ВКАЗІВНИКИ (ПОКАЖЧИКИ)

Базові поняття

Покажчики в мови C/C++ в використовуються набагато інтенсивніше, аніж в інших мовах, тому що іноді деякі обчислення виразити можливо лише за їх допомогою, а частково й тому, що з ними утворюються більш компактні та ефективніші програми, аніж програми з використанням звичайних засобів.

Існує твердження – аби стати знавцем C/C++ , потрібно бути спеціалістом з використання покажчиків.

Базові поняття

Змінні у програмах повинні розміщатися в одному з трьох місць:

в області даних програми	в області стеку	в області вільної пам’яті (купи)

Кожній змінній у програмі може відводитися пам’ять або **статично** (в момент завантаження), або **динамічно** (у процесі виконання програми).

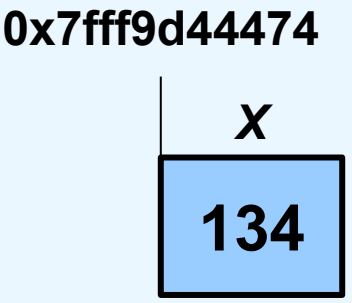
Базові поняття

Поняття змінної можна визначити як ім'я елементу пам'яті, в якому зберігається значення вказаного типу.

Кожен елемент пам'яті має свою унікальну адресу.

На рисунку показана умовна схема, на якому змінна ***X***, що має значення ***134***, а умовна адреса елементу пам'яті, де зберігається ця змінна, дорівнює ***0x7fff9d44474***.

Умовне позначення змінної X в оперативній пам'яті



Базові поняття

В мові C/C ++ розрізняють три види покажчиків, які відрізняються властивостями і набором допустимих операцій.

- покажчики на об'єкт певного типу даних
- покажчики на функцію
- безтіповий покажчик

Покажчик не є самостійним типом даних, він завжди пов'язаний з будь-яким іншим конкретним типом даних.

З поняттям покажчик тісно пов'язана операція взяття **адреси змінної**.

Базові поняття

У мові C/C++ є операція знаходження адреси позначається символом **&** і зазначає, за допомогою якої визначаються адреса комірки пам'яті, що містить задану змінну.

Наприклад, якщо *vr* — ім'я змінної, то *&vr* — адреса цієї змінної.

Для виведення адреси змінної функцією ***printf*** використовується модифікатор ***% p***.

Наприклад, для знаходження адреси використовується операція &, а для виведення на екран адреси змінної функція ***printf*** с модифікатором ***%d***:

```c
#include <stdio.h>
#include <locale.h>

int main() {
    setlocale(LC_ALL,"Russian");
    int X=134;
    printf("Значення змінної X - %d\n", X);
    printf("Адреса змінної X - %p\n", &X);
}
```

Після компіляції і виконання цієї програми на екран буде виведено:

```
Значення змінної X - 134
Адреса змінної X - 0x7fff9d444a74
```

Базові поняття

У C/C++ також існують і змінні типу **вказівник або покажчик**.

Покажчик - це змінна, значеннями якої є адреси в оперативній пам'яті.

Вказівник — це похідний тип даних, який використовується для зберігання адрес змінних і об'єктів.

Для зберігання вказівників (покажчиків) в програмі необхідно **оголосити змінну** (**змінна-вказівник**).

Для оголошення необхідно вказати тип значення, на яке вказує покажчик (наприклад, ***int***, ***float***, і т. п.) та поставити зірочку (*) перед ім'ям змінної.

Базові поняття

Покажчик **містить адресу змінної**, в якій **знаходиться** певне **значення**.

Змінна безпосередньо посилається на своє значення, покажчик посилається на значення певної змінної посередньо (непрямо).

Посилання на значення за допомогою покажчика називається **непрямим адресуванням**.

Оголошення та ініціалізація змінної-покажчика

Покажчик на об'єкт містить адресу області пам'яті, в якій зберігаються дані певного типу (основного або складеного).

Значенням змінної-вказівника є адреса змінної або об'єкта.

Найпростіше оголошення покажчика на об'єкт (в подальшому просто вказівник або покажчик) має вигляд:

<тип> *<ім'я вказівника на змінну заданого типу>;

Зірочка відноситься безпосередньо до імені.

