

Expressions

Chapter 2.3 (part)

Slides #4

$$6 - 1 \times 0 + 2 \div 2 = ?$$

**ANSWER
IT**



Topics

- 1) How can we calculate values?
- 2) How to work with constants like 24 hours-per-day?
- 3) How high (low?) can we count?

Math Expressions

(And not like "Wow! Math is great!")

Expressions

- Expression:
 - A statement that...
 - Usually has an operator.
- Examples:
 - `result = 3;`
 - `result = 1 * x + 2;`
- Expressions usable anywhere a value is needed:
 - `println("Big number {}", 1 + 2);`

Order of Operations

- What is the value of result?
 `int result = 4 + 10 / 2;`
 - Is it 7 or 9? $(4 + 10) / 2$ or $4 + (10 / 2)$
- Each operator is given a precedence:
 - Higher precedence operators are applied first.
 - $/$ is higher than $+$, so the answer is..
 - Add brackets to force an ordering.
- Associativity:
 - Apply the operators from right-to-left, or left-to-right?
 - $+$, $-$ are left to right: do the one on the..
 - $=$, $+=$ are right to left: do the one on the..

Operator precedence

- Operators at same..

evaluated based on
associativity.

- * and / from L to R
- = and += from R to L

- Examples:

- result = -20 + 9 / 5;
- result = (-20 + 9) / 5;
- val = 6 + 5 * 4 / 3 * 2;
- sum = sum + 10;

Prec. Level	Op.	Operation	Associates
1	[]	Array Index	L to R
2	+ -	unary plus unary minus	R to L
3	* / %	mult, div, remainder	L to R
4	+ -	add subtract	L to R
5	<< >>	stream ins. extract.	L to R
6	< <= > >=	comparisons	L to R
7	= += -= *= ...	assignments	R to L

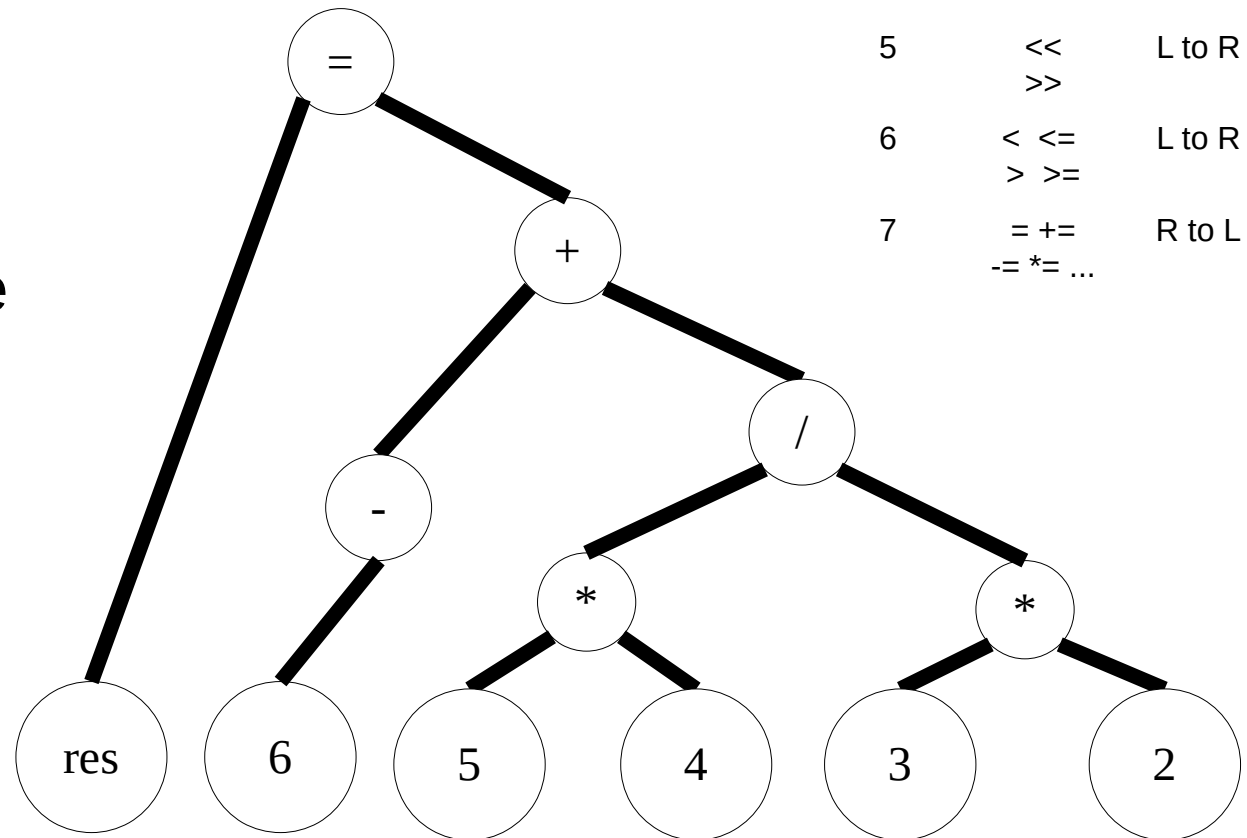
Order can be forced by parentheses.
See text Appendix 2 for full table.

