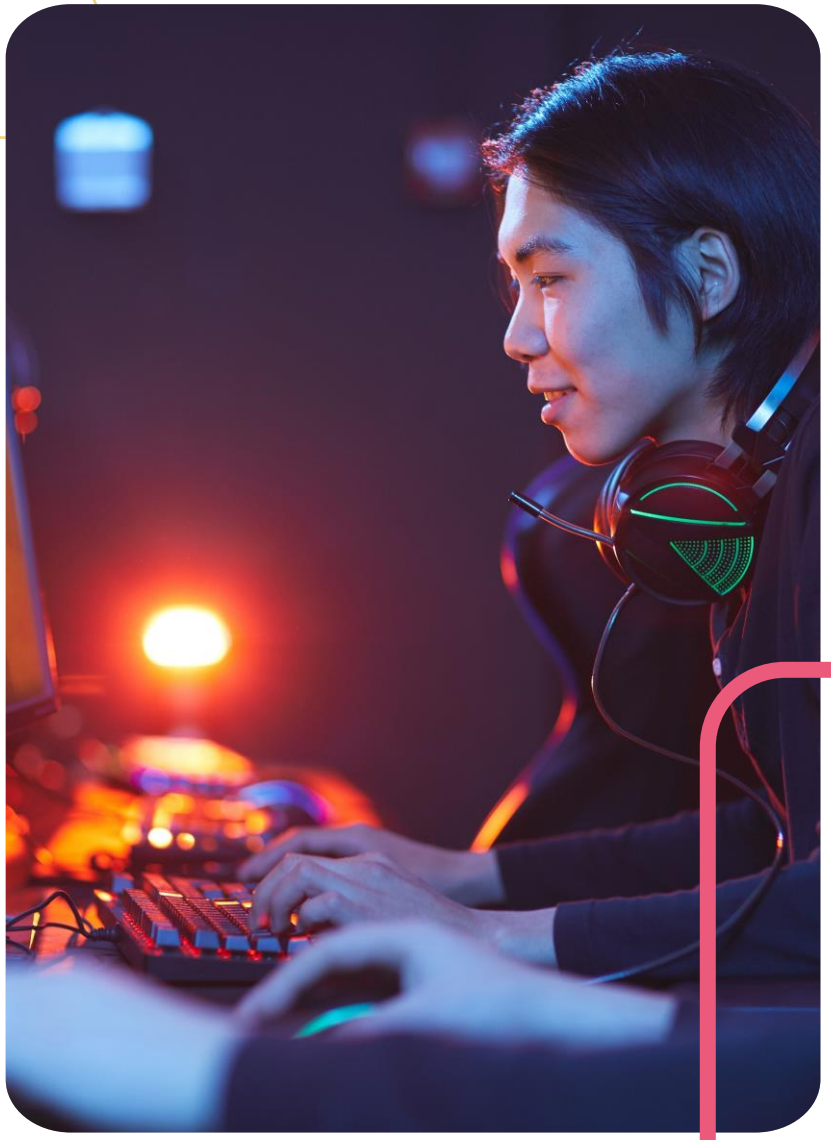
Diploma in

Computer Science

Decision Making



Objectives

- Understand decision making in C
- Learn how to create a repetitive block of code
- Appreciate the different ways of achieving repetition
- Explore switch and goto statements

DID YOU KNOW?

Decision making in computers involves ranking, prioritising, or choosing from among alternatives characterised on multiple criteria or attributes.

Smarter code

Uncertain actions in programs

Decision making sometimes includes choosing between a number of blocks of code to make your computer do the appropriate action

Critical functions such as error handling rely on the decision-making capabilities of programming languages

You need a flowchart!

Planning using complex control structures help you to understand the flow of your code and to avoid errors

Decisions in everyday life

Decisions in computing

A set of conditions which we evaluate

Choose the appropriate course of action

Computer evaluates a set of conditions in the code

Computer decides on the order of execution

Decisions in C programming

if statement

switch statement

conditional operator statement
(?: operator)

goto statement

Decisions in C

- At least one condition
- Two blocks of code
- First block of code executed if the condition is met
- Second block of code executed if condition is not met

Decisions

Operator	Meaning	Example
==	Strict equal to	a == 5
!=	Strict not equal to	a != 5
>	Greater than	a > 5
>=	Greater than or equal	a >= 5
<	Less than	a < 5
<=	Less than or equal	a <= 5

Decisions – logical operators

When one comparison is not enough for computer to execute code

Logical operators allow you to base the execution on multiple conditions

Decisions – multiple logical operators

To evaluate multiple conditions at once

Use multiple logical operators

Apply operator precedence rules

Simple conditions

if statements

Example

```
if(condition)
{
 Statement executed when condition is
met;
}
 statement executed when condition is
not met;
```