Оголошення та ініціалізація змінної-покажчика

Покажчики, як і інші змінні, повинні бути оголошені до моменту їхнього використання.

int *countPtr, count;

де **countPtr** типу **int *** - покажчик на значення цілого типу;

змінна **count** цілого типу.

Можна оглошувати покажчики, що посилаються на змінні будь-якого типа.

Оголошення та ініціалізація змінної-покажчика

Для того, щоб оголосити кілька покажчиків, потрібно ставити її перед ім'ям кожного з них.

Наприклад, в операторі ***int * a, b, * c;*** описуються два покажчика на ціле з іменами **a** та **c,** а також ціла змінна **b**.

Покажчик може бути **константою** або **змінною**:
int * pi; // покажчик на цілу змінну

const int * pci; // покажчик на цілу константу

Оголошення та ініціалізація змінної-покажчика

Якщо заздалегідь невідомо, яку адресу зберігає вказівник, то йому присвоюється значення 0, але вказівник не може бути неініціалізований.

Вказівник приймає значення 0, якщо :

- ✓ вичерпана пам'ять, то оператор new повертає 0
- ✓ це вказівник на кінець динамічної структури (список, дерево)

Вказівник, значення якого рівне 0, називається **порожнім**.

```
int* p=0; //порожній вказівник
```

Оголошення та ініціалізація змінної-покажчика

Вказівник/покажчик *– це змінна для адреси елементу пам'яті. Загальний формат оголошення вказівника має вигляд:*

<тип> *<имя>.

Наприклад, коди (директиви) ***int *p; float *r;*** виділяють дві області пам'яті ***p*** і ***r*** розміром два байти.

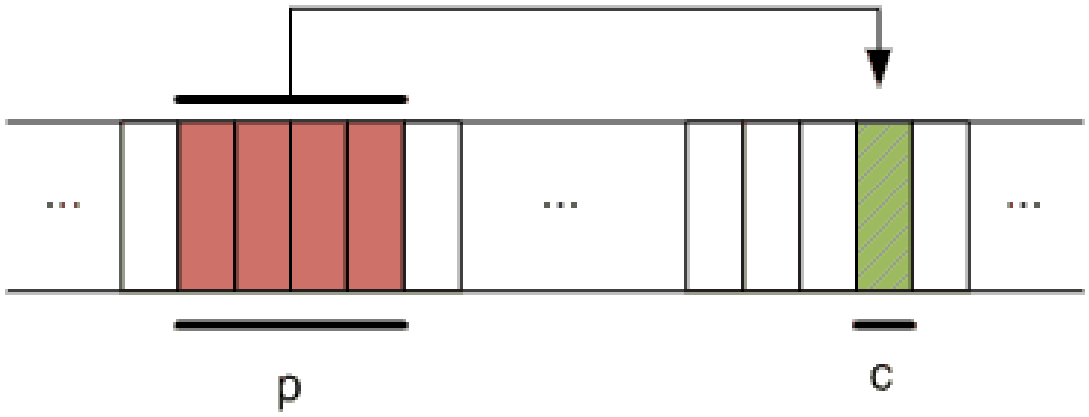
При цьому змінна ***p*** призначена для запису адрес змінних типу ***int***, а змінна ***r*** призначена для адрес змінних типу ***float***.

Оголошення та ініціалізація змінної-покажчика

- Покажчики повинні бути ініційовані або в момент оголошення, або за допомогою операції присвоєння.
- Покажчик може бути ініційований нулем, макросом **NULL**, значенням адреси.
- Покажчик із значенням **NULL** не посилається ні на що.
- Ініціація покажчика числом 0 еквівалентна ініціації покажчика константою **NULL.**
- Для уникнення непередбачених результатів покажчики повинні завжди бути ініційовані до моменту їхнього використання.

