

Структура на програма в С

Част 4 - оператори II

Алекс Осенов, Александър Иванов, Александър Чернаев,
Веселин Русинов, Иван Георгиев, Пламен Тенев, Христо Иванов,
Христо Стефанов

Технологично училище "Електронни системи",
Технически университет, София

28 февруари 2020 г.

Съдържание

- 1 Приоритет на операциите. Асоциативност. Резултат от операция.
- 2 Присвояване
- 3 Аритметични оператори
- 4 Логически оператори
- 5 Сравняващи (релационни) оператори
- 6 Приоритет и асоциативност на операциите в C

Съдържание

- 1 Приоритет на операциите. Асоциативност. Резултат от операция.
- 2 Присвояване
- 3 Аритметични оператори
- 4 Логически оператори
- 5 Сравняващи (релационни) оператори
- 6 Приоритет и асоциативност на операциите в C

Приоритет на операциите

- За да може един израз да бъде изпълнен, трябва да се определи в какъв ред да се изпълнят операциите в израза
- В математиката операциите се изпълняват започвайки от операциите в най-вътрешните скоби на израза, след което се преминава към операциите в по-външните скоби и т.н. докато има скоби или операциите не свършат, а в рамките на скобите операциите се извършват спрямо техния приоритет
- По същите правила се изпълняват и изразите в С

$$(a \times b) - c \div d = (a \times b) - (c \div d)$$

Фигура 1: Приоритет на изпълнение на операции в математически израз

Асоциативност

- Когато имаме последователност от едни и същи операции възниква въпроса, коя от операциите да извършим първо
 - a^{b^c} се изпълнява като $a^{(b^c)}$ или като $(a^b)^c$?
- Асоциативността на една операция помага за определянето на реда на изпълняване на даден израз, когато в него има последователно прилагане на една и съща операция или операции със един и същ приоритет
 - Дясно асоциативните операции изпълняват първо всички операции в дясно от себе си
 - Ляво асоциативните операции изпълняват първо всички операции в ляво от себе си
- В математиката събиране, изваждане, умножение и деление са ляво асоциативни операции, а повдигането на степен е дясно асоциативна операция

Асоциативност

$$a \div b \div c \div d = (((a \div b) \div c) \div d)$$

Фигура 2: Пример за лява асоциативност на операцията делене

$$a^{b^{c^d}} = a^{(b^{(c^d)})}$$

Фигура 3: Пример за дясна асоциативност на операцията повдигане на степен

Резултат от операция

- В математиката резултата от извършване на операция е число с определена големина
- В С възможните резултати от изпълнение на операция са два вида - *временна променлива* или *временен идентификатор (временно име)* на променлива.
- Ако резултата е временна променлива, в нея се запазва стойността от изпълнението на операцията
- Ако резултата е временен идентификатор (временно име) на променлива, тогава резултата може да се използва като идентификатор (име) за същата променлива
 - В математиката аналог на временните идентификатори е полагането - чрез него можем да укажем друго име за променлива. Например ако положим $p = x$, то уравнението $x^2 + x + 1 = 0$ е същото като $p^2 + p + 1$.

Съдържание

- 1 Приоритет на операциите. Асоциативност. Резултат от операция.
- 2 Присвояване
- 3 Аритметични оператори
- 4 Логически оператори
- 5 Сравняващи (релационни) оператори
- 6 Приоритет и асоциативност на операциите в C

Присвояване

- В езика С стойността на някоя променлива може да бъде променена чрез *присвояване* на нова стойност
- Операцията присвояване се отбелязва с *оператор за присвояване*

op1 = op2

Фрагмент 1: Синтаксис на оператор за присвояване

Оператор за присвояване

- За да може да бъде извършена операцията присвояване, трябва като първи операнд да се посочи идентификатор на променлива, а като втори - произволен израз
 - Не е възможно да се присвояват нови стойности на константи/литерали, тъй като самия смисъл на константата/литерала забранява смяната на стойността
 - Присвояването на стойност на временна променлива няма практичесен смисъл и затова е забранено

Оператор за присвояване

- Резултата от изпълнението на оператор за присвояване е временна променлива със стойност - новата стойност на променливата, на която присвояваме
- Присвояването е дясно асоциативна операция

{

```
int a = 10;
int b = 20;
a = 5; // OK, a is an identifier
5 = a; // ERROR, 5 is not an identifier, it is a constant
(a = 5) = 10; /* ERROR, the result of the expression '(a = 5)'
                 is not an identifier, it is a temporary variable */
a = b = 40; // OK, assignment is right-associative operation
```

}

Фрагмент 2: Правилна и неправилна употреба на оператор за присвояване

Съдържание

- 1 Приоритет на операциите. Асоциативност. Резултат от операция.
- 2 Присвояване
- 3 Аритметични оператори
- 4 Логически оператори
- 5 Сравняващи (релационни) оператори
- 6 Приоритет и асоциативност на операциите в C