Example

```
if(a<5)
 printf("This number is too large");
```

if statements

Must be a Boolean condition

Condition can only be true or false

No ambiguity

Example

```
if(a<5)
 printf("This number is too large");
```

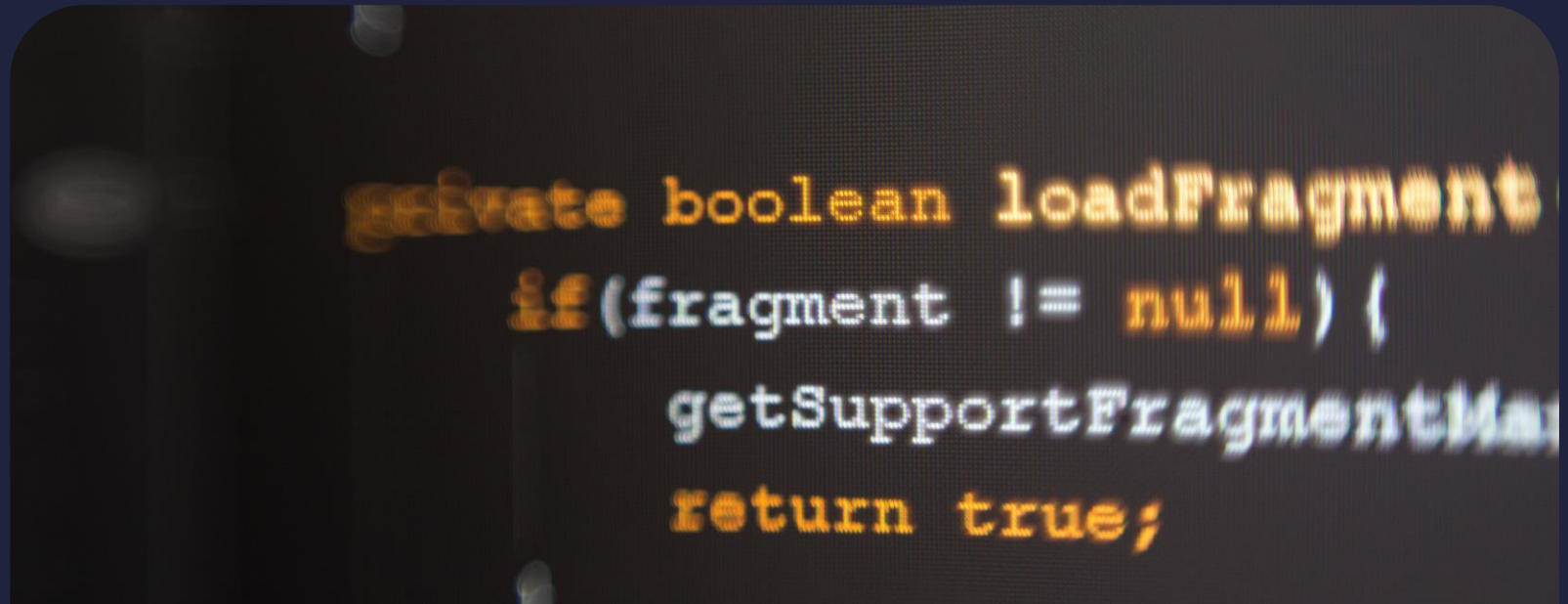
DID YOU KNOW?

Fuzzy logic is the concept of partial truth where the truth value may range between completely true and completely false.

if statements

How would you check if a=5?