Вказівники

```
char c;   // змінна
char *p; // покажчик
p = &c;  // p = адреса c
```



Вказівники

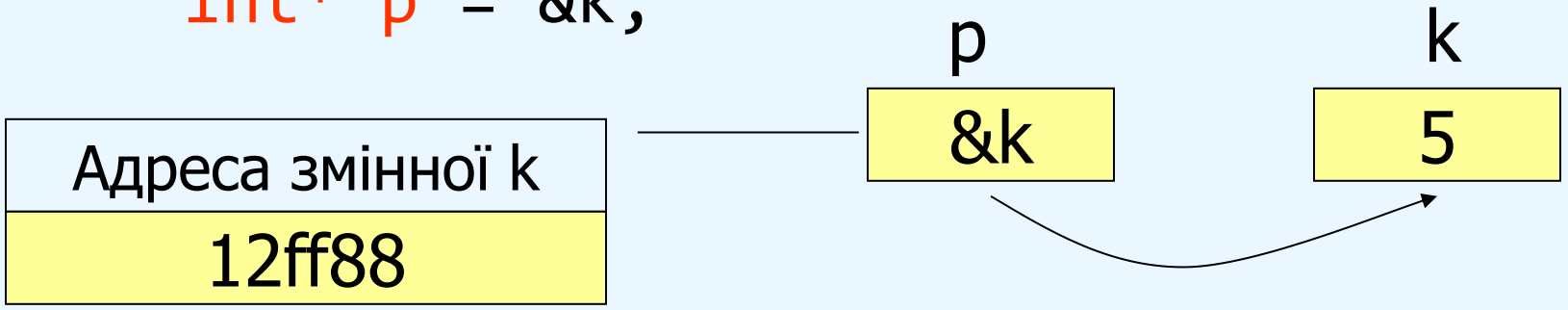
Вказівник - це похідний тип даних, що містить адресу змінної певного типу.

Для типу T тип T* – вказівник на T

Змінна T* містить адресу об’єкту T

Приклад

```
int k = 5;
int* p = &k;
```

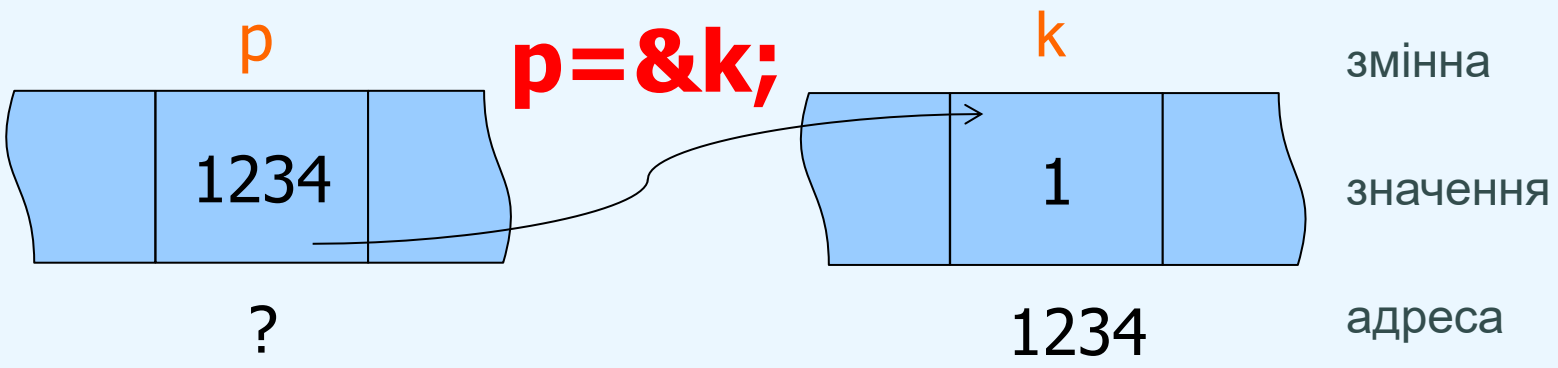


Вказівники і адреси

Вказівник містить адресу об’єкта і забезпечує «непрямий» доступ до цього об’єкта.

