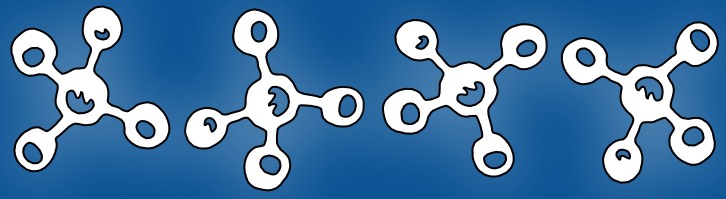
Алгоритмы и Алгоритмические Языки

Семинар #4:

1. Базовые элементы C: операции, выражения, операторы.
2. Битовые операции.
3. Целочисленное деление.
4. Приведение типов.

20.09.2024

Базовые элементы языка: операции, выражения и операторы



Операции (operators)

Операция (operator) – символ или набор символов, задающий некоторую схему вычисления.
Операция (operator) имеет:

- **Арность** и **ассоциативность**.
- **Значение** (результат операции).
- **Побочный эффект** (изменение состояния памяти и состояния процессора).

Арность уже известных операций:

- Унарные операции: **-** (знак “минус”), **+** (знак “плюс”)
- Бинарные операции: **+** (сложение), **=** (присваивание)
- Тернарная операция: **_ ? _ : _** (условная операция)

Операции (operators)

Ассоциативность уже известных операций:

- Левоассоциативная операция: - (вычитание)
  `x - y - z = (x - y) - z`
- Ассоциативная операция: + (сложение)
  `x + y + z = (x + y) + z = x + (y + z)`
- Тоже ассоциативная: композиция функций (математика)
  `(f · g · h)(x) = (f · (g · h))(x) = f(g(h(x)))`

Побочные эффекты операций:

- Операция “=” меняет значение левого операнда на значение правого.

Операции (operators)

Common operators

assignment	increment decrement	arithmetic	logical	comparison	member access	other

`a = b`
`a += b`
`a -= b`
`a *= b`
`a /= b`
`a %= b`
`a &= b`
`a |= b`
`a ^= b`
`a <<= b`
`a >>= b`
	`++a`
	`--a`
	`a++`
	`a--`
		`+a`
		`-a`
		`a + b`
		`a - b`
		`a * b`
		`a / b`
		`a % b`
		`~a`
		`a & b`
		`a | b`
		`a ^ b`
		`a << b`
		`a >> b`
			`!a`
			`a && b`
			`a || b`
				`a == b`
				`a != b`
				`a < b`
				`a > b`
				`a <= b`
				`a >= b`
					`a[b]`
					`*a`
					`&a`
					`a->b`
					`a.b`
						`a(...)`
						`a, b`
						`(type) a`
						`a ? b : c`
						`sizeof`
						`_Alignof`
						(since C11)

Нужны уже сейчас

Выражения (expressions)

Выражение (expression) – это:

- Имя переменной,
- или имя функции,
- или литерал, задающий константу,
- или операция над другими выражениями.

Выражение имеет:

- **Значение** (результат вычисления выражения),
- **Тип** (`int`, `float`, …),
- **Побочный эффект**.

Примеры выражений:

```
exam = 1.0          x + y + 14                  printf(“Hello!\n”)
777                 “E=mc^2, oh, MOOOOO”        sum_ext < SHRT_MIN
(a - 2) * 4         i < cowsay_length           (long) val
```

Операторы (statements)

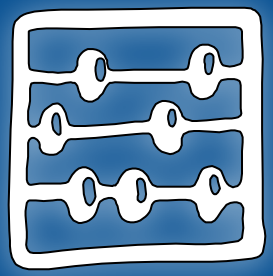
Оператор (statement) – языковая конструкция, задающая набор действий для выполнения.

Операторы не имеют типа и значений, а имеют только побочные эффекты.

