Алгоритмы и Алгоритмические Языки

Семинар #4:

- Приведение типов;
- Приоритет операций;
- Порядок вычисления в выражениях;
- Основные операторы языка Си;

19.09.2023

Приведение типов

Системы типов разных языков

Язык Си - статически строго типизированный язык.

Строгая типизация – у каждой переменной есть тип,
у всех операций есть строгие требования по типам операндов
(`C`, `Haskell`, `Python`).
Нестрогая типизация – требования по типам операндов
более слабые или отсутствуют (`JS`).

Статическая типизация – типы переменных известны при их декларации, во время компиляции (`C`, `Haskell`).
Динамическая типизация – типы переменных известны только во время исполнения, и могут меняться (`Python`, `JS`).

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);
```

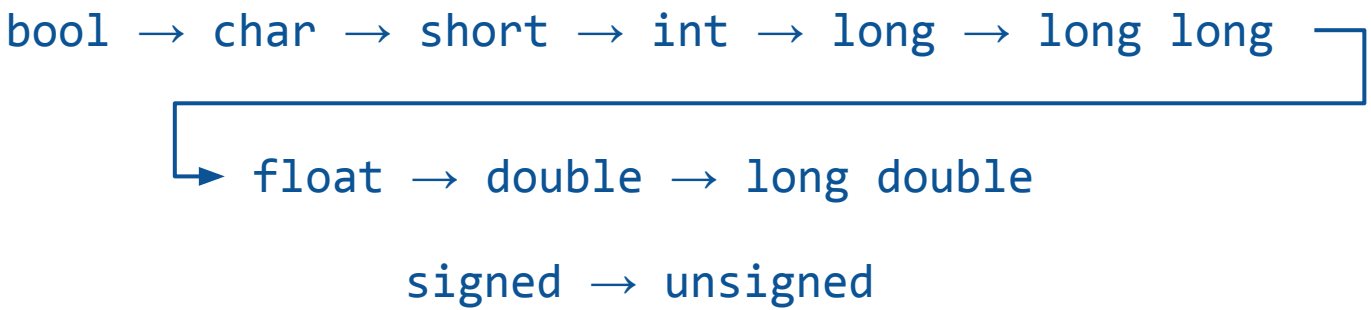
Приведение типов

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);
```

Общие правила преобразований в арифметике:



Приведение типов: задачи

1. Определите типы переменных `rslt1-rslt6`:

```
int          a = 1;
short        b = 2;
unsigned     c = 3;
long long    d = 4;
float        e = 5;
double       f = 6;
```

```
typeof(b / b) rslt1;
typeof(a * c) rslt2;
typeof(d = e) rslt3;
typeof(d + e) rslt4;
typeof(e + f) rslt5;
typeof(d / c) rslt6;
```

Приоритет операций

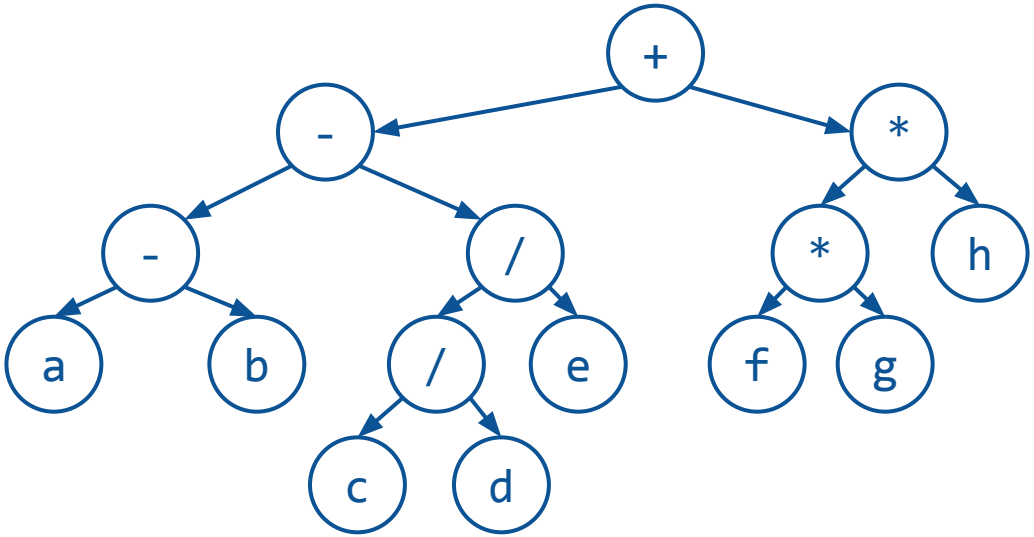
Интерпретация выражений

Рассмотрим выражение языка Си:

```
a - b - c / d / e + f * g * h
```

Что нужно знать, чтобы однозначно расставить скобки?

```
((a - b) - ((c / d) / e)) + ((f * g) * h)
```



Приоритеты операций

Ответ: приоритеты и ассоциативности всех операций!