Приклад

```
int k = 1;
int* p;
```



Приклади опису вказівників


```
int *ptri; //вказівник на змінну цілого типу
char *ptrc; //вказівник на змінну символьного типу
float *ptrf;//вказівник на змінну з плаваючою крапкою
```

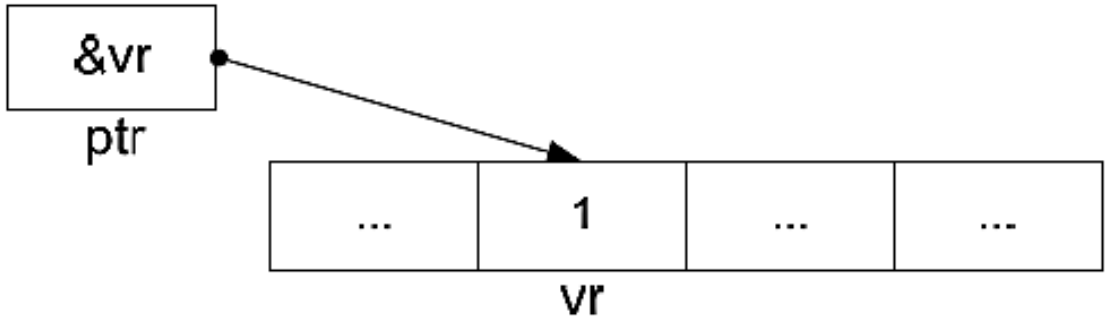
- ✓ Змінні різних типів займають різну кількість комірок пам'яті.
- ✓ При цьому для деяких операцій з вказівниками необхідно знати об'єм відведеної пам'яті.
- ✓ Однак самі змінні типу вказівник мають однаковий розмір.

Приклади опису вказівників

Нехай змінна-вказівник має ім'я ptr (тобто оголошена як int* ptr), тоді в якості значення їй можна присвоїти адресу за допомогою наступного оператора:

ptr=&vr;

```
int vr = 1;
int* ptr = &vr; // ptr містить адресу змінної vr
```



Вказівники

Програма 1.

Оголошує вказівник. Видає випадкову інформацію з області пам'яті, яка виділяється під вказівник.

```
int main()
{
    int *p; //Визначається змінна p, яка є вказівником.
    printf ("В змінній p міститься випадкова адреса: %p \n ", &p);

    return 0;
}
```

Після виконання цієї програми на екрані видачі результатів з'явиться наступне повідомлення:

```
В змінній p міститься випадкова адреса: 0x7ffe5ee815c0
```

Вказівники

При оголошенні вказівника, наприклад ***int *p***, виділяється пам'ять розміром два байти, в яких і знаходиться випадкове число А.

Це число комп'ютер інтерпретує як адреса пам'яті для запису цілого числа.

Число з адреси А, яка записана в вказівнику, доступно для комп'ютера.

Отже, при оголошенні вказівника користувачеві доступні дві області пам'яті.

Вказівники

Перша область відводиться під вказівник. Ця область пам'яті має ім'я ***p***.

Друга область визначається **адресою**, яка зберігається в вказівнику.

Ім’я другої області пам'яті – ****p***. Причому адреса змінної ****p*** рівний ***A***, тобто справедлива тотожність ***&(*p)?A***.

Зауваження 1. Використовувати вказівник, який не був ініціалізізований (не задавався) програмою, не можна!!!

Вказівники

Доступ до інформації з адрес, які зберігаються в вказівниках типу ***int *p***, ***float *r***, тощо, реалізується, наприклад, директивами:

```
int a=6;
int *p;
p=&a;
```

**p = 6* – в елемент пам'яті, адреса якої зберігається в вказівнику p, заслати число 6;

```
printf ("%d \n ", *p);
```

– видає на екран інформацію з елементу пам'яті, адреса якої зберігається в вказівнику p.

Приклади


```
int i = 5, j;  //оголошення змінних

int* p = &i;   //p містить адресу змінної i

*p = 5;  //присвоює змінній i значення 5

j  =  p;  //заборонено (вказівник не може бути
          перетворений в цілочисельний тип int!)

j = *p+1; //j=6

p = &j; //p вказує на j
```

