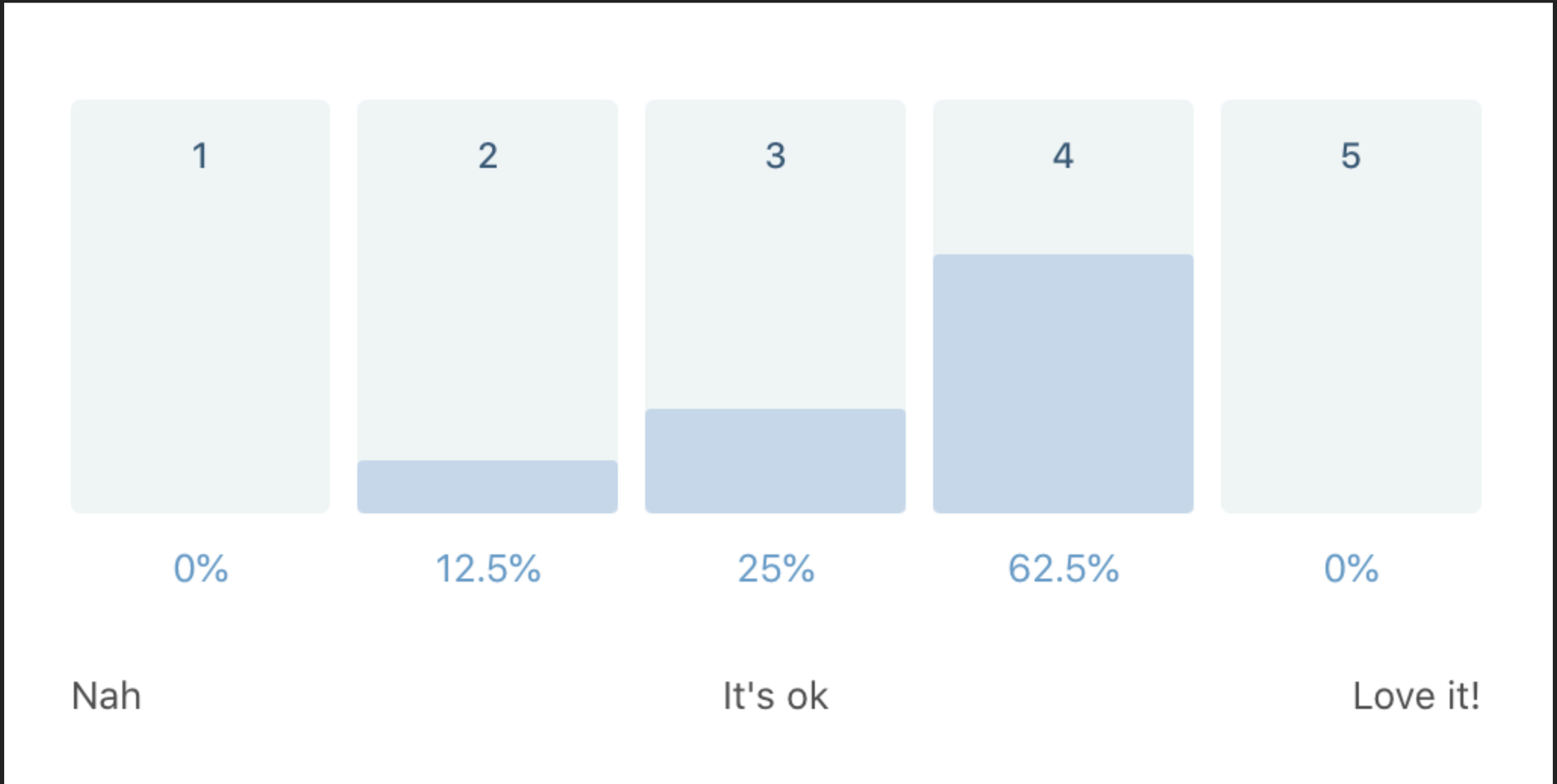
SEPTEMBER 16TH 2020

ELEMENTARY PROGRAMMING

SOME COVID BEST PRACTICES BEFORE WE START

- If you fill ill, go home
- Keep your distance to others
- Wash or sanitise your hands
- Disinfect table and chair
- Respect guidelines and restrictions

FEEDBACK CHECK



Average 3.5

FEEDBACK IN WORDS

What was good 

“Can understand you better” 

“More coding the last time”

“Where can I subscribe?”

To improve 

 “Give us more C!"