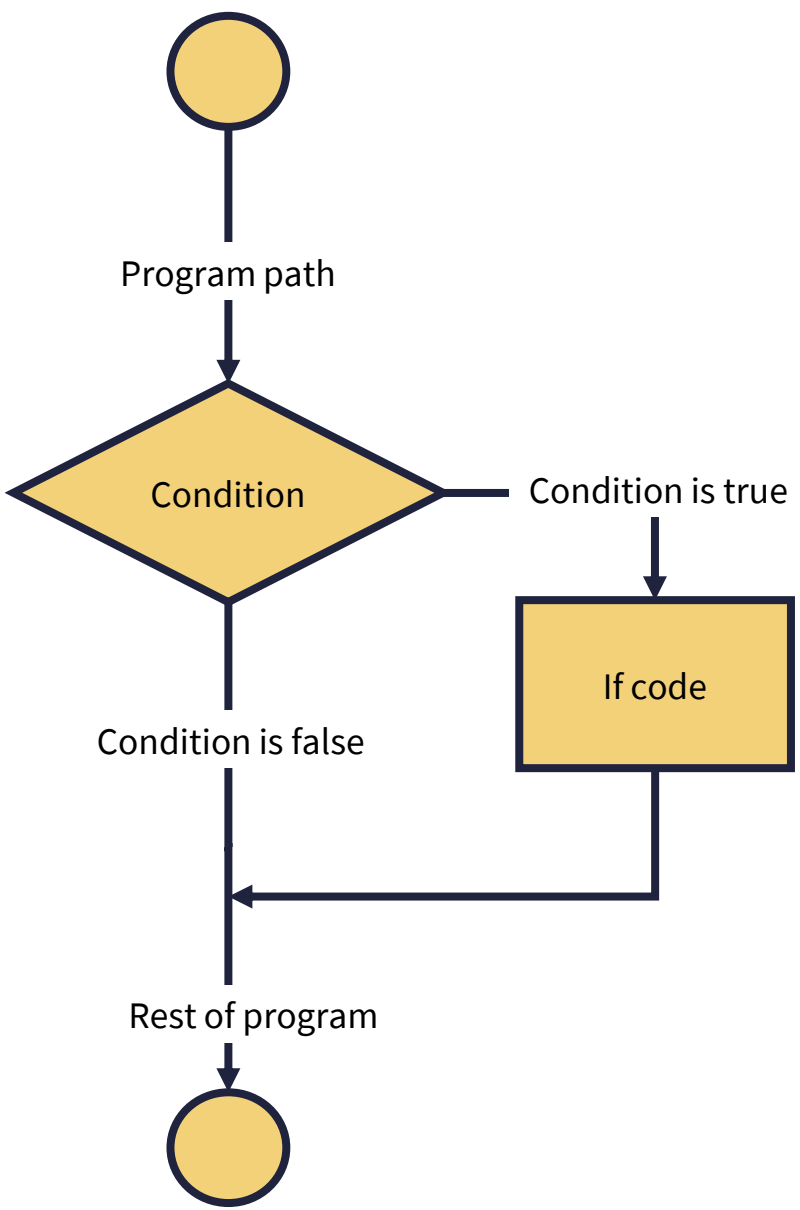
Use the comparison operator ==

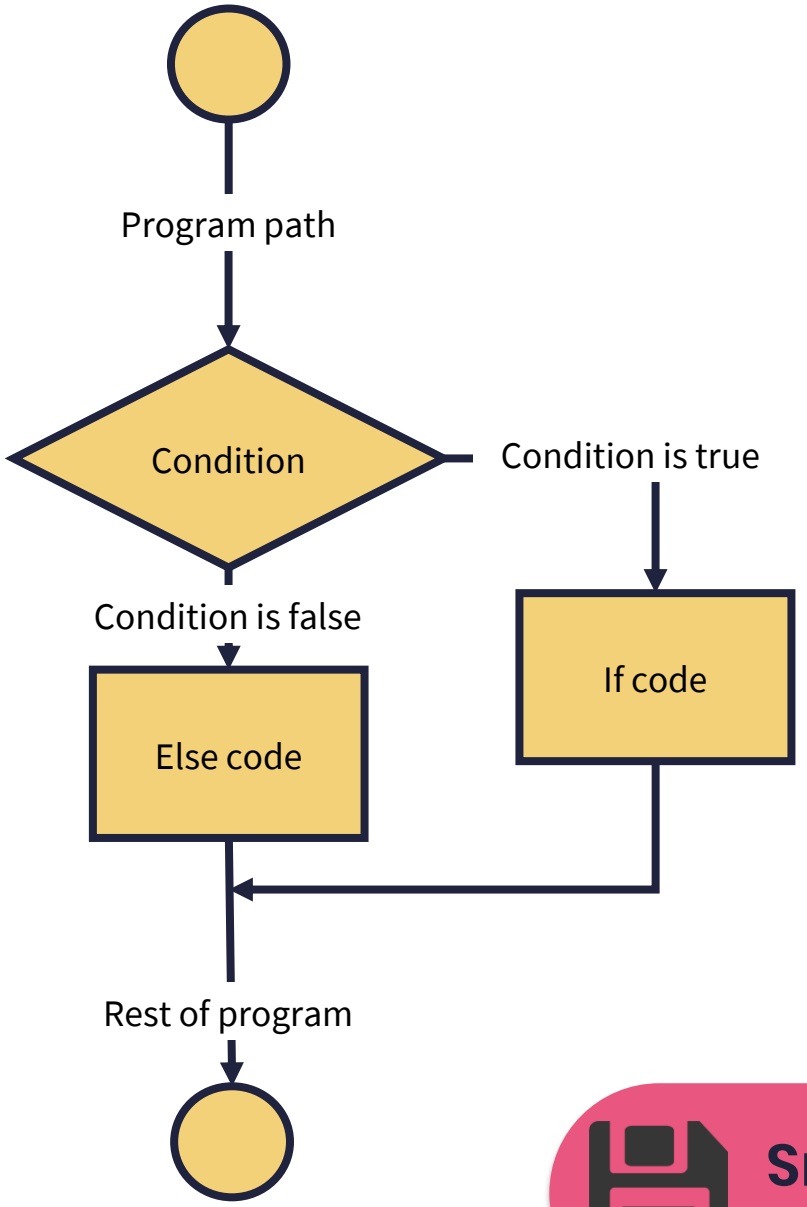


if-else statements

Expands the *if* statement

Removes ambiguity from code


Program path
Condition
Condition is true
If code
Condition is false
Rest of program


Program path
Condition
Condition is true
Condition is false
Else code
If code
Rest of program

Snap & save

else-if statements

Stack statements so there are a number of alternate statements to be executed

if statements keep going on and on use the case statement

A school uses a grading system like this:

A=80-100

B=70-79

C=60-69

D=50-59

E=40-49

F=0-40

Example

if statements

if statement can be followed by an optional else *if...else* statement - useful to test various conditions

if can have zero or one *elses* and it must come after an *else-if*

if can have zero or more *else...ifs* and they must come before the *else*

Once an *else...if* succeeds none of the remaining *else...ifs* or *elses* will be tested