Brackets

- A statement can be correct, but unreadable:
 - $\text{result} = 1 + 2 / 6 - 1 * 3 / 4 - 3 - -3 * +4;$
- Add brackets to make it clear:
 - $\text{result} = 1 + (2 / 6) - (1 * 3 / 4) - 3 - (-3 * (+4));$

Expression tree

- Represent $\text{res} = -6 + 5 * 4 / (3 * 2)$ as a tree:
- Operands as leaves.
- Operators as branching nodes.
- Operations lower in the tree have..
- Evaluate from the..
- Write values on internal nodes.



Prec. Level	Op.	Asso.
1	[]	L to R
2	unary + unary -	R to L
3	* / %	L to R
4	+ -	L to R
5	<< >>	L to R
6	< <= > >=	L to R
7	= += -= *= ...	R to L

Review

- Draw an expression tree for the following:

answer = $5 * x + 6 * (1 - x);$

Assume $x=2$

Prec. Level	Op.	Asso.
1	[]	L to R
2	unary + unary -	R to L
3	* / %	L to R
4	+ -	L to R
5	<< >>	L to R
6	< <= > >=	L to R
7	= += -= *= ...	R to L

Constants

Constants

- We have already used literal constants:
 `int x = 10;                               // Numeric constant`
 `println("Hello world!");               // String literal`
- Raw number in code are..
 `int w = d / 7;`
 `int c = s / 72;`
- Prefer named constants over magic numbers:
 `const int MIN_PER_HOUR = 60;`
 `int h = m / MIN_PER_HOUR;`

const

- const qualifier makes a variable..
 const double PAY_PER_DAY = 475.50;
 const int DAYS_PER_WEEK {7};
 - Constants must be given a value when created.
 - Name is upper case by..
 - Program cannot modify value of a constant:
 PAY_PER_DAY = 99.9; // ERROR!
- Advantages:
 - Program becomes more...
 - Can change value in entire program in one spot.
 - Ex: change tax rate that's used in 100 calculations!

Example with const

```
// Convert days to weeks/years/fortnights.
#include <iostream>
#include <print>
using namespace std;

const int DAYS_PER_WEEK = 7;
const int DAYS_PER_YEAR = 365;

int main()
{
    const int DAYS_PER_FORTNIGHT = 14;

    println("Enter the number of days: ");
    int numDays = 0;
    cin >> numDays;

    int numWeeks = numDays / DAYS_PER_WEEK;
    int numYears = numDays / DAYS_PER_YEAR;
    int numFortnight = numDays / DAYS_PER_FORTNIGHT;

    println("# Days:           {:4}", numDays);
    println("# Weeks:           {:4}", numWeeks);
    println("# Years:           {:4}", numYears);
    println("# Fortnights:      {:4}", numFortnight);
}
```

Constants can be..

```
Enter the number of days:
4641
# Days:           4641
# Weeks:           663
# Years:           12
# Fortnights:      331
```

Guide to Constants

- Which of the following literal constants would be best made into named constants?
 - `int numStudents = 0;`
 - `int next = numStudents + 1;`
 - `int waitlist = numStudents – 72;`

Combined Assignments & Overflow

Assignment Operators

- Combine an operation with assignment:

- +=, -=, *=, /=, %/=

- Examples:

- a += b;
// means a = a + b;
 - a *= b;
// means a = a * b;
 - a /= 2 + 3;
// means...

```
// Sum of numbers from 0 to MAX_COUNT
#include <print>
using namespace std;

const int MAX_COUNT {10};
int main()
{
    // Initialize the accumulator
    int sum {0};

    // Loop through 0 to MAX_COUNT - 1
    int i {0};
    while (i < MAX_COUNT) {
        sum += i;
        i++;
    }
    println("Sum from 0 to {} = {}",
        MAX_COUNT - 1, sum);
}
```

Sum from 0 to 9 = 45

Overflow & Underflow

- Each type has a maximum value it can store.

```
// Work with overflow/underflow
```

```
#include <print>
```

```
#include <climits>
```

```
using namespace std;
```

```
int main() {
```

```
    int test = INT_MAX;
```

```
    println("Start at:          {:12}", test);
```

```
    test += 1;
```

```
    println("Adding 1:          {:12}", test);
```

```
    test -= 1;
```

```
    println("Subtracting 1:    {:12}", test);
```

```
}
```

#include <climits>..

..

INT_MIN, INT_MAX,
CHAR_MIN, CHAR_MAX,
....

```
Start at:          2147483647
Adding 1:          -2147483648
Subtracting 1:    2147483647
```

- Maximum + 1 overflows to the minimum.
- Minimum - 1 underflows to the maximum.

Suggested Review Questions

- Draw an expression tree for the following:
result = 8 * - 1 + 3 / 2
 - Solve it by writing values on the nodes.
 - Write a C++ program to double check your answer.

Summary

- Expressions calculate values using operators.
 - Operator precedence gives us expression trees.
- Use named constants (const), not magic numbers.
- Combined assignment operators like `x += 2;`