DTU wifi is really bad 

NEW FEEDBACK

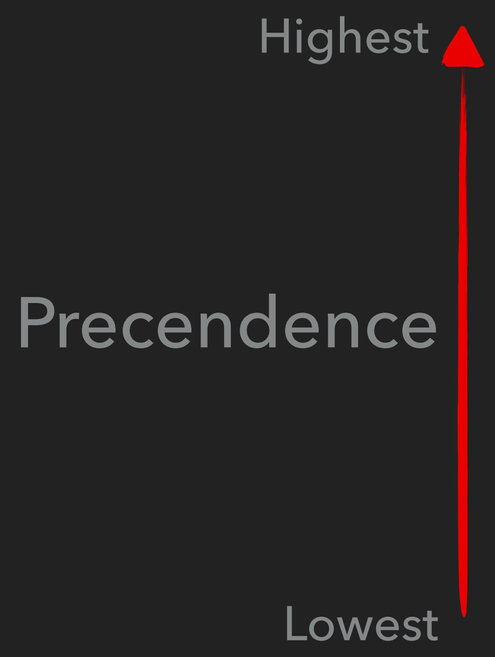
- I would really like for you to take a survey at the end of the session
- Feedback is important, please take the time to do it
- Pretty please <3
- Type this in your browser http://bit.ly/elemprog3

SOME INFO ABOUT THE COURSE - GRADE

- Two assignments during the class
- They will count 20% each for the final grade
- After each deadline we will have an online meeting where you explain the code to me
- One final test that will account for 60%

http://bit.ly/elemprogteams

OPERATOR PRECEDENCE



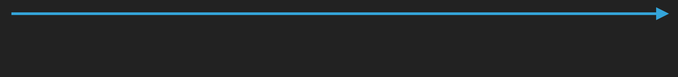
Unary operators + and -

* / %

\+ -

OPERATOR PRECEDENCE

```
i + j * k
```



```
i + (j * k)
```

```
-i * -j
```

```
(-i) * (-j)
```

```
+j + j / k
```

```
(+j) + (j / k)
```

ASSIGNMENT OPERATORS

```
i = i + 1;
```

```
i+=1;
```

```
/=   %=   +=   -=   *=
```

The book tries to warn you about some legal way write expression in C

You can of course check them out, but the book fail to mention they are a bad example of code. Never write your code in that convoluted way.

BOOLEAN TYPE

Boolean, or boolean logic, is a subset of algebra used for creating **true**/**false** statements. Boolean expressions use the operators **AND (&&)**, **OR(||)**, and **NOT(!)** to compare values and return a true or false result. These boolean operators are described in the following four examples:

X	Y	X && Y
0	0	0
1	0	0
0	1	0
1	1	1

X	Y	X || Y
0	0	0
1	0	1
0	1	1
1	1	1

X	!X
0	1
1	0

BOOLEAN TYPE IN C

There is no boolean type in C, there is only boolean logic.

You can do something like this:

```c
#include <stdio.h>

#define TRUE 1
#define FALSE 0

int main(void) {
  if(TRUE) {
    printf("true\n");
  } else {
    printf("false\n");
  }
}
```

OTHER OPERATORS

- Relational operators: `<` `>` `>=` `<=`
- Equality operators: `==` `!=`
- Operator Precedence:

```
i < j == j < k
```

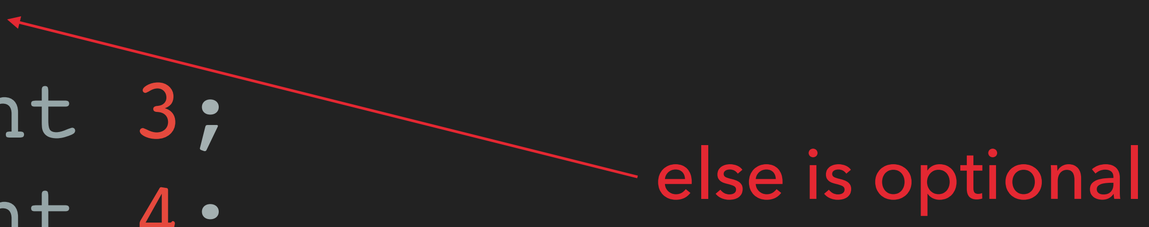


```
(i<j)==(j<k)
```

Relational wins over equality operators

IF STATEMENT

```
1    if ( boolean expression ) {
2     statement 1;
3     statement 2;
4    } else {
5     statement 3;
6     statement 4;
7    }
```



