

Основни синтактични елементи на С

1.Константи

а)целочислени константи-поддържа 3 бройни системи-10,8,16

Целочислените константи получават тип `int` или `long` в зависимост от стойността си. Осмичните и шестнайстичните константи получават тип `int` или `long` в зависимост от стойността си, но ако във водещия разряд попадне значеща цифра , то се преписва беззнаков `int (unsigned)` или `long`. Извън това подразбиране програмата може да управлява своите константи чрез суфикс.

Вътрешното представяне е в допълнителен код.

б)числени плаващи константи-записват се със или без експонента. Ако няма **e** константи те трябва да включват десетична точка, иначе не е задължително. Началото на експоненциалната част се обозначава с буквата

e
(e,E). Може да се пропуска както цялата, така и дробната част, но не и двете едновременно

Всички плаващи константи получават тип `double` , но програмиста може да управлява типа на константите чрез суфикс `F,f` (константите получават тип `float-4` байта). Разликата е в заетата памет.

IEEE 754-стандарт за предотвратяване на плаващите константи.

Стандарта предлага 2 форми

Лекции по С

Написано от
Петък, 02 Март 2012 07:21 -

2) експонентът е с излишък

$$EM = E_g + 127$$

$$EM = E_g + 1023$$

Всички положителни експоненти започват с 1 във водещия бит, а всички отрицателни с 0.

3) $1 \leq M$ не може да се присвои 1000, а само от 0 до 3.

Елементът може да има сложен вид:

[=]

Израза задава ново начало на броя за следващите константи на изброения тип

Плаващи данни

float (4 байта), double (8 байта), long double-10 байта

Масиви и указатели

Лекции по С

Написано от
Петък, 02 Март 2012 07:21 -

1. Масиви-името на масива може да се счита за данна с тип указател, тъй като масива не може да се мести по паметта – това е константа. Това е адреса на първия байт от паметта, който е разпределен за масива. За достъп до елементите на масива се използва така известната индекс операция.

[] - единият от 2та изрази трябва да бъде с характер на адрес, а другия трябва да се изчислява до целочислена стойност.

Израз1 е адресен израз.

Индексната операция се изпълнява по следния алгоритъм:

- изчислява се израз1

- изчислява се израз2 и се умножава по дължината на елементите (компонентите на масива).

- получените 2 стойности се събират и се получава адрес. От този момент индексната операция може да продължи по различен начин в зависимост от това дали се адресира сложна данна и дали индексиранието е вдясно или ляво на операцията за присвояване. Ако се адресира скаларен елемент и сме вдясно от равенството се извършва автоматично косвено адресиране, прочита се адресираната данна и това е резултата от операцията. Ако индексиранието е вляво от равенството изчисления адрес е получател на данните, които са резултат на израза вдясно от равенството. Ако се адресира сложна данна без значение е дали операцията е вляво или вдясно от равенството, изчислителния адрес е резултат от операцията. (Ако е вдясно този резултат може да се ползва, той е операнд, ако е вляво не може да се ползва – сигнал за грешка).

Данни указатели

Лекции по С

Написано от
Петък, 02 Март 2012 07:21 -

Над данна указател може да се извършва операция косвено адресиране *. То може да стои както вляво, така и вдясно на операцията за присвояване. Ако е вдясно се извършва прочитане на данните сочени от указателя, ако се сочи скаларна данна, иначе операцията е грешна. Ако е вляво от равенството адреса сочен от указателя е получател на данната.

±

Тази цяла стойност трябва да се изчисли като се умножи по дължината на данните сочени от указателя. Получената стойност се прибавя към адреса сочен от указателя и това е резултат от операцията. Данните указатели и масиви са сродни. Това произтича от факта, че името на всеки масив може да е различно като константа указател и от това, че използването на индекс операцията е твърде близко да използване на операцията ± .

Структури и обединения

```
struct[]{}
```

```
struct
```

```
union[]{}
```

```
union
```

Съкращения т.с. може да се запише само ако е в зоната на видимост на името на шаблона зададено при пълния спецификатор. Всички компонент на структурата едновременно присъстват в паметта т.е. те са едновременно активни. Всички компоненти на обединениетосподелят обща памет, чийто обем се изчислява от дължината на най-дългия вариант. Няма селектиране на активен вариант във всеки

Лекции по С

Написано от
Петък, 02 Март 2012 07:21 -

моминт може да се чете или записва от всеки вариант => всички варианти са едновременно активни, но споделят обща памет. Няма ограничение по отношение на данните, които могат да бъдат компонентите на структури и обединения. Структурата може да съдържа други структури и обединения, както и за вариантите важи същото.

Компонентите на структурата може да бъде типове поле. Записва се така:

[:],,,;

Типовия спецификатор трябва да определя целочислен тип знаков или биззнаков. Константния израз дава дължината на битовото поле в битове. Ако броят на битовете, който е необходим да се резервират надхвърли дължината на типовия спецификатор отпуска се автоматично нов компонент като преход между двата компонента не се реализира.

