Основні типи даних

Щоб реалізувати алгоритм, програмам необхідно працювати з даними - числами, символами, тобто об’єктами, які містять в собі інформацію, призначену для використання. Деякі дані встановлюються рівними певним значенням ще до того, як програма починає виконуватися, а після її запуску такі значення зберігаються незмінними на всьому протязі роботи програм. Ці дані називаються константами. Дані, які можуть змінюватись, або їм може бути присвоєне значення під час виконання програми, називаються змінними. Різниця між змінною та константою очевидна: під час виконання програми значення змінної може бути змінено (наприклад, за допомогою присвоєння), а значення константи змінити не можна.

Крім різниці між змінними та константами існує ще й різниця між типами даних. Деякі дані в програмі є числами, деякі – символами. Компілятор повинен вміти ідентифікувати та обробляти дані будь-якого типу. В мові Сі передбачено використання декількох основних типів даних. Якщо величина являється константою, то компілятор може розпізнати її тип тільки по тому виду, в якому вона існує в програмі. У випадку зі змінною, необхідно щоб її тип був оголошений в операторі опису. В стандарті мови СІ використовується сім ключових слів, які вказують на різні типи даних:

int

long

short

unsigned

char

float

double

Перші чотири ключових слова використовуються для представлення **цілих**, тобто цілих чисел без десятої чи дробової частини. Якщо ми хочемо підкреслити, що ціле не може бути від’ємним, потрібно до цілого дописувати ключове слово unsigned, наприклад, unsigned short. char призначене для вказівки на букви та інші символи. float, double використовується для представлення чисел з десятиною точкою. Типи, позначені цими ключовими словами, можна розділити на два класи по принципу розміщення в пам’яті машини. Перші п’ять ключових слів визначають цілі типи даних, останні два – типи даних з плаваючою точкою. Ось коротке пояснення їхнього сенсу. Терміни біт, байт, слово – використовуються для опису як елементів даних, які оброблює комп’ютер, так і для елементів пам’яті. Розглянемо ці поняття відносно пам’яті. Найменша одиниця пам’яті називається біт. Вона може приймати одне з двох значень: 0 або 1. Байт в більшості машин складається з 8 біт. Всього в байтовому форматі можна представити 256(два у восьмій степені) різноманітних комбінацій з нулів та одиниць. Ці комбінації можна використовувати для представлення цілих чисел в діапазоні від 0 до 255 або для кодування набору символів. Слово являється природним елементом пам’яті. Існують ЕОМ, у яких слово дорівнює 8 бітам, 16 бітам, 32 бітам або 64 бітам.

Опис різних типів, змінні та константи

**Цілі числа**. У цілого числа ніколи не буває дробової частини. Уявивши ціле число в двійковому вигляді, його не важко помістити в машині. Наприклад, число 3 в двійковому вигляді виглядає як 11. Якщо його помістити в слово 32-розрядної машини, необхідно перші 30 біт встановити в 0, а останні 2 біта – в 1.

**Числа с плаваючою крапкою**. Числа з плаваючою крапкою відповідають тому, що математики називають речовими числами. Спосіб кодування, що використовується для розміщення в пам’ять числа з плаваючою крапкою, повністю відрізняється від розміщення цілого числа. Числа з плаваючою крапкою представляють у вигляді дробової частини та порядку числа, а потім обидві частини поміщають в пам’ять.

Всі дані типів int, short, long є числами зі знаками, тобто значеннями цих типів можуть бути тільки цілі числа - позитивні, негативні і нуль. Один біт використовується для вказівки знака числа, тому максимальне число зі знаком, яке можна уявити в слові, менше, ніж максимальне число без знака.

**Опис даних цілого типу.** При описі даних необхідно написати тільки тип, за яким має слідувати список імен змінних. Наприклад, int dog, rad, nina. Як роздільник між іменами змінних необхідно використовувати кому.

**Цілі константи.** Згідно з правилами мови Сі, число без десяткового дробу і без показника ступеня розглядається як ціле. Тому компілятор по запису константи визначає, ціла вона або речова. Якщо потрібно ввести константу типу long, то потрібно вказати ознаку L або l в кінці числа. Якщо під час запису константи ціле починається з цифри 0, то ця константа інтерпретується як вісімкове число, якщо ж ціле починається з символу 0x або 0X - як шістнадцяткове число.

Ініціалізація змінних цілого типу. Константи застосовуються при ініціалізації змінних. Це означає присвоювання змінної деякого значення перед початком обробки. Можна ініціалізувати змінну в операторі опису.

