**Класи пам'яті і область дії**

Класи пам'яті мови Сі дають можливість визначити, з якими функціями пов'язані якісь змінні, і як довго змінна зберігається в програмі. Ми вже згадували, що локальні змінні відомі тільки функцій, що містить їх. У мові Сі передбачається також, що про глобальних змінних знають відразу кілька функцій:

/\* глобальна змінна ext \*/

int ext; /\* зовнушня змінна \*/

main( )

{

extern int ext;

printf("скільки курсів на сайті intuit.ru?\n");

scanf("%d",&ext);

while(ext != 30)

critic( );

printf("Перегляньте на сайті!\n");

}

critic( )

{

extern int ext;

printf("Помилка. Спробуйте знову.\n");

scanf("%d",&ext);

}

Результат:

скільки курсів на сайті intuit.ru

20

Помилка. Спробуйте знову.

30

Перегляньте на сайті!

Ми зробили змінну Ext зовнішньою, описавши її поза будь-якого визначення функції. Усередині функції, що використовує цю змінну, ми оголошуємо її зовнішньою за допомогою ключового слова extern, що передує специфікації типу змінної. Компілятор шукає визначення цієї змінної поза функцією. Якби ми опустили ключове слово extern в функції critic( ), то компілятор створив би в функції criticнову змінну і теж назвав би її Ext. Тоді інша змінна Ext, яка знаходиться в main( ), ніколи не отримала б нового значення.

Кожна змінна має тип і належить до певного класу пам'яті. Час життя і область дії ідентифікатора визначаються асоційованим з ним класом пам'яті. Існують чотири різновиди класів пам'яті:

auto - **автоматичний** - локальні ідентифікатори, пам'ять для яких виділяється при вході в блок, тобто складовийоператор, і звільняється при виході з блоку. Слово auto є скороченням слова automatic.

static - ***статистичний*** - локальні ідентифікатори, існуючі в процесі всіх виконань блоку. На відміну від ідентифікаторів типу auto, для ідентифікаторів типу static пам'ять виділяється тільки один раз - на початку виконання програми, і вони існують, поки програма виконується.

extern - ***зовнішній*** - ідентифікатори, які називаються зовнішніми, external, використовуються для зв'язку між функціями, в тому числі незалежно скомпільованими функціями, які можуть перебувати в різних файлах. Пам'ять, асоційована з цими ідентифікаторами, є постійною, однак її вміст може змінюватися. Ці ідентифікатори описуються поза функцією.

register - **регістровий**- ідентифікатори, подібні до ідентифікаторів типу auto. Їх значення, якщо це можливо, повинні поміщатися в регістрах машини для забезпечення швидкого доступу до даних.

Якщо клас пам'яті ідентифікатора не вказано явно, то його клас пам'яті задається положенням його визначення в тексті програми. Якщо ідентифікатор визначається всередині функції, тоді його клас пам'яті auto, в інших випадках ідентифікатор має клас пам'яті extern.

Припустимо, що є програма на мові Сі, вихідний текст якої міститься в декількох файлах. Для розділення даних (для зв'язку) у функціях в цих файлах використовуються ідентифікатори, визначені як extern. Якщо функція посилається на зовнішній ідентифікатор, то файл, який містить його, повинен мати опис або визначення цього ідентифікатора. Явне завдання класу пам'яті extern вказує на те, що цей ідентифікатор визначений в іншому файлі, і тут йому пам'ять не виділяється, а його опис дано лише для перевірки типу і для генерації коду.

|  |  |
| --- | --- |
| ! | При описі зовнішніх масивів максимальне значення першого індексу масиву вказувати немає необхідності. Воно буде отримано з відповідної ухвали, що створює додаткові зручності для користувача. |

Для зовнішнього ідентифікатора пам'ять виділяється тільки в тому випадку, якщо клас пам'яті не вказано явно.

|  |  |
| --- | --- |
| ! | Явна вказівка пам'яті extern є відмітною ознакою зовнішнього опису від зовнішнього визначення. |

Хоча опис зовнішнього ідентифікатора може зустрітися в багатьох файлах, тільки один файл повинен містити визначення зовнішнього ідентифікатора. Область дії зовнішніх ідентифікаторів не обмежується файлом, що містить їх визначення, а включає також файли з відповідними описами, з класом пам'яті extern.

Визначення класу пам'яті змінної залежить від того, де змінна описана і яке ключове слово, якщо воно є, використовується.  
Клас пам'яті дозволяє встановити два факти. По-перше, визначити, які функції мають доступ до змінної. Межі, до яких змінна доступна, характеризують її область дії. По-друге, визначити, як довго змінна знаходиться в пам'яті. Тепер докладніше розглянемо властивості кожного типу.

**Автоматичні змінні**

За замовчуванням змінні, описані усередині функції, є **автоматичними**. Можна, однак, це підкреслити явно за допомогою ключового слова auto:

main( )

{

auto int kat;

}

Так надходять, якщо хочуть, наприклад, показати, що визначення змінної не потрібно шукати поза функцією.  
Автоматичні змінні мають локальну область дії. Тільки функція, в якій змінна визначена, знає її. Інші функції можуть використовувати змінні з тим же самим ім'ям, але це будуть незалежні змінні, що знаходяться в різних осередках пам'яті.

