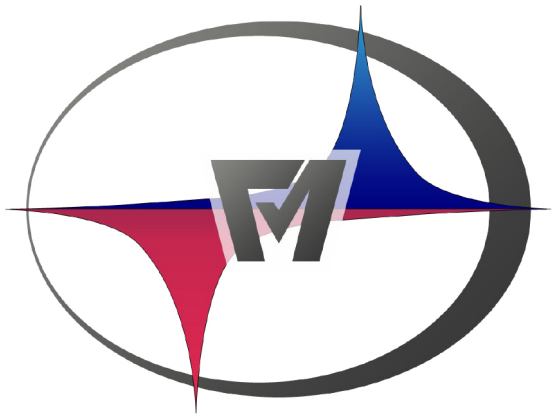

Вказівники та посилання

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

Про що ця лекція???

- Розглянемо операції над вказівниками.
- Подамо поняття константного вказівника і вказівника на константу.
- Опишемо поняття динамічної пам'яті та засоби мови для роботи з нею.

Операції над вказівниками

Унарні операції над вказівниками

- Операція **інкременту** (++) збільшує вказівник на одиницю обсягу пам'яті, яку займає дане, що на нього вказує вказівник;
- Операція **декременту** (- -) зменшує вказівник на одиницю обсягу пам'яті, яку займає відповідне дане;
- **Префіксний** варіант – спершу змінюється значення у відповідній комірці, а тоді використовується сама комірка;
- **Постфіксний** варіант – спершу використовується значення з комірки, а тоді змінюється вміст комірки.
- Унарні операції мають однаковий пріоритет, але виконуються справа наліво.

Операції над вказівниками

Приклад 1.

Використання операції інкременту для доступу до елементів масиву.

Приклад 2.

Особливості використання префіксного та постфіксного варіантів інкременту для доступу та зміни елементів масиву.

Зауваження

Операції інкременту та декременту можна виконувати й над вказівниками, що містять адреси простих змінних, але адресне зміщення не завжди дасть нам значення наступної змінної, оскільки в MSVS між локальними змінними є 8 байтів пам'яті.

Операції над вказівниками

Бінарні операції над вказівниками

- **адитивні операції** – додавання чи віднімання цілого числа (ptr±i) – зміна адреси на величину, яка рівна кількості байт пам'яті, що їх займає вказане число (i) даних того типу, на який вказує вказівник;
- **встановлення різниці** двох вказівників (віднімання вказівників: ptr1–ptr2) – результатом є число елементів даного типу, які можуть бути записані в області між вказаними двома адресами.

Приклад 3.

Використання бінарних операцій для роботи з вказівниками.

Операції над вказівниками

Логічні операції

- **Операції порівняння** – значення адреси можуть співпадати чи не співпадати.
- **Логічні операції** (!, &&, ||) – Значення вказівника, що не дорівнює NULL, дає істинне значення виразу, а для порожньої адреси — хибне.

Приклади роботи зі вказівниками

Приклад 4.

Використовуючи масив – статичний масив вказівників, впорядкувати статичну прямокутну матрицю за неспаданням елементів її першого стовпця.

Вказівники на константу

Вказівник на константу

```
const тип_даного * ім'я_вказівника;
```

- вважається змінною-вказівником на константу певного типу;
- можна одразу присвоїти адресу оголошеної раніше константи або зробити це пізніше звичайним присвоєнням;
- дозволено виконувати усі операції, як і над вказівниками на змінні;
- вмістиме комірки за цією адресою змінювати **НЕ можна**.

Константні вказівники

Константний вказівник на змінну

```
тип_даного ім'я_змінної;
тип_даного * const ім'я_вказівника = &ім'я_змінної;
```

- при оголошенні вказівникові одразу слід надати адресу попередньо оголошеної змінної;
- змінювати адресу, яка міститься у вказівнику, на адресу іншої змінної **НЕ можна**;
- константний вказівник **НЕ можна** ініціалізувати адресою константи відповідного типу;
- вмістиме комірки за цією адресою можна змінювати за допомогою операції розадресації.

Константні вказівники

Константний вказівник на константу

```
const тип_даного ім'я_константи = значення;
тип_даного * const ім'я_вказівника = &ім'я_константи;
```

- Над константним вказівником на константу **НЕ можна виконувати жодної операції**, яка передбачала би зміну як адреси, так і значення самої константи, вказівником на яку ми оперуємо.

Поняття динамічної пам'яті

Статична пам'ять

- відбувається виділення пам'яті при оголошенні змінної (числової, символьної, адресної);
- відповідна змінна займає обсяг ОП доти, поки це визначено класом пам'яті змінної.

Динамічна пам'ять

- пам'ять під змінну виділяється **за потребою під час виконання програми**;
- змінна "живе" в ОП комп'ютера доки пам'ять не звільнять програмно або до кінця роботи програми;
- область ОП, яка виділяється, називають **"купою"** ("heap");
- початкове значення такої змінної є непередбачуване, тому вимагає чіткої ініціалізації.