IF STATEMENT

```
1    if ( boolean expression ) {
2      statement 1;
3      statement 2;
4    } else if(other boolean expression) {
5      statement 3;
6      statement 4;
7    } else {
8      statement 5;
9      statement 6;
10   }
```

CONDITIONAL ASSIGNMENT AS TERNARY OPERATOR

```
1    int x;
2    scanf("%d", &x);
3    int result;
4    if ( x < 1 ) {
5      result = 2;
6    } else {
7      result = 0;
8    }
```

```
1    int x;
2    scanf("%d", &x);
3    int result = x < 1 ? 2 : 0;
```

SWITCH

```
1   switch ( expression ) {
2     case constant-expression-1: {
3       statements-1;
4       break;
5     }
6     ...
7     case constant-expression-2: {
8       statements-2;
9       break;
10    }
11    default: {
12      statements-1;
13    }
14  }
```

Constant expression: means a **char**, or a **int** or something like **1 + 2**

LOOPS - WHILE

```
1 while( expression ) {
2   statements;
3 }
```

LOOPS - DO...WHILE

```
1 do {
2   statements;
3 } while ( expression );
```

LOOPS - THE FOR STATEMENT

```
1 for ( expr1; expr2; expr3 ) {
2  statements;
3 }
```

LOOPS - EXAMPLES

```
for (i = 0; i < n; i++) {}
```
→ counting from 0 to n-1

```
for (i = 1; i <= n; i++) {}
```
→ counting from 1 to n

```
for (i = n-1; i >= 0; i--) {}
```
→ counting from n-1 to 0

```
for (i = n; i > 0; i--) {}
```
→ counting from n to 1

LOOPS - EXAMPLES

```
1 for(sum=0, i=1; i<=N; i++) {
2       sum += i;
3 }
```

THE BREAK AND CONTINUE STATEMENT

```
1  int i;
2  int j = 0;
3  for(i=0;i<n;i++) {
4   if (i % 0 == 0) {
5     continue;
6   }
7   if ( i == n-1) {
8     break;
9   }
10  j=j+i;
11 }
12 printf("%d",j)
```

EXERCISE #1

Calculating the Number of Digits in an Integer

Although the `while` statement appears in C programs much more often than the `do` statement, the latter is handy for loops that must execute at least once. To illustrate this point, let's write a program that calculates the number of digits in an integer entered by the user:

```
Enter a nonnegative integer: 60
The number has 2 digit(s).
```

Our strategy will be to divide the user's input by 10 repeatedly until it becomes 0; the number of divisions performed is the number of digits. Clearly we'll need some kind of loop, since we don't know how many divisions it will take to reach 0. But should we use a `while` statement or a `do` statement? The `do` state-

EXERCISE #2

In everyday programming, we'll often need to compare an expression against a series of values to see which one it currently matches. We saw in Section 5.2 that a cascaded `if` statement can be used for this purpose. For example, the following cascaded `if` statement prints the English word that corresponds to a numerical grade:

```
if (grade == 4)
  printf("Excellent");
else if (grade == 3)
  printf("Good");
else if (grade == 2)
  printf("Average");
else if (grade == 1)
  printf("Poor");
else if (grade == 0)
  printf("Failing");
else
  printf("Illegal grade");
```

EXERCISE #3

Calculating a Broker's Commission

When stocks are sold or purchased through a broker, the broker's commission is often computed using a sliding scale that depends upon the value of the stocks traded. Let's say that a broker charges the amounts shown in the following table:

Transaction size	*Commission rate*
Under $2,500	$30 + 1.7%
$2,500–$6,250	$56 + 0.66%
$6,250–$20,000	$76 + 0.34%
$20,000–$50,000	$100 + 0.22%
$50,000–$500,000	$155 + 0.11%
Over $500,000	$255 + 0.09%

The minimum charge is $39. Our next program asks the user to enter the amount of the trade, then displays the amount of the commission:

```
Enter value of trade: 30000
Commission: $166.00
```

NEW FEEDBACK

- I would really like for you to take a survey at the end of the session
- Feedback is important, please take the time to do it
- Pretty please <3
- Type this in your browser http://bit.ly/elemprog3

SOME COVID BEST PRACTICES BEFORE WE LEAVE

- Disinfect table and chair
- Maintain your distance to others
- Wash or sanitise your hands
- Respect guidelines and restrictions outside