Control Structures

Branches

- Simple branch
- Multiple branches

Loops

- For loop
- While loop

Simple branches:

```
if (condition) {
    block
}
```

```
if (condition) {
    block 1
} else {
    block 2
}
```

If with init statement

C++17

```
if (init statement; condition) {
    block
} // with or without else
```

Question: Will it print "Condition is true" ?

✔ ✗

In [ ]:
```
#include <iostream>
using namespace std;
auto a = true, b=false;

if (a) {
    cout << "Condition is true!" << endl;
}
```

Question: Will it print "Condition is true" ?

✔ ✗

In [ ]:
```
//auto a = true, b=false;
if (a && b) {
    cout << "Condition is true!" << endl;
} else {
    cout << "Condition is false!" << endl;
}
```

Question: Will it print "Condition is true" ?

✔ ✗

In [ ]:
```
// auto a = true, b=false;
if (a || b) {
    cout << "Condition is true!" << endl;
} else {
    cout << "Condition is false!" << endl;
}
```

Question: Will it print "Condition is true" ?

✔ ✗

In [ ]:
```
// C++ 17
// auto a = true, b=false;
if(a=false; (a == b) || (a && b)) {
    cout << "Condition is true!" << endl;
} else {
    cout << "Condition is false!" << endl;
}
```

Ternary operator:

This operator allows you to do an if-else statement in one line and needs to be within an expression or statement

`(condition) ? true-clause : false-clause`

example:

In [ ]:
```
#include <iostream>
using namespace std;

bool c = true, d = false;
cout << "Condition is " << ( (c||d)  ? "true!" : "false!") << endl;
```

Multiple branches:

```
if (cond 1) {
        block 1
} else if (cond 2) {
        block 2
} else if (cond 3) {
        ...
} else {
        block N
}
```

```
switch (cond) {
    case expr1:
        block
        break;
    case expr2:
        block 2
        break;
    ...
    default:
        block N
}
```

Question: Will it print "Elevetor error" ?

✔ ✗

In [ ]:
```
#include <iostream>
using namespace std;
int level = 2;

const int ground=0, first = 1, second=2;

if (level==ground) {
    cout << "Ground floor" << endl;
} else if (level == first) {
    cout << "First floor" << endl;
} else if (level == second) {
    cout << "Second floor" << endl;
} else {
    cout << "Elevator error!" << endl;
}
```

Question: Will it print "Second floor" ?

✔ ✗

In [ ]:
```
//int level = 2;
//const int ground=0, first = 1, second=2;

switch (level) {
    case ground:
        cout << "Ground floor" << endl;
        break;
    case first:
        cout << "First floor" << endl;
        break;
    case second:
        cout << "Second floor" << endl;
        break;
    default:
        cout << "Elevator error!" << endl;
        break;
}
```

For Loops

For loops are used to iterate a block of code.

```
for (init; until; next) {
   block
}
```

- init: counter initial condition
- until: counter ending condition
- next: counter update

Each of the previous three components can be omitted

Range based loops C++11

[Range based for loops](https://en.cppreference.com/w/cpp/language/for) iterate over a range.

```
for (elem:range) {
   block
}
```

An init statement is also available in C++20

```
for (init-statement; elem:range) {
   block
}
```

In [ ]:
```
#include <iostream>

int numbers[5]={0,1,2,3,4};
for (auto i=0; i<5; i++){
   if (numbers[i]%2==0)
      std::cout << numbers[i] << " is even!\n";
   else
      std::cout << numbers[i] << " is odd!\n";
}
```

In [ ]:
```
for (auto number: numbers) {
   if (number%2==0)
      std::cout << number << " is even!\n";
   else
      std::cout << number << " is odd!\n";
}
```

While loop

The while loop and do-while are other ways to iterate a block of code.

```
while (condition){
   block
}
```

```
do {
   block
} while (condition)
```

In [ ]:
```
//int numbers[5]={0,1,2,3,4};
int i=0;
while (i<5) {
   if (numbers[i]%2==0)
      std::cout << numbers[i] << " is even!\n";
   else
      std::cout << numbers[i] << " is odd!\n";
   i++;
}
```

for-while loops transformation

```
for (init; until; next) {
   block
}
```

```
init;
while (until){
   block
   next
}
```