Приклади

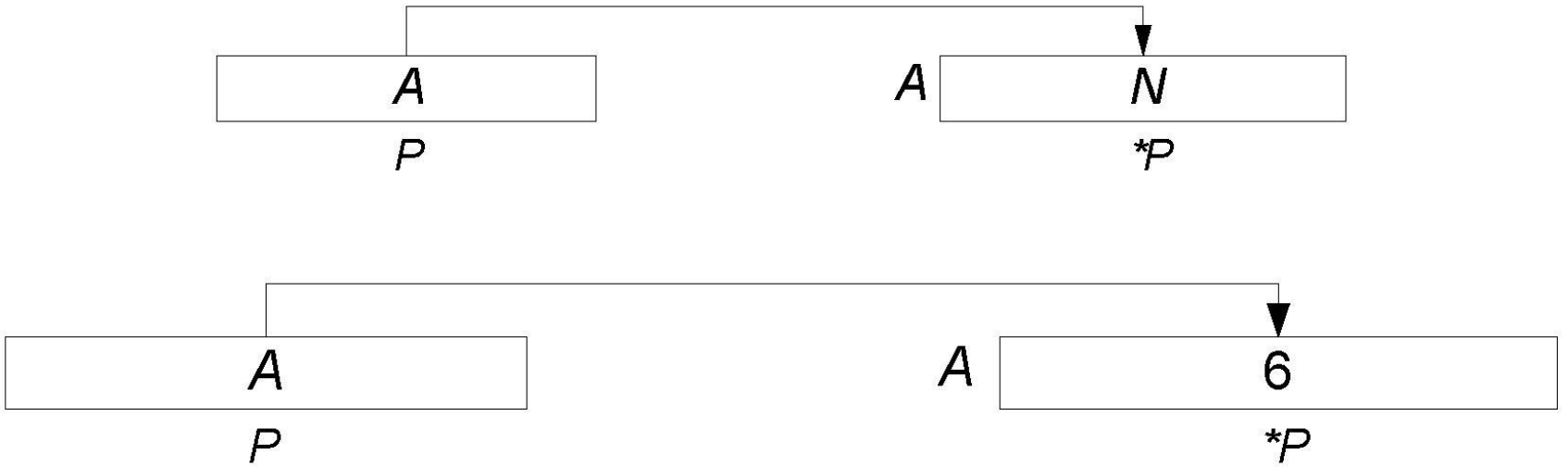

```c
#include <stdio.h>

int main() {
    char c = 'A';
    int i = 7776;
    int *pi = &i;
    char *pc = &c;

    printf("pi=%p, *pi=%d, &pi=%p\n", (void*)pi, *pi, &pi);
    printf("pc=%p, *pc=%c, &pc=%p\n", (void*)pc, *pc, &pc);

    return 0;
}
```

input

```
pi=0x7ffc91038074, *pi=7776, &pi=0x7ffc91038078
pc=0x7ffc91038073, *pc=A, &pc=0x7ffc91038080


...Program finished with exit code 0
Press ENTER to exit console.
```

Вказівники

Схема пам'яті до і після виконання коду ***p = 6***:



Для видачі вказівника на екран або для введення його з клавіатури використовується формат ***%p***.

Вказівники

Програма 2.

Оголошує покажчик.