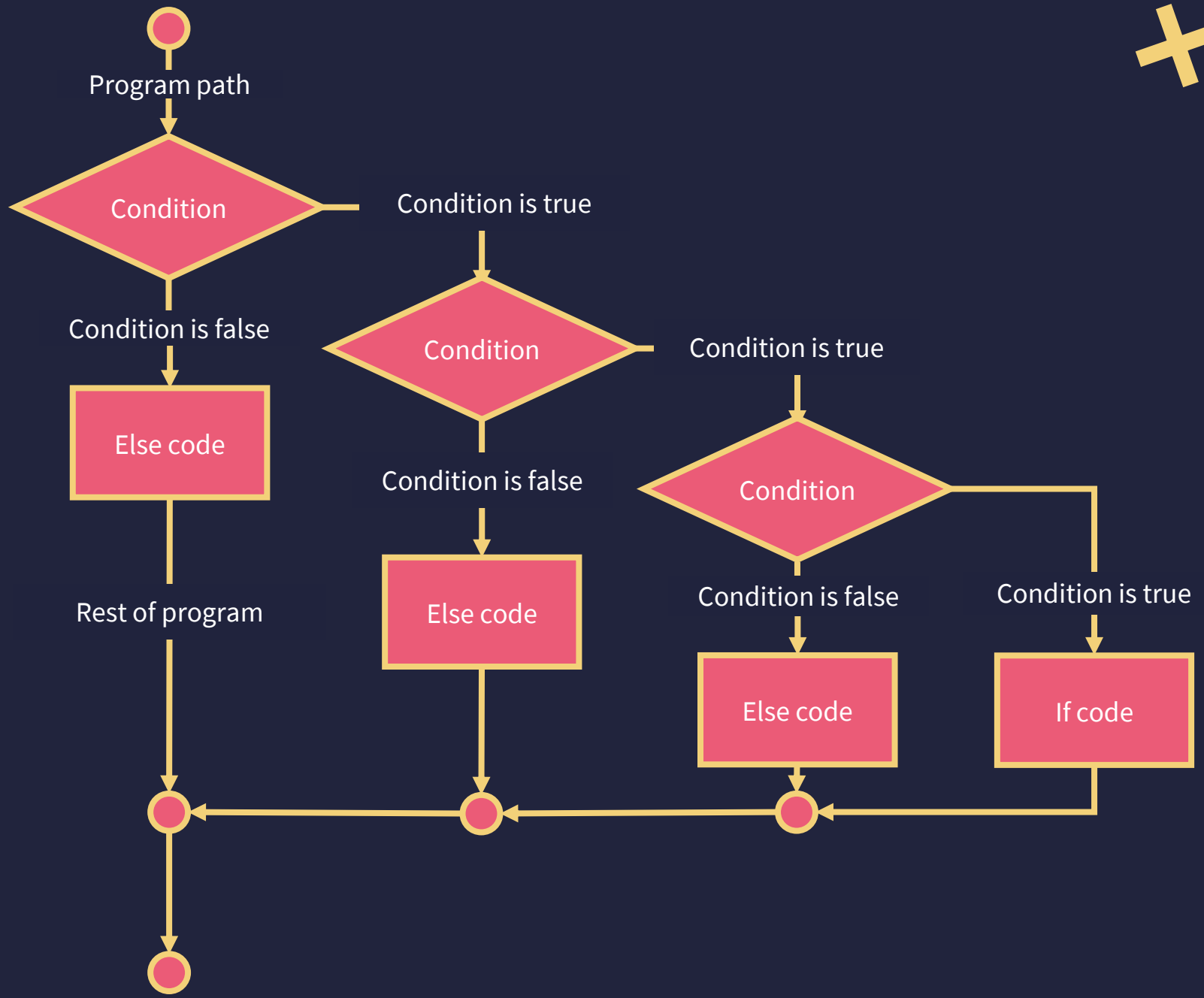
If you find yourself confused by an *if* statement, read it as though you were reading an actual English sentence.

Nested *if* statements

A statement within a statement

if statements exist one level lower at each stage and are evaluated backwards

Nested *if* statements



Switch statements

Switch statements, like trees, have lots of branches that each follow their own path.

Switch statements

Typically used with multiple options

Each block of code that can be executed is a case

Each case has a unique identifying constant

Guidelines for using switch statements

The expression can be a variable or an expression – result is either an integer or a character.

The case label must be a literal or a variable declared to be const.

