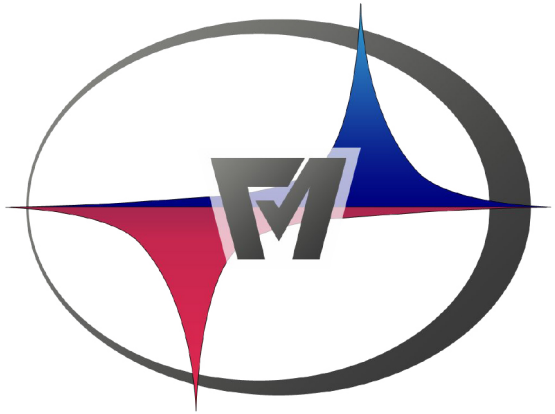

Вказівники

к.т.н., доц., доцент кафедри прикладної математики
Рижа Ірина Андріївна

Про що ця лекція???

- Опишемо поняття вказівника, його синтаксис та потребу у застосуванні.
- Розглянемо зв'язок між масивами та вказівниками.

Вказівники

У відповідних частинах оперативної пам'яті зберігаються:

- вхідні дані;
- результати роботи програми;
- проміжні дані тощо.

Усі дані (крім абсолютних констант) мають адресу розташування в оперативній пам'яті комп'ютера.

Можливість доступитися до даного за його адресою з використанням спеціального типу даних:

+ Паскаль і похідні;
+ C і похідні;

– Basic;
– Visual Basic;
– Java

Потреба в застосуванні вказівників

Для чого потрібні вказівники?

1. доступ до елементів масиву, бо такий спосіб є швидшим від операції індексації;
2. передавання аргументів у функцію, в якій ці аргументи зазнають змін;
3. передавання у функцію масивів і стрічкових змінних;
4. виділення пам'яті під час виконання програми;
5. створення складних динамічних структур (списки, дерева, графи).

Адреса змінної. Оператор &

Операція встановлення адреси & (амперсанд)

– унарна операція для визначення адреси певного даного в ОП. Числове зображення адреси змінної у пам'яті комп'ютера:

- подається у 16-ій системі числення;
- у середовищі MSVS є 32-ох розрядна;
- адреса кожного байту ОП – вісім 16-их цифр.

Якщо оголошено змінні чи поіменовані константи, тоді операцією & можна встановити відповідні адреси та вивести їх на екран.

Зверніть увагу!

Пам'ять не виділяється під:

- абсолютну константу: `cout<<&2;`
- окремий вираз: `cout<<&(a+b);`

Приклади

Приклад 1.

Розташування глобальних даних в ОП.

Приклад 2.

Розташування локальних даних в ОП.

Синтаксис оголошення вказівника

Вказівник

– це комірка пам'яті, що призначена для зберігання адреси даного;

- **змінна-вказівник** (або просто вказівник) – зберігає адресу змінної;
- **вказівник на константу** – зберігає адресу поіменованої константи;
- **константа вказівник** – вказівник, адресний вміст у якому змінюватися не може;

Якого типу сам вказівник???

- **Не існує** окремого зарезервованого слова, що встановлює приналежність даного до **типу вказівник**.
- Вказівник містить **адресу даного конкретного типу**.

Синтаксис оголошення вказівника

тип_даного * ім'я_змінної_вказівника;

- тип-даного – будь-який зі стандартних типів даних мови, а також тип користувача, який задекларований раніше;
- у MSVS під змінну-вказівник комп'ютер виділяє **4 байти** (32 розряди);
- **ВАЖЛИВО**, щоби при оголошенні вказівника символ * був наявним **обов'язково!**

```
int *ptr_int_1,ptr_int_2;
int *ptr1; int * ptr2;int* ptr3; int*ptr4, * ptr5;
```

Приклади

Приклад 3.

Оголошення вказівників та присвоєння їм адреси даних.

Синтаксис оголошення вказівника

Вказівники повинні мати значення!