Примеры операторов:

```
- expression;
- { statement }
- if (expression) statement [else statement]
- while (expression) statement
- return expression;
  ...
```

Битовые операции



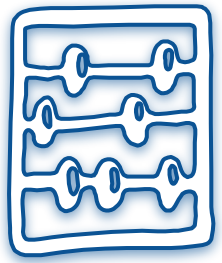
Битовые операции

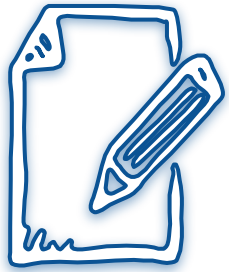
Битовые операции:

Синтаксис	Название
~a	Побитовое НЕ, NOT, инверсия
a | b	Побитовое ИЛИ, OR, дизъюнкция
a & b	Побитовое И, AND, конъюнкция
a ^ b	Побитовое ИСКЛЮЧАЮЩЕЕ ИЛИ, XOR
a << b	Сдвиг a на b бит влево
a >> b	Сдвиг a на b бит вправо

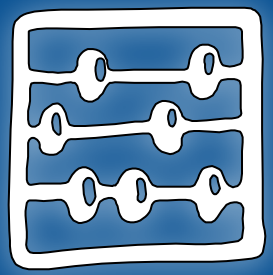
Битовые операции: задачки

1. Вычислить $x \cdot 2^y$.
2. Занулить 12 младших бит в 32-битном числе x.
3. Поменять биты с 8 по 15 в в 32-битном числе x
   на противоположные.
4. Занулить k младшие бит беззнакового числа x.
5. Изменить порядок следования байт в 32-битном числе x.





Целочисленное деление



Целочисленное деление

Деление с остатком в математике: **остаток – неотрицателен.**

$$a \equiv r \pmod{b} \Leftrightarrow \exists q \in \mathbb{Z} : a = qb + r, 0 \leq r < b$$

(Остаток – неотрицательное число)

В языке Си: **знак остатка совпадает со знаком делимого**.

$$a \equiv r \pmod{b} \Leftrightarrow \exists q \in \mathbb{Z} : a = qb + r$$

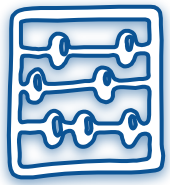
$$-b < r < b, \operatorname{sign} r = \operatorname{sign} a$$

```
q = a / b
r = a % b
```

Целочисленное деление: задача

1. Для заданных пар **a** и **b** произвести деление по модулю**:**

a	b	q = a / b	r = a % b
25	8	???	???
25	-8	???	???
-25	8	???	???
-25	-8	???	???



Приведение типов

Приведение типов

Неявное приведение типов:

- При приравнивании, передаче аргумента, при `return`-е.
- При арифметических операциях.

Пример с приведением типов:

```
// Loss of precision due to 32-bit arithmetic:
float operand_a = 100000000.0f;
float operand_b = 1.0f;

int result_with_losses = operand_a + operand_b;

printf("(float)%f + (float)%f = (int)%d\n",
    operand_a, operand_b, result_with_losses);
```

Приведение типов

Явное приведение типа:

```
int result_lossless = operand_a + (double) operand_b;

printf("(float)%f + (double)%f = (int)%d\n",
    operand_a, (double) operand_b, result_lossless);
```

Правила неявных преобразований в арифметике:

rank(bool) < rank(signed char) <
< rank(short) < rank(int) < rank(long) < rank(long long) <
< rank(float) < rank(double) < rank(long double)

rank(signed X) = rank(unsigned X)
rank(signed char) = rank(char) = rank(unsigned char)

Приведение типов в стандарте C

The following may be used in an expression wherever an **int** or **unsigned int** may be used:

— An object or expression with an integer type (other than **int** or **unsigned int**) whose integer conversion rank is less than or equal to the rank of **int** and **unsigned int**.

— A bit-field of type **_Bool**, **int**, **signed int**, or **unsigned int**.

If an **int** can represent all values of the original type (as restricted by the width, for a bit-field), the value is converted to an **int**; otherwise, it is converted to an **unsigned int**. These are called the *integer promotions*.[59] All other types are unchanged by the integer promotions.

The integer promotions preserve value including sign. As discussed earlier, whether a "plain" **char** can hold negative values is implementation-defined.

Приведение типов в стандарте C

First, if the corresponding real type of either operand is **`long double`**, the other operand is converted, without change of type domain, to a type whose corresponding real type is **`long double`**.

