Control Statements

C++ for Developers

Operators

When using control statements we use different operators:

Comparison Operators:

== is equal

!= is not equal

< is less than

> is greater than

<= is less than or equal

>= is greater than or equal

Logical Operators:

&& and

|| or

!

If / Else if / Else

Checks the control statements and performs the code based on their results.

```cpp
bool thisIsTrue = true;
bool thisIsFalse = false;

// IF THE 'IF' STATEMENT IS TRUE
if (thisIsTrue) {
    std::cout << "If is true" << std::endl;
} else if (thisIsFalse) {
    std::cout << "Else if is true" << std::endl;
} else {
    std::cout << "Neither was true" << std::endl;
}
```

```cpp
if (student1.age >= 18 && student1.grade == scholarschipGradeMinimum) {
    std::cout << student1.name + " has received a scholarschip!" << std::endl;
} else if (student2.age >= 18 && student2.grade == scholarschipGradeMinimum) {
    std::cout << student2.name + " has received a scholarschip!" << std::endl;
} else {
    std::cout << "No one received the scholarship!" << std::endl;
}
```

Switch

A switch statement in C++ lets you choose which block of code to run based on the value of a single variable.

The variable is compared against different constant values, called cases. If a match is found, the code inside that case runs. If no cases match, the default block runs instead.

Switch

```cpp
enum DaysOfTheWeek {
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY
};

DaysOfTheWeek today = WEDNESDAY;
```

```cpp
switch(today) {
    case MONDAY:
        std::cout << "WOO IT'S MONDAY!! TIME TO LEARN" << std::endl;
        break;

    case TUESDAY:
        std::cout << "Aaw, it's Tuesday... No Carl time :'(" << std::endl;
        break;

    case WEDNESDAY:
        std::cout << "YAY WEDNESDAY!! CARL TIME!!" << std::endl;
        break;

    case THURSDAY:
        std::cout << "Aaaw, it's Thursday... So long until the next lesson!" << std::endl;
        break;

    case FRIDAY:
        std::cout << "Friday, time to prepare for the weekend!" << std::endl;
        break;

    case SATURDAY:
        std::cout << "Woho Saturday! I can code all day!" << std::endl;
        break;

    case SUNDAY:
        std::cout << "Yay! Tomorrow it's lesson again!" << std::endl;
        break;

    default:
        std::cout << "Impressive, you have created a new day of the week!" << std::endl;
        break;
}
```

Exercises

Do exercises:

- #1. Compare a Number with 10
- #2. Got milk?

Loops

C++ for Developers

While-loop

Only loops while the expression is ‘true’. This also means, if the expression is not true before the loop, the loop will not run at all.

```
int loopMax = 5;
int loopCounter = 0;

while(loopCounter++ < loopMax) {
    std::cout << "While loop:" << loopCounter << std::endl;
}
```

lafftale for developers

Do While-loop

Will run the loop at least once before checking if the statement is true.

```
int doWhileMax = 0;
loopCounter = 0;

do {
    std::cout << "Do-While loop:" << loopCounter << std::endl;
} while (loopCounter++ < doWhileMax);
```

For-loop

Creates an index we can use to loop through containers. Loops as long as the control statement returns ‘true’. Works both with decrementing and incrementing operators.

```
int ages[] = {25, 32, 29, 19, 43};

for(int i = 0; i < sizeof(ages) / sizeof(ages[0]); i++) {
    std::cout << "For-loop: " << ages[i] <<  std::endl;
}
```

Enhanced For-loop

Loops through the elements of a container without an index. Pairs well with the ‘auto’ keyword.

```
for(int age : ages) {
    std::cout << "Enhanced with int: " << age << std::endl;
}

for(auto age : ages) {
    std::cout << "Enhanced with auto: " << age << std::endl;
}
```

Looping keywords

You can use different keywords to manipulate the loop behaviour:

- break; will break the loop (if not inside an switch case)
- continue; will go directly to the top of the loop

Exercises

Do exercises:

- #1. Print numbers 0 to 10
- #2. Numbers between two values
- #3. Repeat until user quits


lafftale
for developers