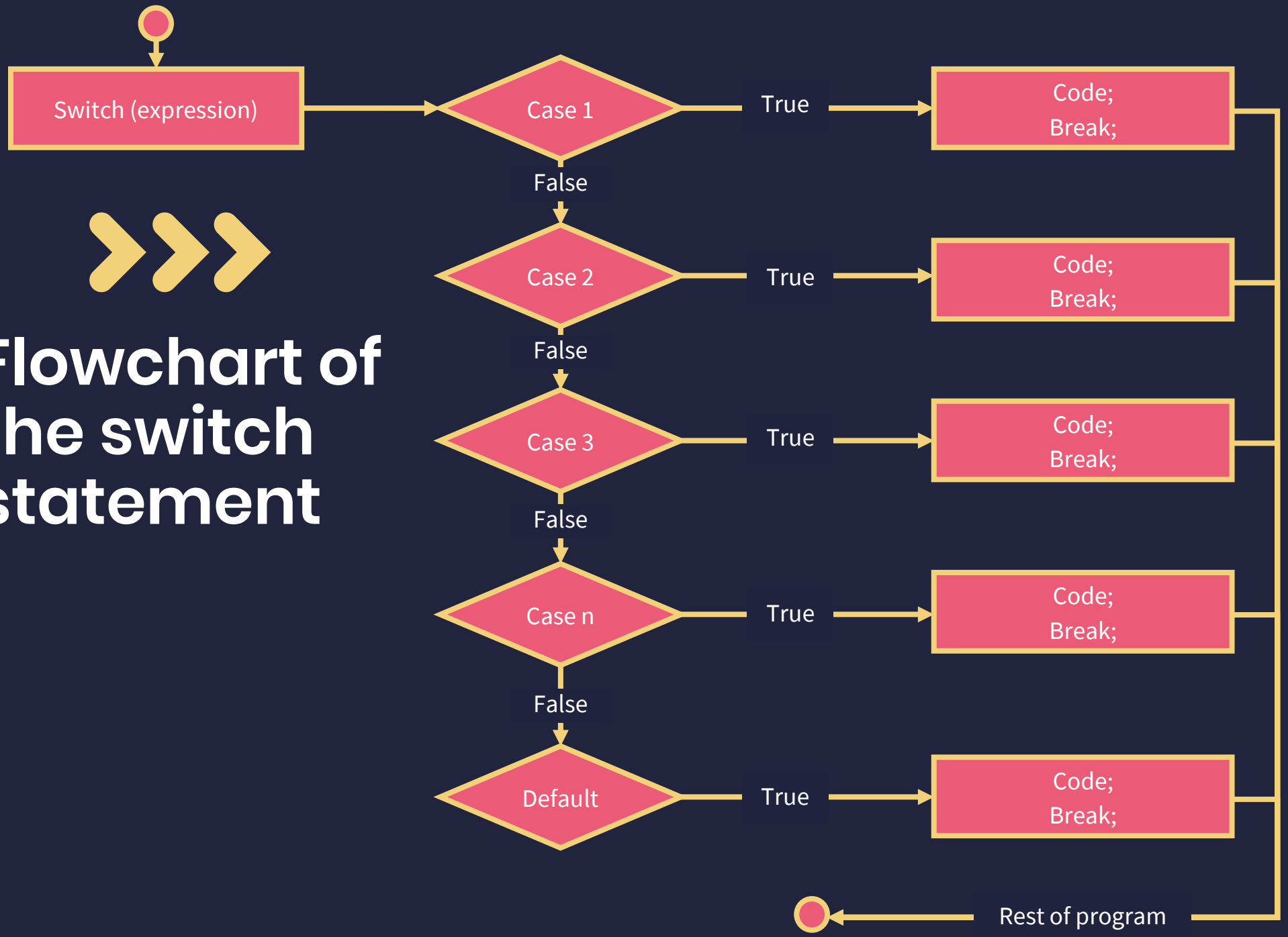
You can have as many cases as you like but no duplicates.

Guidelines for using switch statements

The optional default case is executed when none of the cases match the expression evaluation.

Use the break statement to break the flow of control when the computer encounters a matching case and executes the relevant code.