У випадку використання **неініціалізованого вказівника** (“сміттєвої” адреси) відповідна адреса може виявитись адресою будь-чого: **від коду програми до коду операційної системи**.

Надання вказівнику значення:

- ініціалізація адресою вже існуючого даного під час оголошення;
- наступне присвоєння конкретної адреси даного після оголошення;
- надання адресній комірці значення `NULL` чи просто `0`;
- присвоєння значення адреси, яка зберігається в комірці, що також є вказівником на дане такого ж самого типу.

Звертання до змінної за її адресою

Чи можна за відомою адресою існуючої змінної встановити вміст інформації за цією адресою?

Непряме звертання

– використання адреси змінної замість її ідентифікатора для звертання до самої змінної, до її значення, яке міститься у відповідній комірці.

Операція розадресації

– операція, яка дозволяє втілювати непряме звертання.

Операція розадресації (*)

*** вказівник**

- є **унарною** адресною операцією;
- виконується над даним, яке є **вказівником**;
- дозволяє використати **дане**, яке є за адресою, записаною в комірці, що є вказівником;
- **може бути:**
  - у **правій** частині оператора присвоєння – використовується значення змінної за її адресою;
  - у **лівій** частині оператора присвоєння – змінюється вмістиме комірки, адреса якої записана у відповідному вказівнику;
  - окремим **операндом** у виразі;
  - **фактичним параметром** функції;
- дозволяє **використовувати і змінювати** значення змінної, прямий доступ до якої неможливий.

Приклади

Приклад 4.

Оголошення вказівників та присвоєння їм адреси даних.

Оператор присвоєння для вказівників

Синтаксис:

- ім'я_змінної_вказівник_1 = & ім'я_змінної;
- ім'я_змінної_вказівник_2 = ім'я_змінної_вказівник_1;
  де обидва вказівники на змінні **однакового** типу.

Зверніть увагу!

Неявного приведення типу для вказівників не передбачено!

Приклади

Приклад 5.

Використання оператору присвоєння для вказівників.

Приклад 6.

Особливості явного приведення типів для вказівників.

Вказівник на void

Вказівник на void

– тип вказівника, який може вказувати на **дане будь-якого типу**

void * ім'я_вказівника;

- Призначені для передавання параметрів-вказівників у функції, що працюють незалежно від типу даних, на які вказує вказівник.
- Вказівникові на void **можна** присвоювати як значення конкретного вказівника, так і адресу змінної довільного типу.
- Вказівникові на конкретний тип присвоювати значення адреси, яка була записана у вказівник на void **НЕ можна**.
- **НЕ можна** звертатися за відповідним вмістом через операцію розадресації *.

Приклади

Приклад 7.

Використання вказівників на void.

Зв'язок між масивами та вказівниками

- Ім'я статичного масиву є коміркою, яка містить адресу області оперативної пам'яті, що виділена для зберігання однотипних даних, об'єднаних у масив.
- Доступ до елементів кожного масиву здійснюється за операцією індексації **[ ]**, тобто розадресації:

  ім'я_масиву[індекс_елемента масиву].

- **Вказівнику** на відповідний тип, можна присвоїти **адресу області, де зберігається масив** даних такого типу.
- Використовуючи операцію індекації до змінної-вказівник, можна мати доступ до кожного елемента масиву:

  ***(ім'я_вказівника + лічильник).**

Приклади

Приклад 8.

Зв'язок між масивом даних певного типу та вказівником на цей тип.

- Записи
  ar_int[i], *(ar_int + i), ptr_ar_1[i], *(ptr_ar_1 + i), ptr_ar_2[i] та *(ptr_ar_2 + i) **є майже еквівалентними**;
- виведення елементів масиву можна подати так:

```
for(int i = 0; i < n; i++)
cout<<*(ar_int +i)<< "\t"<<*(ptr_ar_1 + i) << "\t"<<*(ptr_ar_2+ i)<< "\n";
```

- Перший елемент будь-якого одновимірного масиву масиву може бути взятий так *ім'я_масиву.

Дякую за увагу!

Далі буде...