Наприклад,

int dog=5;

int rad=077;

int nina =0X99;

|  |  |
| --- | --- |
| ! | У мові Сі введено три класи цілих чисел, що мають різні розміри. Тим самим користувачеві мови Сі надали можливість вибору типу змінної з вимогою завдання. Наприклад, якщо змінна типу int займає одне слово, а змінна типу long займає два слова, значить тип long дозволяє обробляти великі числа. Якщо у задачі не використовуються великі числа, то нема чого вводити в програму змінні типу long, тому що якщо замість числа, що займає одне слово в пам'яті, використовується число, що займає два слова, робота машини сповільнюється. |

**Опис даних типу unsigned.** Цей тип є модифікатором типів: int, short, long. Ми можемо використовувати комбінацію ключових слів unsigned int, unsigned short, unsigned long, тобто змінна не може приймати від'ємного значення. Для вказівки типу unsigned int досить написати unsigned. Цілі беззнакові константи записуються так само, як і звичайні константи, заборонено тільки використання знака мінус.  
Наприклад, unsigned age;

**Опис даних типу char**. Цей тип визначає цілі числа без знака в діапазоні від 0 до 255 (це залежить від реалізації, char може збігатися як з signed char так і c unsigned char). Таке ціле зазвичай розміщується в одному байті пам'яті. Для опису символьної змінної застосовується ключове слово char. Правила опису більш ніж однієї змінної і ініціалізації змінних залишаються тими ж, що і для інших основних типів.  
Наприклад, char dog, cat;

**Символьні константи**. Символи в мові СІ закривають в апострофи.

Напрклад, char dog; dog='b';

Якщо апострофи опущені, то компілятор вважає, що використовується неописана змінна b. У стандартній мові Сі значенням змінної або константи типу char можуть бути тільки поодинокі символи.

Приклади символьних констант: ' A ', ' a ', ' 7 ', ' $ '.

Спеціальні(керуючі) символьні константи:

|  |  |
| --- | --- |
| Новий рядок (перехід рядка) | '\n' |
| Горизонтальна табуляція | '\t' |
| Вертикальна табуляція | '\v' |
| Повернення на крок | '\b' |
| Повернення каретки | '\r' |
| Переклад формата | '\f' |
| Зворотна коса | '\\' |
| Апостроф | '\'' |
| Лапки | '\"' |
| Нульовий символ (пусто) | '\0' |

Крім того, будь-який символ може бути представлений послідовністю трьох вісімкових цифр: '\ ddd'. Символьні константи вважаються даними типу int.

**Строкові константи.** Строкова константа являється послідовністю символів коду ASCII, укладеної в лапки.

Наприклад:"Це строкова константа"

В кінці кожного рядка компілятор поміщає нульовий символ '\ 0', що відзначає кінець цього рядка. Кожна строкова константа, навіть якщо вона ідентична іншій строкової константи, зберігається в окремому місці пам'яті. Якщо необхідно ввести в рядок символ лапок ("), то перед ними треба поставити символ зворотної косої (\). У рядок можуть бути введені будь-які спеціальні символьні константи, перед якими стоїть символ \. Символ \ і наступний за ним символ нового рядка ігнорується.

**Константи що перераховуються.** Імена, зазначені в описі констант що перераховуються, трактуються як цілі числа (див. Опис перерахування).

**Опис даних типу float і double.** Числа з плаваючою крапкою в мові Сі описуються типом float. Числа з плаваючою крапкою аналогічні числам в звичайній алгебри запису, який використовується при роботі з дуже великими або малими числами.  
Приклад алгебраїчного запису чисел з плаваючою крапкою:

|  |  |  |
| --- | --- | --- |
| **Число** | **Алгебраїчний запис** | **Запис для вводу в машину** |
| 5000 | = 5.0 \* 103 | = 5.0e+3 |
| 0.000077 | = 7.7 \* 10-5 | = 7.7e-5 |

Зазвичай для розміщення в пам'яті числа з плаваючою крапкою відводиться 32 біта - 8 біт для представлення порядку і знака і 24 біта - для мантиси, тобто коефіцієнта при ступеня десяти. Для представлення даних типу double (обчислення з подвійною точністю) використовується подвоєне число бітів. Інший спосіб визначення даних типу double полягає в використанні ключових слів long float.

Змінні з плаваючою крапкою описуються та ініціалізуються таким же чином, що і змінні цілого типу.

