Презентація

на тему:

“Динамічна пам’ять.”

підготував студент II курсу
групи КС-151
Кучеренко Андрій

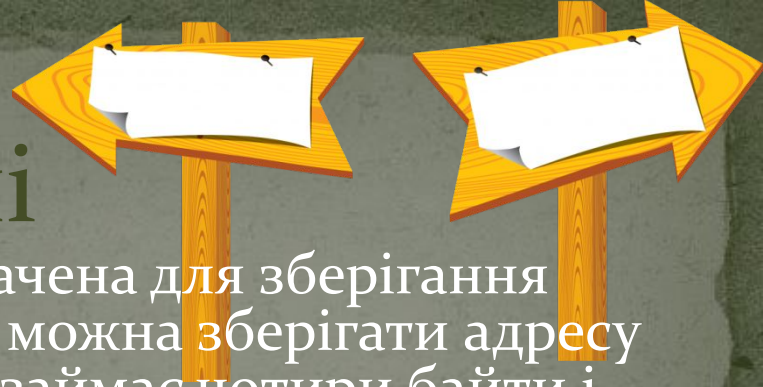
Зміст

- Загальні поняття.
- Вказателі.
- Стандартні функції.
- Динамічні змінні.
- Вказателі на процедури і функції.
- Дерева.

Загальні поняття

У IBM PC-сумісних комп'ютерах пам'ять умовно розділена на сегменти. Компілятор формує сегменти коду, даних і стека, а решта доступна програмі пам'ять називається динамічної (хіпом).

Динамічні змінні створюються в хіпі під час виконання програми. Звернення до них здійснюється через покажчики. За допомогою ЦИХ змінних можна Обробляти дані, обсяг яких на початок виконання програми не відомий. Пам'ять під такі дані виділяється блоками, які зв'язуються один з одним. Цей спосіб зберігання даних називається динамічними структурами.



Вказателі

Вказателем називається змінна, призначена для зберігання адрес областей пам'яті. У покажчику можна зберігати адресу даних або програмного коду. Адреса займає чотири байти і зберігається у вигляді двох слів, одне з яких визначає сегмент, друге - зміщення.

Вказателі діляться на стандартні і визначаються програмістом. Величини стандартного тіпа pointer предназначени для зберігання адрес даних довільного типу:

var p: pointer;

Програміст може визначити покажчик на дані або підпрограму конкретного типу. Як і для інших нестандартних типів, це робиться в розділі type:

type pword = ^ word; {Читається як "покажчик на word"}

...

var pw: pword;

Такі вказателі називаються типізований. Можна описати покажчик на будь-який тип даних, крім файлових.

Тип покажчика на дані можна описати і безпосередньо при описі змінної:

var pw: ^ word;

Для покажчиків визначені тільки операції перевірки на рівність і нерівність і присвоювання. Правила присвоєння покажчиків:

Будь-якому вказівником можна привласнити стандартну константу nil, яка означає, що покажчик не посилається на якусь конкретну осередок пам'яті.

Покажчики стандартного типу pointer сумісні з покажчиками будь-якого типу.

Вказівником на конкретний тип даних можна привласнити тільки значення покажчика того ж або стандартного типу.

Операція @ і функція addr дозволяють отримати адресу змінної:

var x: word; {Змінна}

pw: ^ word; {Покажчик на величини типу word}

...

pw: = @w; {Або pw: = addr (w); }

Для звернення до значення змінної, адреса якої зберігається в покажчику, приміряється операція разадресації (разименованія), що позначається за допомогою символу ^:

pw ^: = 2; inc (pw ^); writeln (pw ^);

З величинами, адреса яких зберігається в покажчику, можна виконувати будь-які дії, допустимі для значень цього типу.

Стандартні функції

- *addr (x): pointer - повертає адресу x (аналогічно операції @), де x - ім'я змінної або підпрограми;*
- *seg (x): word - повертає адресу сегменту для x;*
- *ofs (x): word - повертає зсув для x;*
- *cseg: word - повертає значення регістра сегмента коду CS;*
- *dseg: word - повертає значення регістра сегмента даних DS;*
- *ptr (seg, ofs: word): pointer - по заданому сегменту і зміщення формує адресу типу pointer.*

Динамічні змінні

Динамічні змінні створюються в хіпі під час виконання програми за допомогою підпрограм new або getmem. Динамічні змінні не мають власних імен - до них звертаються через покажчики.

Процедура new (var p: тіп_указателя) виділяє в динамічної пам'яті ділянку розміру, достатнього для розміщення змінної того типу, на який посилається вказівник p, і адреса початку цієї ділянки заносить в цей покажчик.

Функція new (тіп_указателя): pointer виділяє в динамічної пам'яті ділянку розміру, достатнього для розміщення змінної базового типу для заданого типу покажчика, і повертає адресу початку цієї ділянки.

new застосовується для типізованих покажчиків

Процедура getmem (var p: pointer; size: word) виділяє в динамічної пам'яті ділянку розміром в size байт і привласнює адресу його початку покажчику p. Якщо виділити необхідний обсяг пам'яті не вдалося, програма аварійно завершується. Покажчик може бути будь-якого типу.

Вказателі на процедури і функції

Покажчик на підпрограму визначається як змінна процедурного (функціонального) типу, наприклад:

```
type
fun = function (x: real): real;
    {Функціональний тип}
var
pf: fun; {Покажчик на функції типу fun}
```

Вказівником на підпрограму можна привласнити nil, значення іншого покажчика того ж типу або ім'я конкретної підпрограми.

```
type
fun = function (x: real): real;
    {Функціональний тип}
var pf: fun; {Покажчик на функції типу
    fun}
function f (x: real): real; far; {Конкретна
    функція}
begin
тіло функції
end;
...
pf: = f;
```

Після виконання цього фрагмента програми в змінної pf буде зберігатися адреса точки входу в функцію f. Тепер функцію f можна викликати через змінну pf:

```
y: = pf (x);
```

Функція, адреса якої присвоюється змінної, повинна компілюватися в режимі дальнього адресації. Для цього в заголовку вказується директива far або до опису функції задається ключ компіляції {$ F +}.

Дерева

Дерево - це сукупність елементів, які називаються вузлами (при цьому один з них визначено як корінь), і відносин (батьківський-дочірній), що утворюють ієрархічну структуру вузлів. Вузли можуть бути величинами будь-якого простого або структурованого типу, за винятком файлового. Вузли, які не мають жодного подальшого вузла, називаються листям.

У двійковому (бінарному) дереві кожен вузол може бути пов'язаний не більше ніж двома іншими вузлами. Рекурсивно двоичное дерево визначається так: двійкове дерево буває або порожнім (не містить жодного вузла), або містить вузол, званий коренем, а також два незалежних поддерева - ліве піддерево і праве піддерево.

Двійкове дерево пошуку може бути або порожнім, або воно має таку властивість, що кореневий елемент має більше значення вузла, ніж будь-який елемент в лівому поддереве, і менше або рівне, ніж елементи в правому піддереві. Зазначене властивість називається характеристичним властивістю двійкового дерева пошуку та виконується для будь-якого вузла такого дерева, включаючи корінь. Далі будемо розглядати тільки двійкові дерева пошуку. Таку назву двійкові дерева пошуку отримали з тієї причини, що швидкість пошуку в них приблизно така ж, що і в відсортованих масивах: O (n) = C • log2n (в гіршому випадку O (n) = n).

Дякую за увагу!