Засоби мови для роботи з динамічною пам'яттю

При роботі з масивами

1. необхідно ще до запуску на виконання знати, **який обсяг пам'яті слід зарезервувати під масив**;
2. необхідно оголосити розмірність масиву за допомогою абсолютної чи поіменованої константи;
   - якщо розмір масиву є **меншим** за зарезервовану пам'ять, то частина пам'яті не використовується;
   - якщо зарезервований обсяг пам'яті є **недостатнім**, то потрібно вносити зміни у код, перекомпільовувати його;
3. потрібно мати інструмент, для виділення бажаного розміру ОП у процесі виконання програми.

Оператор new

Операція виділення динамічної пам'яті

передбачає виділення в області динамічної пам'яті, достатньої для зберігання даного відповідного розміру (типу).

- адреса першого байта виділеної області пам'яті записується у вказівник на відповідний тип;
- доступ до даного здійснюється не через ім'я комірки, а через її адресу, яка зберігається у вказівнику.

Оператор new

Синтаксис

```
тип_даного * ім'я_вказівника;
ім'я_вказівника = new тип_даного;
```

або

```
тип_даного * ім'я_вказівника = new тип_даного;
```

- Назви типу даного при оголошенні вказівника і при виконанні операції new є **однаковими**!
- Доступ до вмісту цієї області здійснюється **виключно** операцією розадресації, бо ця область імені не має.

Приклад 5.

Доступ до динамічно виділеної пам'яті.

Оператор new

Динамічне виділення пам'яті під масив

тип_даного * ім'я_вказівника;
ім'я_вказівника = new тип_даного[розмір масиву];

- Розмір масиву – додатне ціле значення (змінна, поіменована чи абсолютна константа).
- В області heap буде виділено суцільну область, розміром:

розмір_масиву × sizeof (тип_даного).

Приклад 6.

Динамічне виділення пам'яті під масив.

Оператор new

Важливо!

- **НЕ** можна виділити область розміром, кратним розміру типу даного, наприклад:

```
ptr_dbl_ar=new double*n;
ptr_dbl_ar=new n*double;
```

- Якщо випадково у комірку з адресою **динамічного масиву** внести якусь іншу адресу, то виділена частина динамічної пам'яті стане **НЕДОСТУПНОЮ** як програмі, так і системі аж до моменту завершення програми.

Отже,

наявність такого потужного інструменту дозволяє раціонально використовувати пам'ять під час виконання програми.

Оператор delete

Операція звільнення динамічної пам'яті

передбачає звільнення області динамічної пам'яті для майбутнього використання з під даних, які перестали бути потрібними.

delete ім'я_вказівника; або **delete [ ] ім'я_вказівника;**

Важливо!

- Якщо при звільненні пам'яті, виділеної під масив, не вказати [ ], то в область heap буде повернено лише одну комірку (1-ий елемент масиву), а решта стануть **НЕДОСТУПНИМИ** до кінця виконання програми.

Втрата пам'яті

– недоступність даних у динамічній пам'яті через втрату адреси цієї області, а також при незвільненні області з-під зайвої інформації.

Приклади

Приклад 7.

Обчислити середнє значення вибірки з трицифрових чисел вказаної розмірності, яка генерується комп'ютером.

Поняття посилання у C++

Посилання (reference)

– це інше ім'я (псевдо) вже існуючої, оголошеної змінної чи вказівника:

- використовується для передавання аргументів у функцію;
- має ту ж саму адресу, що й змінна, на яку воно оголошене;
- оголошується **НЕ** на тип, а на дане конкретного типу;
- кожне посилання оголошується тільки для конкретної змінної;

Якщо над посиланням на змінну виконається деяка операція, то ця операція виконається над самою змінною.

Поняття посилання у C++

Синтаксис оголошення посилання

тип_даного & ім'я_посилання = ім'я_існуючої_змінної;

- **Посилання повинно бути ініціалізованим при оголошенні.**
- Посилання **НЕ** можна оголошувати для абсолютної константи.
- Посилання можна оголошувати на вказівник і використовувати це друге, альтернативне ім'я для доступу до змінної чи динамічного виділення пам'яті.

Приклад 8.

Використання посилання на змінну.

Приклад 9.

Використання посилання на вказівник.

Поняття посилання у C++

Синтаксис оголошення const посилання

```
const тип_даного ім'я_константи = значення;
const тип_даного & ім'я_посилання = ім'я_константи;
```

- оголошується для константи;
- дозволяє передати у функцію параметр, який є константою за означенням;
- поіменована константа, якщо це не формальний параметр функції, вже має бути оголошеною.

Дякую за увагу!

Далі буде...