Напрклад:

float dog, cat, bigword=5.77e+34;

**Константи з плаваючою крапкою.** У мові Сі є кілька можливостей записи констант з плаваючою крапкою. Найбільш загальна форма запису константи - це послідовність десяткових цифр зі знаком, що включає в себе десяткову крапку, потім символ e або Е і показник ступеня за основою 10 зі знаком. Знак (+) можна не писати. Нижче наведено кілька правильно записаних констант з плаваючою крапкою:

1.1e+12

3.14159

|  |  |
| --- | --- |
| ! | Під час розробки програми необхідно скласти список необхідних змінних і вказати при цьому, якого вони повинні бути типу. Описувати ці дані в самому початку тіла функції, в якій вони використовуються. Імена змінних вибирайте таким чином, щоб вони вказували на їх зміст. При ініціалізації змінної стежте за тим, щоб тип константи відповідав типу змінної, хоча мова Сі розглядає такі невідповідності менш жорстко, ніж Паскаль, але краще вчитися уникати поганого тону в програмуванні! |

### Символьні рядки

**Символьний ряд**- це послідовність символів, можливо порожня (""). Розглянемо приклад:

"Великий попит на освіту в області  
  інформаційних дисциплін пояснюється не тільки  
  популярністю комп'ютерів в сучасному суспільстві,  
  але і реальною користю від їх застосування. "

Лапки не є частиною рядка. Вони вводяться тільки для того, щоб відзначити її початок і кінець. У мові Сі немає спеціального типу, який можна було б використовувати для опису рядків. Замість цього рядка подаються у вигляді "набору" елементів типу char. Це означає, що символи в рядку можна уявити розташованими в сусідніх комірках пам'яті - по одному символу в комірці. Символ \ 0 в мові Сі використовується для того, щоб відзначати кінець рядка. Нуль-символ не виводиться на друк і в таблиці коду ASCII (American Standard Code for Information Interchange) має номер 0. Наявність нуль-символу означає, що кількість комірок масиву символів має бути на одну більше, ніж число символів рядка. Масив можна уявити як сукупність кількох осередків пам'яті, об'єднаних в один рядок.

**Масив -** це впорядкована послідовність елементів даних одного типу. У нашому прикладі ми створили масив з 177 елементів пам'яті, в кожну з яких можна помістити один символ типу char. Це можна зробити за допомогою оператора опису:

char String[177];

Квадратні дужки вказують, що змінна String - масив з 177 елементів, а char задає тип кожного елемента. Довжину рядка в символах (без завершального символу) визначає функція strlen (). Звернення до неї в нашому прикладі виглядає так:

strlen(String);

Результат - ціле число.

Препроцесор мови Сі

Повернемося до констант. Щоб ввести ту чи іншу константу в програму, потрібно вказати її фактичне значення, як було сказано вище. Можна використовувати замість цього "символічні константи" і дозволити компілятору замінити символічні константи числами. Як можна створити таку константу? Можна це зробити так:

float cost = 0.0012;

Такий спосіб задання констант у великих програмах неекономічний. У Сі є інший, кращий спосіб. Цей спосіб реалізується за допомогою препроцесора мови Сі. Препроцесор дає можливість задавати константи. Для цього в початок програми потрібно додати рядок, аналогічну наступною:

#define COST 0.0012

При компіляції програми кожен раз, коли з'явиться константа COST, вона буде замінюватися величиною 0.0012. Такий процес називається підстановкою під час компіляції. Зауваження з приводу формату: спочатку йде ключове слово #define (воно повинно починатися з самої лівої позиції), потім йде символічне ім'я константи, а потім її величина. Символ "крапка з комою" не використовується, тому що це не оператор мови Сі.

Якщо в якості першого символу в рядку програми використовується символ #, то цей рядок є командним рядком препроцесора (макропроцесора). Командний рядок препроцесора закінчується символом переведення на новий рядок. Якщо безпосередньо перед кінцем рядка поставити символ зворотної косої (\), то командний рядок буде тривати на наступному рядку програми.

Препроцесор використовується для обробки тексту програми до етапу її компіляції. Зазвичай препроцесори служили засобом розширення мов з метою забезпечення додаткових можливостей. Незважаючи на безліч препроцесорів, створених для розширення можливостей мов програмування, всі вони були нестандартними. Для деяких мов, наприклад для мов загального призначення ПЛ / 1 і Сі, препроцесори поставлялися як частина їх стандартного середовища.

Препроцесор для мови Сі забезпечує засоби для визначення макросів, визначення констант, включення файлів і умовну компіляцію. Препроцесор мови Сі викликається автоматично при зверненні до компілятору. Програма може бути оброблена тільки препроцесором без компіляції, якщо в команді сс вказати ключ-Е:

cc -E ім’я\_файлу

Результат роботи препроцесора поміщається в потік стандартного виведення stdout. Обробка програми препроцесором без компіляції дозволяє програмісту проаналізувати дію ухвал препроцесора і макровизовов.

|  |  |
| --- | --- |
| ! | Символічну константу після #define краще писати прописними літерами. У процесі використання мови Сі виробилася традиція писати константи прописними буквами. Якщо при перегляді програми зустрічається ім'я, написане великими буквами, відразу стає ясно, що це константа, а не змінна. Давайте не порушувати традицію! |

Директиву #define можна використовувати для визначення символьних і рядкових констант. У першому випадку необхідно використовувати "апостроф", а в другому лапки.