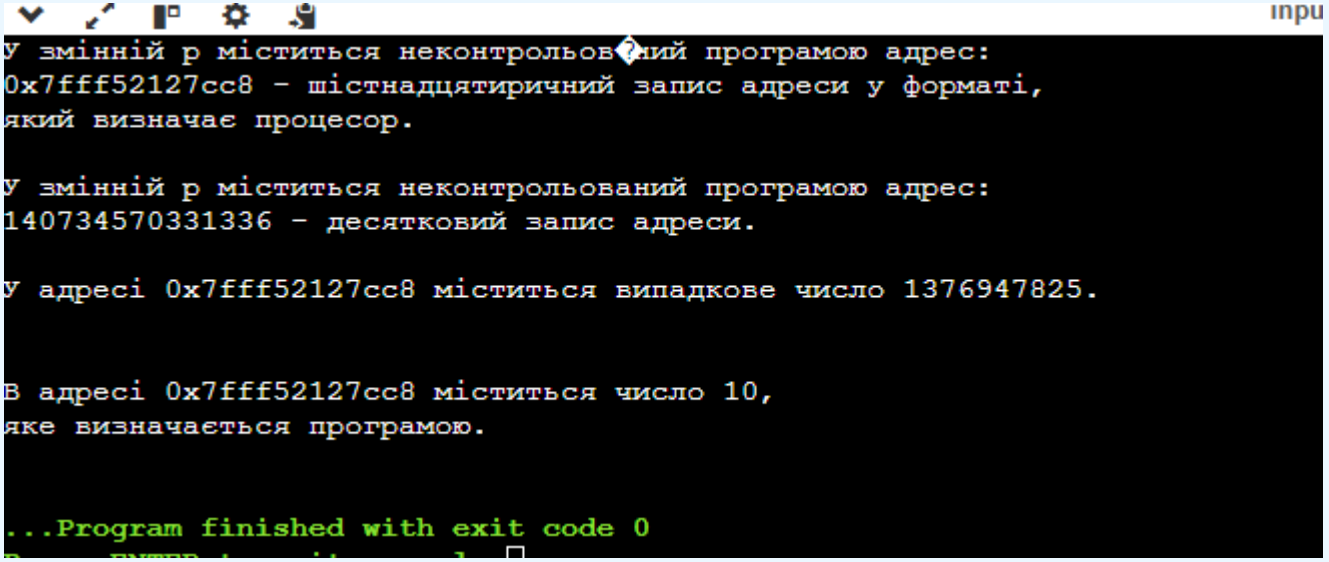
Видає інформацію з областей пам'яті ***p*** і ****p***, які доступні програмі.

```c
#include <stdio.h>

int main() {
    int *p;

    printf("У змінній p міститься неконтрольований програмою адрес:\n"
           "%p - шістнадцятиричний запис адреси у форматі,\n"
           "який визначає процесор.\n\n",
           p);

    printf("У змінній p міститься неконтрольований програмою адрес:\n"
           "%ld - десятковий запис адреси.\n\n",
           (long)p);

    printf("У адресі %p міститься випадкове число %d.\n",
           p, *p);

    *p = 10;

    printf("\n\nВ адресі %p міститься число %d,\n"
           "яке визначається програмою.\n",
           p, *p);

    return 0;
}
```

Вказівники

Програма 2.

Після закінчення роботи даної програми на екрані монітора з'явиться вміст змінних ***p***, ****p*** до ініціалізації ****p*** і після її ініціалізації, тобто після виконання коди ****p = 10***.



Способи ініціалізації вказівників/покажчиків

Вказівник можна ініціалізувати, використовуючи директиву ***new***.

Наприклад, директива ***p = new int***, де ***p*** – вказівник, виділяє пам'ять для запису цілого числа.

Адреса цієї області пам'яті заноситься в вказівник ***p*** і дана пам'ять ставиться під контроль програми.

Способи ініціалізації вказівників/покажчиків

У вказівник можна вводити інформацію з клавіатури, використовуючи директиву введення формату ***scanf()*** із специфікацією %p.

Програма 3. Оголошує вказівник. Вказівнику привласнюється адреса однотипної змінної.

```c
#include <stdio.h>

int main() {
    int *p;
    int x = 222;

    printf("До ініціалізації вказівника!!!!:\n");
    printf("В адресі %p міститься число *p= %d\n\n", (void *)p, *p);

    p = &x;

    printf("\n\nПісля ініціалізації p = &x!!!!:\n");
    printf("Адреса x = %p\n", (void *)&x);
    printf("У вказівнику p адрес = %p, у якому число = %d\n", (void *)p, *p);

    return 0;
}
```

```
До ініціалізації вказівника!!!!:
В адресі 0x7ffdbf8885e0 міститься число *p= -1478324064


Після ініціалізації p = &x!!!!:
Адреса x = 0x7ffdbf8884ec
У вказівнику p адрес = 0x7ffdbf8884ec, у якому число = 222


...Program finished with exit code 0
Press ENTER to exit console.
```

Способи ініціалізації вказівників/покажчиків

Програма 4.

Оголошує вказівник. Для ініціалізації вказівника використовується директива ***new***.