Битови полета не могат да бъдат компоненти на обединенията, но могат да бъдат структури, които включват битови полета. При разполагане на компонентите на структури и обединения могат да се прескочат байтове с цел да се излезе на подходяща адресна граница. Не трябва да се разчита, че дължината на данната задължително ще съвпада с изчислената, получена чрез дължините на отделните компоненти. Желателно е компонентите да се подреждат по големина, от най-голяма към най-малка.. Достъпа до компонентите може да се изпълни по 2 начина, както за структурите, така и за обединенията:

. изразът се изчислява до ст-ст с тип структура или обединение.

-> израът се изчислява до ст-ст адрес, който е указател към структура или обединение. Ако компонентът е сложна данна той може да се уточнява по същия начин.

Struct {

Лекции по С

Написано от
Петък, 02 Март 2012 07:21 -

```
int a,b;
```

```
int c[20];
```

```
struct {
```

```
char d[15];
```

```
float x;} y;
```

```
int z;} q; *p;
```

Всяка структура образува собствен клас. Доуточняват се структурите.

q.a

q.b

q.c[5]

[q.y.x](#)

[q.y.d](#) [2]

$p = \&q;$

[q.y.d](#) [2] $p \rightarrow y.d[2]$

Изрази

Видове операнди:

-константи: участват с ст-ст, която зависи от начина на записване. Конст. низове представяват адрес на 1-вия B от паметта където се съдържа низ

-имена: имената на скал.данни в израза представят текущата ст-ст на данната; имената на струк. и обединенията представят самите тях; имената на ф-ции по същество са адрес на вх. точка на ф-цията;

-обръщение към ф-ции: участват с върнатата от ф-ята ст-ст

-междинни резултати: операнди, които се получават при изпълнение на операциите

1. Приоритетна таблица на операциите.

Стрелката показва в какъв ред е приоритета.

1.() . -> [] ---> свръхприоритетни, те са операндно генериращи операции.

2. + - ~ ! * & мултипликативни операциите

4. + - ---> адитивни

5. ---> изместване

6. = ---> релации

7. == != ---> сравнение

8. & ---> побитово И

9. ^ ---> сума по mod2 (побитово)

10. | ---> побитово ИЛИ

11. && ---> логическо и

12. || ---> логическо или

13. !: ако определя някаква памет. При **суфиксно записване** – прочита се операнда и това е резултата от операцията. Паралелно и независимо се променя и паметта свързана с него с 1.

Лекции по С

Написано от
Петък, 02 Март 2012 07:21 -

```
Int i, j, k, m;
```

```
i=10;
```

```
j=5;
```

```
k=++i; // k=11, i=11
```

```
m=j++; // m=5, j=6
```

size of () - Операндът може да е данна или типово име. Резултатът от операцията е брой байтове необходими за съхраняване на данната или низа. Ако се използва име на масив, не се получава паметта на указателя, а се получава обема памет за самия масив.

```
Int a[5]
```

```
size of(a)->10
```

```
size of(int)->2
```

типово преобразуване () - Типът на операнда се преобразува насилствено към посоченото типово име.

б) мултипликативни операции - *, / - За цели и плаващи данни, % само за целочислени

данни.% - остатъкът от делението

в) адитивни операции - +, - , цели и плаващи данни и указатели. Указател – указ. е реализуема операция, ако двата указателя са насочени към един и същ масив. Изваждат се двата указателя, резултата се дели на дължината на данните сочени от указателя и това е резултата от операцията. Дава разлика като брой елементи.

г) изместване - Двата операнда трябва да са целочислени, измества се първия, а втория определя на колко разряда. При изместване на дясно, ако местения операнд е знаков, размножава се знака т.е. прави се аритметично изместване, а ако е беззнаков се прави логическо изместване на разрядната решетка и влизат нули.

д) Релации – цели и плаващи данни и иказатели.ОАП. Указателите могат да се сравняват и за само ако са насочени към един и същи масив;сравняват се за == и != винаги, но не по адрес, а по разредност на решетката. Резултатът е 0, ако релацията не е изпълнена и 1, ако е изпълнена.

A0:f1:f2)(...) // Частен случай: име

Списъкът изрази в извикването представлява аргументите за текъщата активация. Изразите магат да се изчисляват за текущата активация. Изразите могат да се изчисляват за произволни данни с изключение на функции и масиви. Масив може да се използва, но той се тълкува като указател. Име на функция също може да се замести, но тя се тълкува като указател.

Извикването предизвиква три проблема:

- съответствието формален параметър – аргумент се извършва по позиция – То е позиционно! Самото заместване на данните е винаги по стойност! Няма заместване по указател!

Лекции по C

Написано от
Петък, 02 Март 2012 07:21 -

Ако искаме да предаваме по адрес – правим формален параметър, указатели, работим с указатели, а отвън подаваме адрес. За да е възможна работата с променлив брой аргументи, те се обработват от дясно на ляво и така се заместват в стек. По този начин задължително декларирания брой формални параметри винаги ще намери точно съответствие с някакви аргументи. Тази конвенция за заместване носи името C-конвенция.