Precedence	Operator	Description	Associativity
1	`++ --`	Suffix/postfix increment and decrement	Left-to-right
	`()`	Function call	
	`[]`	Array subscripting	
	`.`	Structure and union member access	
	`->`	Structure and union member access through pointer	
	`(`*type*`){`*list*`}`	Compound literal(C99)	
2	`++ --`	Prefix increment and decrement[note 1]	Right-to-left
	`+ -`	Unary plus and minus	
	`! ~`	Logical NOT and bitwise NOT	
	`(`*type*`)`	Cast	
	`*`	Indirection (dereference)	
	`&`	Address-of	
	`sizeof`	Size-of[note 2]	
	`_Alignof`	Alignment requirement(C11)	

Приоритеты операций

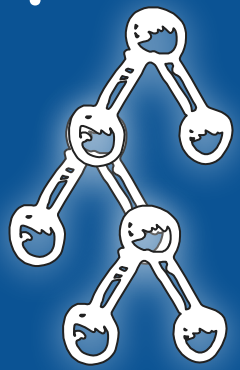
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <= > >=	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13	?:	Ternary conditional[note 3]	Right-to-left
14[note 4]	= += -= *= /= %= <<= >>= &= ^= |=	Simple assignment Assignment by sum and difference Assignment by product, quotient, and remainder Assignment by bitwise left shift and right shift Assignment by bitwise AND, XOR, and OR	
15	,	Comma	Left-to-right

Приоритеты операций: задачи

Убрать из записи выражений избыточные скобки:

1. `i = ((i++) + (++i))`
2. `d ^= ((a & b) | c)`
3. `x = (((a < b)? (a + 1) : b) - 1)`
4. `(x >> (y & 0xF)) & 0xF`
5. `(a & 0x0F) | (b & 0xF0)`

Порядок вычисления в выражениях



Порядок вычисления: задачи

Вычислить значение переменной x после выполнения фрагментов кода:

```
int i = 5;
int x = i++ + ++i;
```

```
int i = 5;
int x = (i -= 1) + ++i;
```

Порядок вычисления в выражениях

В общем случае порядок вычисления подвыражений не определён (Unspecified Behavior), он задаётся компилятором и платформой.

Точка последовательных вычислений (sequence point) между выражениями A и B – гарантия того, что побочные эффекты выражения A будут видны выражению B.

Список sequence point-ов:

- Между вычислением аргументов ф-ии и её вызовом;
- После вычисления 1-го аргумента `&&` и `||`, `?:` и `,`;
- После оператора или объявления переменных;
- Конец вычисления управляющих выражений `if`/`for`/`while`/`do`.

Порядок вычисления в выражениях

Два правила, нарушение которых приводит к UB.

Между двумя последовательными sequence point-ами С одним объектом нельзя производить неупорядоченные операции:

- Записи и записи (W+W);
- Чтения и записи (R+W), если W и R – не в левой и правой частях оператора присваивания.

Порядок вычисления: задачи

Указать выражения, содержащие UB:

1. `i = i + 1`
2. `i = i++`
3. `i++ * i++`
4. `i++ && i++`
5. `a[i] = i++`
6. `a[i++] = i`
7. `i = f(i++)`
8. `i += i`
9. `a ^= b ^= a ^= b`
10. `a ^= b, b ^= a, a ^= b`
11. `i++ ? i++ : ++i`

HSE University

Основные операторы языка Си

Условные операторы

Оператор if:

```
if (expr) statement
if (expr) statement else statement
```

Оператор switch:

```
switch (expr)
{
    case constant: statement
    ...
    default: statement
}
```

Циклы

Оператор while:

```
while (expr) statement
```

Оператор do-while:

```
do statement while (expression);
```

Оператор for:

```
for (init; cond; iteration) statement
```

Операторы переходов

Оператор break:

break;

Выход из цикла;

Оператор continue:

continue;

Переход к следующей итерации цикла;

Оператор return:

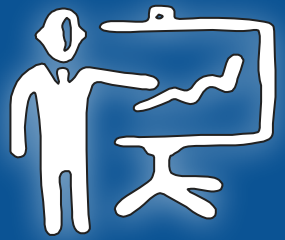
return expression;

Выход из функции, передача возвращаемого значения;

Оператор goto и декларация метки:

<Использование не рекомендовано>

Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com