*Автоматична змінна* починає існувати при виконанні функції, що містить її. Коли функція завершує свою роботу і повертає керування туди, звідки її викликали, автоматична змінна зникає. Область дії автоматичної змінної обмежена блоком, тобто {}, В якому змінна описана.

|  |  |
| --- | --- |
| ! | Ми завжди повинні описувати наші змінні на початку тіла функції (блоку). Областю дії їх є вся функція. Можна описати змінну всередині подблока. Тоді змінна буде відома тільки в цій частині функції, однак цього краще не робити. Це поганий стиль програмування! |

### Зовнешні змінні

*Змінна, описана поза функцією, є зовнішньою*.

Глобальні змінні визначаються на тому ж рівні, що і функції, тобто вони не локальні ні в якому блоці. Постійні глобальні змінні не започатковано нулем, якщо явно незадано інше початкове значення. Областю дії є вся програма. Вони повинні бути описані у всіх файлах програми, в яких до них є звернення. Деякі компілятори вимагають, щоб глобальні змінні були визначені тільки в одному файлі, і описані як зовнішні в інших файлах, де вони використовуються. Глобальні змінні повинні бути описані у файлі до першого використання.

Приклад:

int global\_flag;

Зовнішню змінну можна описати і в функції, яка використовує її, за допомогою ключового слова extern. Групу extern - описів можна зовсім опустити, якщо вихідні визначення змінних з'являються в тому ж файлі і перед функцією, яка їх використовує. **Включення ключового слова** extern **дозволяє функції використовувати зовнішню змінну, навіть якщо вона визначається пізніше в цьому або іншому файлі.** Обидва файли повинні бути скомпільовані, пов'язані або зібрані в один і той же час.

Якщо слово extern не включене в опис всередині функції, то під цим ім'ям створюється нова автоматична змінна. Ми можемо помітити другу змінну як автоматичну за допомогою слова auto.

### Статичні змінні

**Статичні** тут означає, що змінні залишаються в роботі. Вони мають таку ж область дії, як автоматичні змінні, але вони не зникають, коли містить їх функція закінчить свою роботу. Компілятор зберігає їх значення від одного виклику функції до іншого.

Приклад:

/\* Статична змінна \*/

main( )

{

int count;

for(count = 1;count <= 3; count ++)

{

printf("Підрахунок студентів %d:\n", count);

man\_woman ( );

}

}

man\_woman( )

{

int man = 1;

static int woman = 1;

printf("юнаків = %d та дівчат = %d\n",

man++, woman++);

}

*Функція* man\_woman збільшує кожну змінну після друку її значення. Робота цієї програми дає наступні результати:

Підрахунок студентів 1:

юнаків = 1 и дівчат = 1

Підрахунок студентів 2:

юнаків = 1 и дівчат = 2

Підрахунок студентів 3:

юнаків = 1 и дівчат = 3

*Статична змінна* woman пам'ятає, що її значення було збільшено на 1, в той час як для змінної man початкове значення встановлюється кожного разу заново. Це вказує на різницю в ініціалізації: man инициализируется кожен раз, коли викликається man\_woman ( ), в той час як woman инициализируется тільки один раз при компіляції функції man\_woman ( ).

**Зовнішні статичні змінні**

Можна описати **статичні** змінні поза будь-якої функції. Це створює **зовнішню статичну змінну**. Різниця між зовнішньою змінною і зовнішньої статичною змінною полягає в області їх дії. Звичайна зовнішня змінна може використовуватися функціями в будь-якому файлі, а зовнішня статична змінна може використовуватися тільки функціями того ж самого файлу, причому після визначення змінної. Статичну змінну ми описуємо поза будь-якої функції.

**реєстрові змінні**

Зазвичай змінні зберігаються в пам'яті машини. *Реєстрові змінні* запам'ятовуються в регістрах центрального процесора, де доступ до них і робота з ними виконуються набагато швидше, ніж в пам'яті. В іншому реєстрові змінні аналогічні *автоматичним змінним*.

Приклад:

main( )

{

register int pleat;

}

Компілятор порівнює наші вимоги до кількості доступних регістрів, тому ми можемо і не отримати те, що хочемо. У цьому випадку змінна стає простий автоматичної змінної.

**Особливості роботі з мовою СІ**. Який клас пам'яті застосовувати? Відповідь на питання - автоматичний. Цей клас пам'яті обраний за замовчуванням. Використання зовнішніх змінних дуже спокусливо. Якщо описати всі змінні як зовнішні, то не буде турбот при використанні аргументів і покажчиків для зв'язку між функціями в прямому і зворотному напрямках. Але тоді виникає проблема з функцією С, змінює змінні в функції А, а ми цього не хотіли! Така проблема значно переважує гадану привабливість широкого використання зовнішніх змінних. Одне із золотих правил програмування полягає в дотриманні принципу "необхідно знати тільки те, що потрібно". Організуйте роботу кожної функції автономно, наскільки це можливо, і використовуйте глобальні змінні тільки тоді, коли це дійсно необхідно!

Операція отримання адреси & непридатна до реєстрових змінних. Будь-які змінні в блоці**, крім формальних параметрів функції**, можуть бути визначені як статичні.

Підіб’ємо підсумки.

*Класи пам'яті, які описуються всередині функції*:

1. *автоматичний,* тривалість існування - тимчасово, область дії - локальна;
2. *регістровий*, тривалість існування - тимчасово, область дії - локальна;
3. *статичний,* тривалість існування - постійно, область дії - локальна.

*Класи пам'яті, які визначаються поза функцією*:

1. *зовнішній,* тривалість існування - постійно, область дії глобальна (всі файли);
2. *зовнішній статичний*, тривалість існування - постійно, область дії - глобальна (один файл).