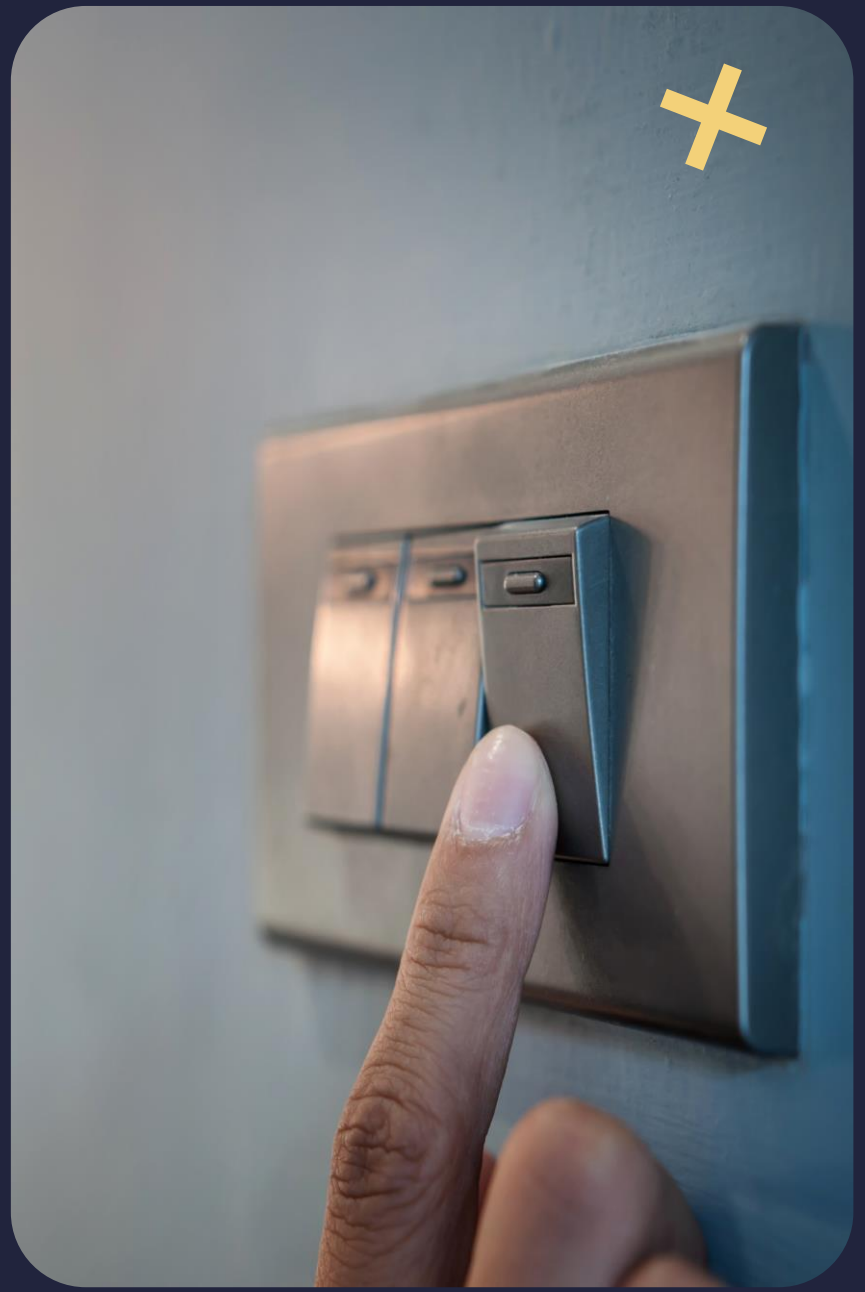
Flowchart of the switch statement



Syntax of the switch statement

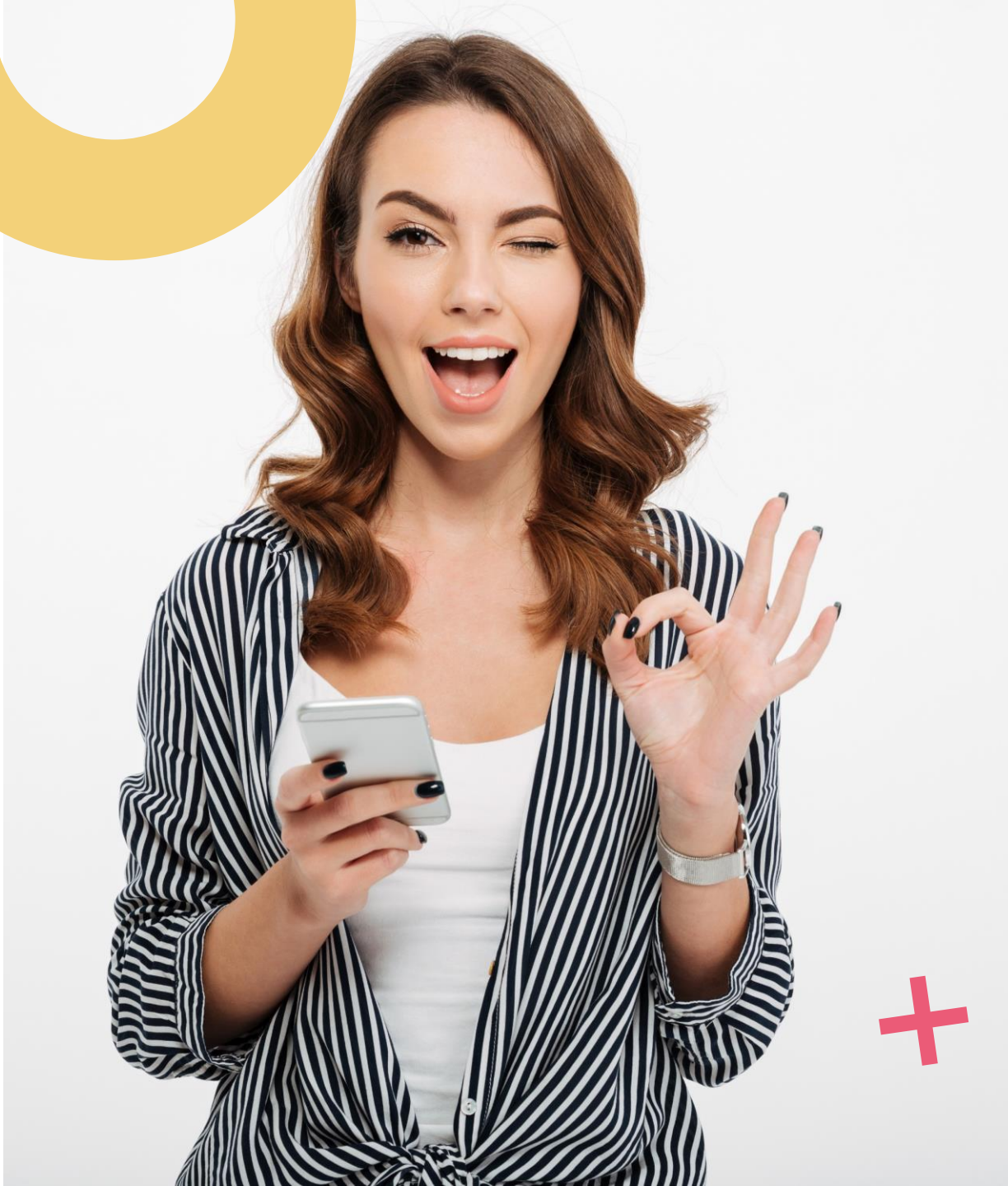
Example

```
switch(expression)
{
        case value-1:
                code;
                break;
        case value-2:
                code;
                Break;
        case value-n:
                code;
                break;
        default:
                code;
                break;
```



Have some ‘switch statement’ fun...

