

Вказівники

Вказівник (або покажчик) – особливий тип даних, значенням якого є адреса певного байта оперативної пам'яті. Значення покажчика - це беззнакове ціле. Воно повідомляє, де розміщена змінна, і нічого не говорить про саму змінну.

Вказівники можуть бути:

- константними, зберігають незмінну адресу ділянки ОП (такими є імена масивів і символічних рядків);
- змінними, їм можна надавати значення адрес різних ділянок ОП.

Змінна типу *вказівник* оголошується подібно звичайним змінним із застосуванням унарного символу “ * “ .

Форма оголошення наступна:

*тип * ім'я вказівника ;*

де *тип* - найменування типу змінної, адресу якої буде містити змінна-вказівник (на яку він буде вказувати).

Вказівник містить адресу першого байту змінної визначеного типу. Тип змінної, що адресується, і на яку посилається вказівник, визначає об'єм ОП, що виділяється змінній, та зв'язаному з нею вказівнику.

Приклад:

*char *pc;*

int ml [5];

*float *pf;*

/ - ім'я масиву на 5 значень типу int;*

ml - покажчик-константа/*

Константа **NULL** зі стандартного файлу *stdio.h* призначена для ініціалізації вказівників нульовим (незайнятим) значенням адреси.

Базові операції:

Унарні операції **&** та ***** :

& ім'я змінної - одержання адреси. Визначає адресу розміщення значення змінної визначеного типу;

*** ім'я-вказівника** - отримання значення визначеного типу за вказаною адресою. Визначає вміст змінної, розміщеної за адресою, що міститься у даному покажчику. Це – непряма адресація (інші назви - "зняття значення за вказівником" або "розіменування").

Оператор присвоювання значення адреси вказівнику :

Ім'я_змінної_вказівник = & ім'я змінної;

Наприклад:

*int i, *pi; /* pi -змінна вказівник*/*

pi = &i; / pi одержує значення адреси 'i' */*

Одне з основних співвідношень при роботі з вказівниками – це симетричність операцій адресації та непрямой адресації. Вона полягає в тому, що:

****&x == x,***

тобто вміст за адресою змінної **x** є значення **x**.

Над вказівниками можна виконувати всі операції порівняння. *Приклад:*

```
int *pc, *pnew;
```

```
if ( pc==pnew ) pc=NULL;
```

До значення вказівника можна додавати (чи віднімати) довільне ціле число: $p=p + k$. Результатом буде збільшена на $k**sizeof*$ (базовий тип вказівника p) нова адреса. Величина, яка додається до значення вказівника (чи віднімається) є кратною розміру об'єкта.

Нетипізовані вказівники

Оголошуються `void *p;`

Сумісний з усіма вказівниками програми та адресами усіх об'єктів. Можна присвоювати значення *void* –вказівників вказівникам з довільним типом і навпаки.

Застосовують для оголошення параметрів функцій, що можуть набувати значення адрес об'єктів різних типів, а також щоб забезпечити можливість повернення з функції адреси неозначеного об'єкта.

Бібліотечна функція *malloc ()*, яка виділяє в динамічній пам'яті ділянки заданого обсягу, має тип результату *void ** (повертає адресу першого байта виділеної ділянки), що дає змогу використовувати її для розташування в пам'яті об'єктів різних типів

Приклад

```
char *pst ;
```

```
double *par;
```

```
pst = malloc ( 300 );
```

```
par = malloc ( n * sizeof(double) );
```

Вказівник *pst* отримає адресу першого байта ділянки, а вказівник *par* – адресу першого байта першого елемента динамічного масиву з *n* даних, що мають тип *double*. Можна виконувати присвоєння

**pst = 'x';*

**(par + 3) = 14.7;*

Першим символом рядка, на який вказує *pst*, стане символ 'x', а константу *14.7* буде записано як четвертий елемент масиву дійсних чисел.

Хоча значення нетипізованого вказівника можна присвоювати довільному іншому вказівникові, рекомендують застосовувати операцію явного перетворення типу (явне перетворення обов'язкове для мови C++)

```
pst = (char *)malloc ( 300 );
```

```
par = (double *)malloc ( n * sizeof (double) );
```

Використовуючи операцію перетворення типу, можна присвоїти вказівнику значення іншого вказівника, що має відмінний базовий тип (без додаткового нетипізованого вказівника)

```
pst = (char *) par;
```

Обережно з присвоєнням вказівнику адреси об'єкта іншого типу !