Otherwise, if the corresponding real type of either operand is **`double`**, the other operand is converted, without change of type domain, to a type whose corresponding real type is **`double`**.

Otherwise, if the corresponding real type of either operand is **`float`**, the other operand is converted, without change of type domain, to a type whose corresponding real type is **`float`**.[64)]

Otherwise, the integer promotions are performed on both operands. Then the following rules are applied to the promoted operands:

Приведение типов в стандарте C

Otherwise, the integer promotions are performed on both operands. Then the following rules are applied to the promoted operands:

If both operands have the same type, then no further conversion is needed.

Otherwise, if both operands have signed integer types or both have unsigned integer types, the operand with the type of lesser integer conversion rank is converted to the type of the operand with greater rank.

Otherwise, if the operand that has unsigned integer type has rank greater or equal to the rank of the type of the other operand, then the operand with signed integer type is converted to the type of the operand with unsigned integer type.

Otherwise, if the type of the operand with signed integer type can represent all of the values of the type of the operand with unsigned integer type, then the operand with unsigned integer type is converted to the type of the operand with signed integer type.

Otherwise, both operands are converted to the unsigned integer type corresponding to the type of the operand with signed integer type.

Приведение типов: алгоритм (1/2)

Пусть есть переменная **a** типа **A** и переменная **b** типа **B**.
Рассмотрим алгоритм получения типа **C** выражения (**a + b**).

1) Если **A = long double** или **B = long double**, то **C = long double**.
2) Иначе, если **A = double** или **B = double**, то **C = double**.
3) Иначе, если **A = float** или **B = float**, то **C = float**.

Если (1)-(3) не выполнены, то **A** и **B** - это целочисленные типы.
4) Для **A** и **B**: **rank(A/B) ≤ rank(int)**, то **A/B = int** или **A/B = unsigned int**.
5) Если при этом **A = B**, то **C = A = B**.

Если (1)-(5) не выполнены, то **A ≠ B**.
6) Если при этом **A** и **B** оба **signed**, или оба **unsigned**
и **rank(A) ≤ rank(B)**, то **C = B.**

Приведение типов: алгоритм (2/2)

Если (1)-(6) не выполнены, то **A** и **B** имеют разную знаковость.
Пусть **S** – знаковый тип из **A** и **B**, а **U** – беззнаковый тип из **A** и **B**.
7) Если при этом **rank(S) ≤ rank(U)**, то **C = U**.

Если (1)-(7) не выполнены, то **rank(S) > rank(U)**.
8) Если при этом все значения **U** представимы в типе **S**, то **C = S**.

Если (1)-(8) не выполнены, то не все значения **U** представимы в типе **S**.
Такое возможно для типов **unsigned long** и **long long**, т.к. может быть **ULONG_MAX > LLONG_MAX**.
9) В таком случае будет **C = unsigned S**.
При **A = unsigned long** и **B = long long** будет **C = unsigned long long**.

Приведение типов: задачи

1. При заданных переменных:

```
int a = 5, b = 7;
unsigned c = 3, d = 8;
float f = 0.3;
```

```
short p = 3, q = 8;
long long x = 19;
double g = 104;
```

Определите типы выражений:

1. `q / p`
2. `d * a`
3. `x = f`
4. `f + x`
5. `g + f`
6. `x / c`

Приведение типов: задачи

1. При заданных переменных:

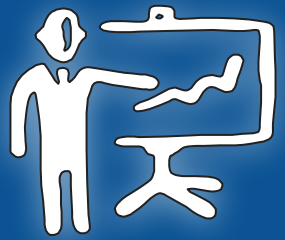
```
int a = 5, b = 7;
unsigned c = 3, d = 8;
float f = 0.3;
```

```
short p = 3, q = 8;
long long x = 19;
double g = 104;
```

Определите типы выражений:

1. `q / p (int, правило (4))`
2. `d * a (unsigned, правило (6))`
3. `x = f (long long, это просто тип x)`
4. `f + x (float, правило (3))`
5. `g + f (double, правило (2))`
6. `x / c (long long, правило (7))`

Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com