Recreate the USSD menu on your phone.



The magic in the USSD menu on your phone happens in the switch expression

It’s all about the number options and the fact that the switch statement accepts integers and characters

When you dial the USSD code, you are presented with options that look something like this

1. Self service
2. Check balance
3. Buy airtime
4. SIM swap
5. Port my number
6. Other services

Example

What expressions can you use in a switch structure?

Can be a mathematical or logical expression, a variable, or a constant.

Conditional operator

Can be used to create short expressions that work the same way as *if-else* statements

Ternary operator

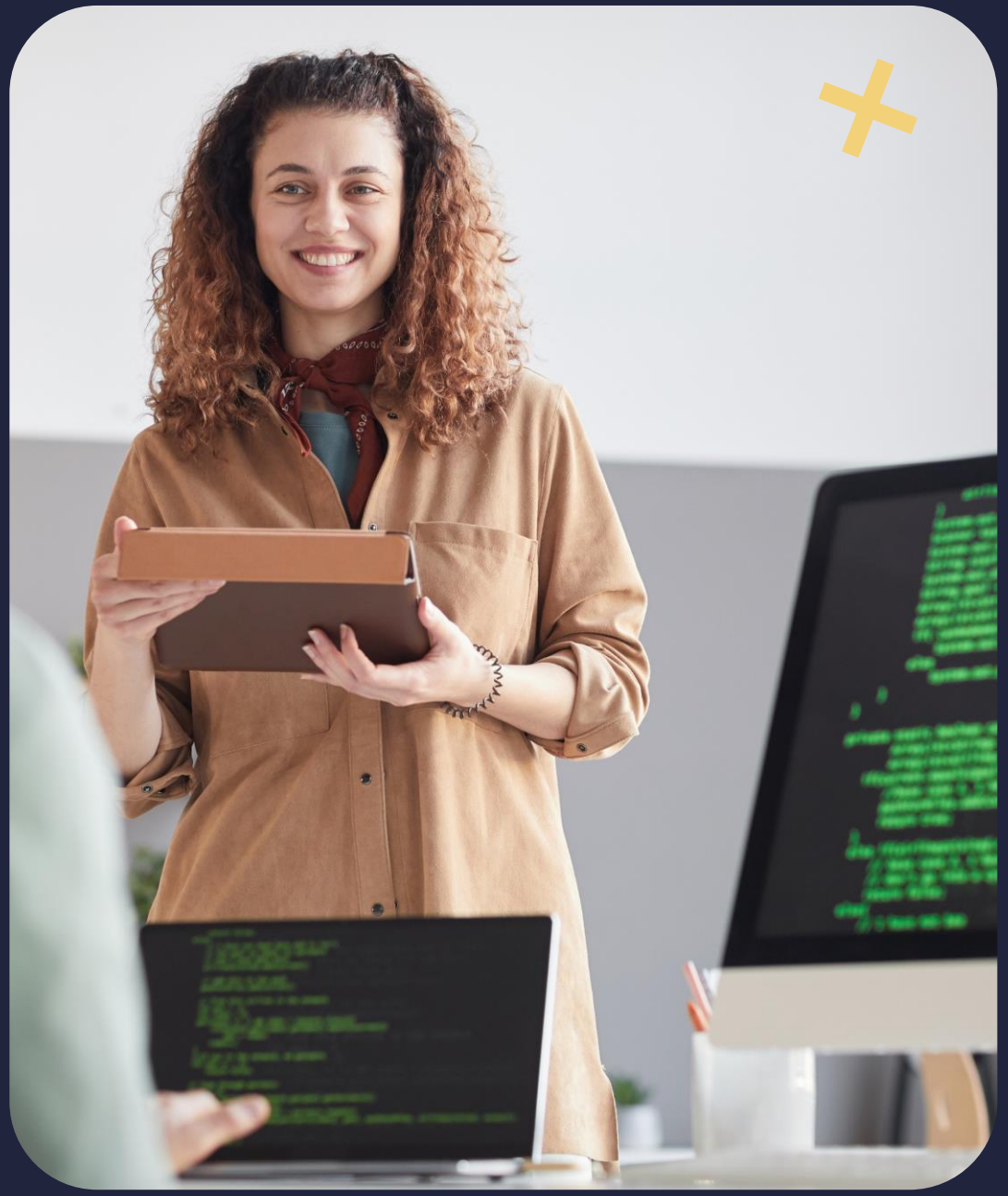
Takes three arguments:

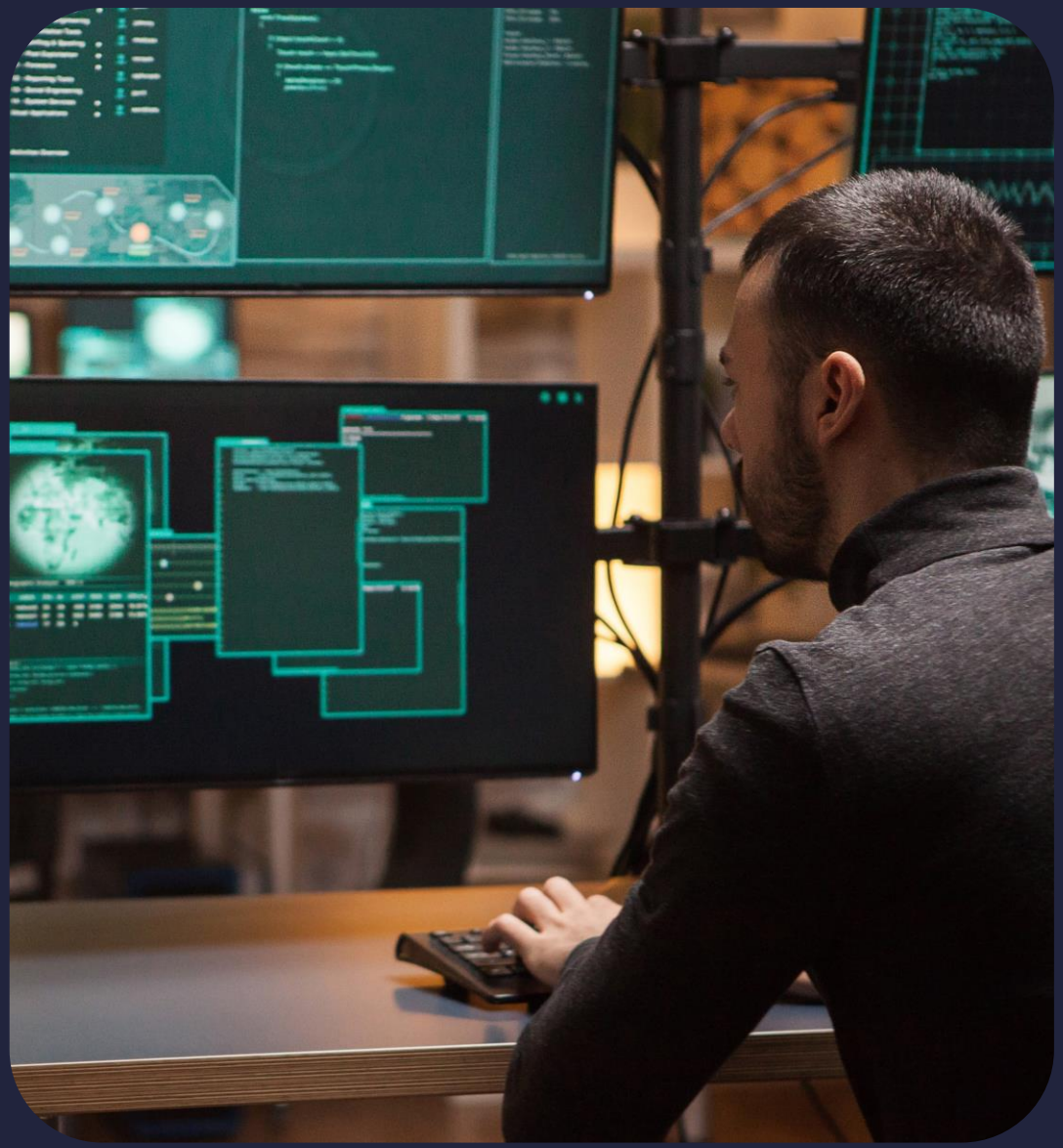
- A comparison argument
- The result adopted if comparison is true
- The value adopted if comparison is false

Example

```
int a = 4, b = 5, c = 0;
if (a < b) {
 c = a;
}
else {
 c = b;
}

printf("%d is smaller", c);
```





Example

```
int a = 4, b = 5, c = 0;
c = (a < b) ? a : b;
printf("%d is smaller", c);
```

Removed 5 lines of code by using the conditional operator

Arguments *value_if_true* and *value_if_false* must be the same type and simple expressions

The ternary operator can also be used in nested statements.

Example

```
int a = 4, b = 5, c=0;
if (a == 4) {
 if (b == 5) {
 c = 3;
 } else {
 c = 5;
 }
} else {
 c = 0;
}
printf ("%d\n", c);
```

Example

```
int a = 4, b = 5, c=0;
c = (a == 4 ? (b == 5 ? 3 : 5) : 0);
printf ("%d\n", c);
```

Use ternary operators to write programs faster.

The *if-else* statement vs the conditional operator

if-else statement	Ternary operator
Block of code	Single statement
Can’t assign a value to a variable	Can assign a value to a variable
Works best with multiple conditions	Not best choice for multiple conditions

Goto statements

goto statements

An unconditional jump to another section of the program

Can make it difficult to follow flow of program

Prioritise readability

Syntax of the *goto* statement

```
goto label;
..
.
label: statement;
}
```

Example

Used in conjunction with other statements

goto statement flowchart