Наприклад:

#define NULL '\0'

#define USA '$'

#define RUSSIA "Рубль"

#undef USA

Команда #undef USA скасовує попереднє визначення для ідентифікатора USA. Через команду #define можна задавати вираження - макроси, які обчислюються і при компіляції підставляються в програму. Щоб уникнути помилок при обчисленні виразів макроозначення необхідно закривати в дужки:

#define ідентифікатор1 (ідентифікатор2,\_) рядок

Приклад:

#define abs(A) (((A)>0) ? (A): - (A))

Кожне входження вираження abs (arg) в тексті програми замінюється на ((arg> 0)? (Arg): - (arg)), причому параметр макроозначення A замінюється на arg.

Пpиклад:

#define nmem(P,N)\

(P)->p\_mem[N].u\_long

Символ \ продовжує макроозначення на другий рядок. Це макроозначення зменшує складність вираження, що описує масив об'єднань усередині структури.

Макроозначення препроцесора мови Сі мають дві форми - просту і параметризовану,

#define ідентифікатор рядок\_заміни

#define ідентифікатор (x1,x2,\_,xn) рядок\_заміни

де рядок\_заміни може містити ідентифікатори, ключові слова, роздільники, такі як кругла або прямокутна дужка, або рядки знаків, що не містять будь-яких роздільників.

**Примітка**. Командний рядок #include може зустрічатися в будь-якому місці програми, але зазвичай всі включення розміщуються на початку вихідного тексту:

#include < ім’я\_файлу >

Приклад:

#include <math.h>

Препроцесор заміняє цей рядок вмістом файла math.h. Кутові дужки означають, що файл math.h буде взятий з деякого стандартного каталогу, зазвичай це / usr / include. Поточний каталог не проглядається:

#include " ім’я\_файлу ".

Приклад:

#include "ABC"

Препроцесор заміняє цей рядок вмістом файла ABC. Так як ім'я файлу закрито в лапки, то пошук проводиться в поточному каталозі, в якому міститься основний файл вихідного тексту. Якщо в поточному каталозі даного файлу немає, то пошук проводиться в каталогах, визначених ім'ям шляху в опції -i препроцесора. Якщо і там файлу немає, то проглядається стандартний каталог.

Командні рядки препроцесора використовуються для умовної компіляції різних частин вихідного тексту в залежності від зовнішніх умов. **Умовної компіляцією** називається вибіркова компіляція тільки тих частин програми, які задовольняють певним умовам. Наприклад, можна скомпілювати тільки ті частини програми, які необхідні для конкретної версії системи.

Умовна компіляція має наступні переваги:

1. Забезпечується засіб параметризації під час компіляції. Наприклад, така можливість може бути використана для генерації програм з різною структурою.
2. Забезпечується ефективне використання оперативної пам'яті, оскільки під час виконання немає необхідності тримати в пам'яті невикористовувані коди програми.
3. Забезпечується можливість прийняття рішень під час компіляції, а не під час виконання. Часто це виявляється більш ефективним підходом.

Для умовної компіляції використовується інструкція препроцесора if. Вона має дві форми - з частиною else і без неї:

#if - заголовок

Текстові рядки для випадку "істина"

#endif

и

#if - заголовок

Текстові рядки для випадку "істина"

#else

Текстові рядки для випадку "неправда"

#endif

де if - заголовок є керуючою рядком препроцесора, а текстові рядки можуть містити довільний текст. Керуюча рядок препроцесора if -заголовка містить умову, на основі аналізу якого проводиться вибір відповідних текстових рядків:

#if константний\_вираз

Приклад:

#ifdef ABC

Істина, якщо ідентифікатор ABC визначений раніше

командою #define.

#ifndef ідентифікатор

Приклад:

#ifndef ABC

Істина, якщо ідентифікатор ABC не визначений

в даний момент.

#else

...

#endif

Якщо попередні перевірки #if, #ifdef або #ifndef дають значення Істина, то рядки від #else до #endif ігноруються при компіляції. Якщо ці перевірки дають значення Брехня, то рядки від перевірки до #else, а при відсутності #else - до #endif, ігноруються. Команда #endif позначає кінець умовної компіляції.

Приклад:

#ifdef DEBUG

fprintf (stderr, "location. X=%d\n",x);

#endif

За допомогою команд препроцесора можна змінити номер поточного рядка або ім'я файлу що компілюється:

#line ціла\_константа "ім’я\_файлу"

Приклад:

#line 20 "ABC"

Препроцесор змінює номер поточного рядка і ім'я файлу що компілюється. Файл може бути опущено.

**Примітка**. Хоча препроцесори і розширюють можливості мов програмування, їх використання не позбавлене недоліків: використання препроцесорів вимагає додаткового перегляду тексту програми і, як наслідок, додаткового часу.