити пам'ять із-підколишньої вершини стеку: Dispose (current);



Включення елемента в стек	Виключення елемента зі стеку
Var x : PStack; …… New(x); x^.Data := …….; x^.Next := Stack; Stack := x;	**Var** N : Integer; x : PStack; …… N := Stack^.Data; x:= Stack; Stack := Stack^.Next; Dispose(x);

Завдання.

Написати програму: а) реалізації стеків за допомогою списків;

б) реалізації стеків за допомогою масивів, використовуючи МАСИВ ВГОРУ. Виконати: додавання елемента в стек; видалення елемента зі стеку; перевірка, чи порожній стек; перегляд елемента у вершині стека без видалення; очищення стека.

Контрольні запитання.

1. Що таке структура даних? Що таке стек?
2. Чим відрізняються статичні та динамічні структури даних?
3. У чому полягає схожість та відмінність структур даних список та стек?
4. Чи можна дістатися середини або кінця стеку, обійшовши його початок (вершину)?
5. Наведіть приклади з життя де зустрічається «принцип стеку».
6. У чому відмінність реалізацій стеку МАСИВ ВНИЗ та МАСИВ ВГОРУ?
7. У чому переваги представлення двох стеків один до одного?