Аритметични оператори

- Езикът С позволява извършването на аритметични операции, използвайки *аритметични оператори*

| Име на оператора | Синтаксис ¹ | Операция |
|--------------------|------------------------|-------------------------------------|
| Събиране | $op1 + op2$ | Събиране на $op1$ и $op2$ |
| Изваждане | $op1 - op2$ | Изваждане на $op1$ и $op2$ |
| Умножение | $op1 * op2$ | Умножение на $op1$ и $op2$ |
| Деление | $op1 / op2$ | Деление на $op1$ и $op2$ |
| Остатък от деление | $op1 \% op2$ | Остатък от деление на $op1$ и $op2$ |

Таблица 1: Бинарни аритметични оператори в С

¹ $op1$ и $op2$, могат да са произволни изрази

Бинарни аритметични оператори

- Бинарните аритметични операции работят винаги върху данни от един и същ тип и връщат временна променлива от същия тип
- Ако се подадат operandи от различен тип, се извършва *неявно преобразуване на типа* (*implicit type cast/conversion*) на някой от operandите, за да се уеднаквят типовете
- Неявното преобразуване преобразува типа на operandата с по-малък обхват от стойности към типа на operandата с по-голям обхват от стойности
- Например:

```
{  
    int a = 4;  
    long int b = 7;  
    long int c = a + b; // OK, 'a' is implicitly converted to 'long int'  
    float d = 57.8;  
    float e = d + c; // OK, 'c' is implicitly converted to 'float'  
}
```

Фрагмент 3: Неявно преобразуване на типове при бинарни аритметични оператори

Унарни аритметични оператори (1/2)

- Операторът за смяна на знак сменя знака на стойността на своя operand и връща резултата във временна променлива.
- Операторът унарен плюс връща стойността на операнда непроменена във временна променлива.

| Име на оператора | Синтаксис ² |
|------------------------------|------------------------|
| Смяна на знак (унарен минус) | -op1 |
| Унарен плюс | +op1 |

Таблица 2: Унарни аритметични оператори в С (1/2)

²op1 може да е произволен израз

Унарни аритметични оператори (2/2)

- Съществуват поредици от операции, които много често се използват при писането на програми
- Езикът С предоставя операции, които изпълняват някоя от тези поредици като една операция, с цел по-малко писане на код

| Име на оператора | Синтаксис ³ |
|---------------------------|------------------------|
| Префиксно инкрементиране | <code>++op1</code> |
| Постфиксно инкрементиране | <code>op1++</code> |
| Префиксно декрементиране | <code>--op1</code> |
| Постфиксно декрементиране | <code>op1--</code> |

Таблица 3: Унарни аритметични оператори в С (2/2)

³ $op1$ може да е само идентификатор

Унарни аритметични оператори

- *Операторът за префиксно инкрементиране* увеличава стойността на своя operand с 1 и връща новата стойност на operand-a като временна променлива. Operandът е задължително идентификатор.
- *Операторът за постфиксно инкрементиране* увеличава стойността на своя operand с 1 и връща старата стойност на operand-a като временна променлива. Operandът е задължително идентификатор.
- *Операторът за префиксно декрементиране* намалява стойността на своя operand с 1 и връща новата стойност на operand-a като временна променлива. Operandът е задължително идентификатор.
- *Операторът за постфиксно декрементиране* намалява стойността на своя operand с 1 и връща старата стойност на operand-a като временна променлива. Operandът е задължително идентификатор.

Унарни аритметични оператори

```
{  
    int a = 1;  
    int b = 1;  
    b = ++a; // b is now 2, a is now 2  
    b = a++; // b is still 2, a is now 3  
    b = -a; // b is now -3, a is unchanged  
    a = b--; // a is now -3, b is now -4  
    a++; // a is now -2  
    b = +a; // b is now -2, a is unchanged  
}
```

Фрагмент 4: Работа с унарни аритметични оператори

Съдържание

- 1 Приоритет на операциите. Асоциативност. Резултат от операция.
- 2 Присвояване
- 3 Аритметични оператори
- 4 Логически оператори
- 5 Сравняващи (релационни) оператори
- 6 Приоритет и асоциативност на операциите в C

Логически оператори

- Логическите оператори изпълняват логически операции за истинност върху своите operandи
- При изпълнение на логическите операции за истинна стойност се приема всяка стойност на operand различна от 0, а за лъжлива стойност, стойност на operand равна на 0
- Резултата от операциите са или истина или лъжа. При истина винаги се връща временна променлива със стойност 1, а при лъжа - със стойност 0
- При изпълнение се изчислява първо оп1 и само ако не може да се определи резултата на операцията от оп1, се изчислява и оп2

| Име на оператора | Синтаксис ⁴ | Операция |
|------------------|------------------------|---------------|
| Логическо И | оп1 && оп2 | Логическо И |
| Логическо ИЛИ | оп1 оп2 | Логическо ИЛИ |
| Логическо НЕ | !оп1 | Логическо НЕ |

Таблица 4: Логически оператори

⁴оп1 и оп2 могат да са произволни изрази

Логически оператори

```
{  
    int a = 10;  
    int b = 0;  
    int c = 10 || 3; // 'c' is now 1  
    int d = 0 && 10; // 'd' is now 0  
    int e = !c; // 'e' is now 0  
    int f = c || a++; // 'f' is now 1, 'a' is still 10  
    int g = b || a++; // 'g' is now 1, 'a' is now 11  
}
```