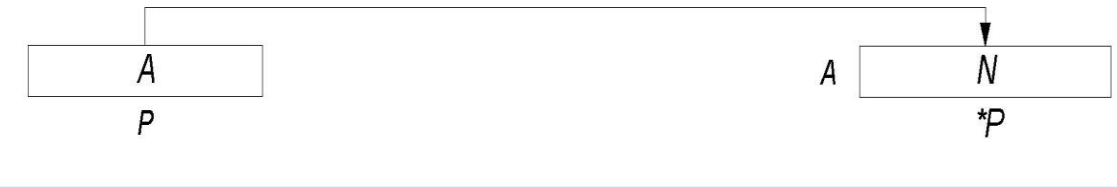
```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *p;

    printf("Стан пам'яті до ініціалізації!!!!:\n\n");
    printf("В неконтрольованій адресі %p міститься випадкове число %d\n", (void *)p, *p);

    *p = -8;
    printf("\nВ неконтрольованій адресі %p міститься визначене програмою число %d\n", (void *)p, *p);
    printf("Натисніть будь-яку клавішу...\n");

    getchar();

    p = (int *)malloc(sizeof(int));
    if (p == NULL) {
        printf("Помилка виділення пам'яті!\n");
        return 1;
    }

    printf("\nВ контрольованій адресі %p міститься випадкове число %d\n", (void *)p, *p);
    printf("Натисніть будь-яку клавішу...\n");

    getchar();

    *p = -777;
    printf("\nВ контрольованій адресі %p міститься визна

    free(p);

    return 0;
}
```

```
Стан пам'яті до ініціалізації!!!!:

в неконтрольованій адресі 0x7ffe71d63008 міститься випадкове число 1909870192

В неконтрольованій адресі 0x7ffe71d63008 міститься визначене програмою число -8
Натисніть буд яку клавішу...


В контрольовані адресі 0x5b9fab173ac0 міститься випадкове число 0
Натисніть будь-яку клавішу...


В контрольовані адресі 0x5b9fab173ac0 міститься визначене програмою числ -777


...Program finished with exit code 0
Press ENTER to exit console.
```

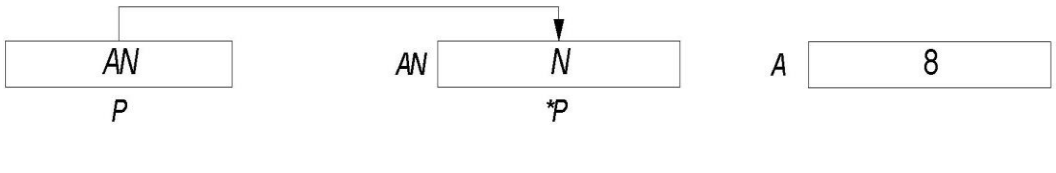
Способи ініціалізації вказівників/покажчиків

```
Стан пам'яті до ініціалізації!!!!:

►неконтрольованій адресі 0x7ffe71d63008 міститься випадкове число 1909870192

В неконтрольованій адресі 0x7ffe71d63008 міститься визначене програмою число -8
Натисніть буд►яку клавішу...


В контрольовані►адресі 0x5b9fab173ac0 міститься випадкове число 0
Натисніть будь-яку клавішу...


В контрольовані►адресі 0x5b9fab173ac0 міститься визначене програмою числ►-777


...Program finished with exit code 0
Press ENTER to exit console.
```

Розберемо роботу програми. Схема пам'яті після оголошення змінних.



Після виконання коду ***p = new int*** комп'ютер виділяє пам'ять з адресою AN і адреса даної пам'яті заноситься в вказівник ***p***.



Способи ініціалізації вказівників/покажчиків

```c
#include <stdio.h>

int main() {
    int *p;
    int x = 2;

    printf("Стан пам'яті до ініціалізації!!!!:\n\n");
    printf("В неконтрольованій адресі %p міститься випадкове число %d\n\n", (void *)p, *p);

    printf("Адреса змінної x у форматі, який визначений процесором = %p\n", (void *)&x);

    printf("\n\nВведіть адресу змінної x у форматі (який визначений процесором, див. попередній вивід): ");
    scanf("%p", (void **)&p);

    printf("\n\nСтан пам'яті після ініціалізації вказівника:\n\n");
    printf("У змінній p адреса %p. У цій адресі число %d\n", (void *)p, *p);

    return 0;
}
```

```
input
Стан пам'яті до ініціалізації!!!!:

неконтрольованій адресі 0x7fff24f0c3d0 міститься випадкове число -1750957888

Адреса змінної x у форматі, який виз чений процесором = 0x7fff24f0c2dc

Введіть адресу змінн x у форматі (який визначений процесором, див. попер ній вивід): 0x33f

Стан пам'яті пі ля ініціалізації вказівника:


...Program finished with exit code 139
Press ENTER to exit console.
```

Програма 5. Оголошує вказівник. Адреса в вказівник вводиться з клавіатури директивою ***scanf,*** в якій використовується специфікація ***%p***.