Фрагмент 5: Работа с логически операции

Съдържание

- 1 Приоритет на операциите. Асоциативност. Резултат от операция.
- 2 Присвояване
- 3 Аритметични оператори
- 4 Логически оператори
- 5 Сравняващи (релационни) оператори
- 6 Приоритет и асоциативност на операциите в C

Сравняващи (релационни) оператори

- Сравняващите оператори (релационни оператори) са бинарни оператори, които сравняват два операнда
- Аналогично на бинарните аритметични оператори, ако типовете на operandите се различават, се извършва *неявно преобразуване на типовете на operandите*
- Ако релацията между operandите е изпълнена, резултатът е временна променлива със стойност 1, иначе е временна променлива със стойност 0

Сравняващи (релационни) оператори

| Име на оператора | Синтаксис ⁵ | Стойност на върнатата врем. променлива |
|------------------------|------------------------|---|
| Равно на | $op1 == op2$ | 1, ако са равни, иначе 0 |
| Различно от | $op1 != op2$ | 1, ако не са равни, иначе 0 |
| По-голямо от | $op1 > op2$ | 1, ако стойността на $op1$ е по-голяма, иначе 0 |
| По-малко от | $op1 < op2$ | 1, ако стойността на $op1$ е по-малка, иначе 0 |
| По-голямо или равно на | $op1 >= op2$ | 1, ако стойността на $op1$ е по-голяма или равна на $op2$, иначе 0 |
| По-малко или равно на | $op1 <= op2$ | 1, ако стойността на $op1$ е по-малка или равна на $op2$, иначе 0 |

Таблица 5: Сравняващи (релационни) оператори

⁵ $op1$ и $op2$ могат да са произволни изрази

Съдържание

- 1 Приоритет на операциите. Асоциативност. Резултат от операция.
- 2 Присвояване
- 3 Аритметични оператори
- 4 Логически оператори
- 5 Сравняващи (релационни) оператори
- 6 Приоритет и асоциативност на операциите в C

Приоритет и асоциативност на операциите в С

| Приоритет | Оператор | Операция | Асоциативност |
|-----------|----------|----------------------------------|------------------|
| 1 | ++ | Постфиксно инкрементиране | От ляво на дясно |
| | - | Постфиксно декрементиране | |
| | () | Извикване на функция | |
| | [] | Индексиране в масив | |
| | . | Избор на елемент | |
| | -> | Избор на елемент чрез указател | |
| 2 | ++ | Префиксно инкрементиране | От дясно на ляво |
| | -- | Префиксно декрементиране | |
| | + | Унарен плюс | |
| | - | Унарен минус | |
| | ! | Логическо НЕ (отрицание) | |
| | ~ | Побитово НЕ (побитово отрицание) | |
| | (type) | Преобразуване до тип | |
| | * | Индирекция (дереферериране) | |
| | & | Адрес на | |
| | sizeof | Размер в байтове на | |

Приоритет и асоциативност на операциите в С

| Приоритет | Оператор | Операция | Асоциативност |
|-----------|----------|-----------------------------|------------------|
| 3 | * | Умножение | От ляво на дясно |
| | / | Деление | |
| | % | Остатък от деление | |
| 4 | + | Събиране | От ляво на дясно |
| | - | Изваждане | |
| 5 | << | Побитово отместване в ляво | От ляво на дясно |
| | >> | Побитово отместване в дясно | |
| 6 | < | По-малко от | От ляво на дясно |
| | <= | По-малко или равно на | |
| | > | По-голямо от | |
| | >= | По-голямо или равно на | |
| 7 | == | Равно на | От ляво на дясно |
| | != | Различно от | |
| 8 | & | Побитово И | От ляво на дясно |
| 9 | ^ | Побитово изключващо ИЛИ | От ляво на дясно |
| 10 | | Побитово ИЛИ | От ляво на дясно |

Приоритет и асоциативност на операциите в С

| Приоритет | Оператор | Операция | Асоциативност |
|-----------|----------|--|------------------|
| 11 | && | Логическо И | От ляво на дясно |
| 12 | | Логическо ИЛИ | От ляво на дясно |
| 13 | ?: | Тернарно условие | От дясно на ляво |
| 14 | = | Пряко присвояване | От дясно на ляво |
| | += | Присвояване чрез сума с | |
| | -= | Присвояване чрез разлика с | |
| | *= | Присвояване чрез произведение с | |
| | /= | Присвояване чрез частно с | |
| | %= | Присвояване чрез остатък с | |
| | <=> | Присвояване чрез побитово отм. в ляво | |
| | >=> | Присвояване чрез побитово отм. в дясно | |
| | &= | Присвояване чрез побитово И | |
| | ^= | Присвояване чрез побитово изкл. ИЛИ | |
| | = | Присвояване чрез побитово ИЛИ | |
| 15 | , | Запетая | От ляво на дясно |