Способи ініціалізації вказівників/покажчиків

Дана програма спочатку видасть адресу змінної x у форматі процесора.

Потім в цьому ж форматі з клавіатури в вказівник p вводиться адреса змінної x.

Далі після введення в p адреси x виводимо вміст змінної p (там має бути адреса x) і *p (тобто вміст x = 2).

Способи ініціалізації вказівників/покажчиків

```c
#include <stdio.h>

int main(void) {
    int **p;

    printf("Три області пам'яті, які доступні програмі:\n\n");

    printf("У p міститься неконтрольована програмою адреса:\n"
           "%p - шістнадцятиричний запис адреси у форматі,\n"
           "який визначає процесор.\n\n", p);

    printf("У *p міститься неконтрольована програмою адреса:\n"
           "%p - шістнадцятиричний запис адреси у форматі,\n"
           "який визначає процесор.\n\n", *p);

    printf("У **p міститься неконтрольоване програмою число: %d\n\n", **p);

    printf("\nАдреси даних областей пам'яті:\n\n");
    printf("Адреса p = %p\n", (void *)&p);
    printf("Адреса *p = %p\n", (void *)&(*p));
    printf("Адреса **p = %p\n", (void *)&(**p));

    return 0;
}
```

Програма 6.
Оголошує вказівник на вказівник. Видає інформацію з областей пам'яті, які доступні програмі.

```
Три області пам'яті, які доступні п►ограмі:

У p міститься неконтрольована програмою адреса:
0x7fff135b9110 - шістнадцятиричний запис адреси у форматі,
який визначає процесор.

У *p міститься неконт►ольована програмою адреса:
0x5f2ca5ff10a0 - шістнадцятири►ий запис адреси у форматі,
який визначає процесор.

У **p міститься неконтрольоване програмою число: -98693133


Адреси даних областей пам'яті:

Адреса p = 0x7fff135b9020
Адреса *p = 0x7fff135b9110
Адреса **p = 0x5f2ca5ff10a0


...Program finished with exit code 0
Press ENTER to exit console.
```

Способи ініціалізації вказівників/покажчиків

Відмітимо, що якщо змінна ***p*** визначається, наприклад, кодом ***int **p***, то при виконанні коду ***p = new *int*** виділяється пам'ять під вказівник і адресу цієї області пам'яті заноситься в ***p***.

На закінчення перерахуємо всі операції над вказівниками, які допустимі в мові C/C++:

- привласнення для однотипних вказівників;
- збільшення (віднімання) значення вираження цілого типа;
- унарні операції ++, --;
- порівняння для однотипних вказівників;
- різниця між однотипними вказівниками. Результатом різниці є число цілого типа.

Способи ініціалізації вказівників/покажчиків

У мові C/C++ ім'я масиву є вказівником на перший елемент.

Це означає, що якщо, наприклад, визначений масив int ***x[n]***, то в змінній x міститься адреса першого елементу масиву.

Значення змінної x в цьому випадку можна привласнювати вказівнику того ж типа.

Способи ініціалізації вказівників/покажчиків

Програма 7. Демонструє факт, що ім'я масиву є вказівником на перший елемент.

```c
#include <stdio.h>

int main(void) {
    const int n = 10;
    int a[n], i;

    printf("В змінній a знаходиться адреса %p\n", (void *)a);
    printf("Адреса 1-го елементу масиву %p\n", (void *)&a[0]);

    return 0;
}
```

```
В змінній a знаходиться адреса 0x7fffaa879bf0
Адреса 1-го елементу масиву 0x7fffaa879bf0


...Program finished with exit code 0
Press ENTER to exit console.
```

Після виконання цієї програми на екрані видачі результатів з'явиться наступне повідомлення:

У змінній a знаходиться адреса 0xN

Адреса 1-го елементу масиву 0xN

Дякую за увагу!