



ESTRUTURAS DE REPETIÇÃO - WHILE e FOR

TÓPICO 3



@projetotouuberlandia



Universidade
Federal de
Uberlândia



PROEXC
diretoria de extensão



direc
Faculdade de
Direito



Faculdade de
Computação
Tecnológico da
Uberlândia



TOU
Tecnológico da
Uberlândia



UBERHUB
CODE CLUB



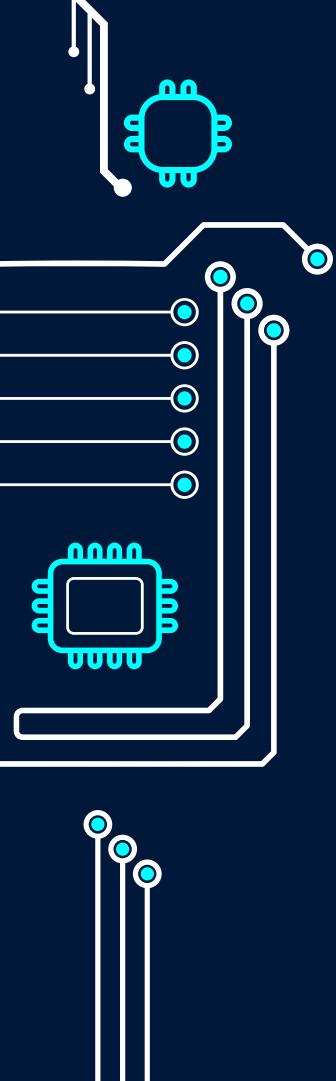
O QUE VAMOS APRENDER?

Neste tópico, vamos aprender sobre as estruturas de repetição, também conhecidas como loop ou laço.

São estruturas que permitem executarmos uma sequência de comandos várias vezes de acordo com uma condição ou contador.

Estruturas:

- WHILE
- FOR



WHILE

- Estrutura mais simples.
- Repete um conjunto de comandos enquanto uma condição permanecer verdadeira.
- Se a condição for falsa, os comandos dentro do while não serão executados e o fluxo do diagrama será desviado para o primeiro comando após o bloco while.





ESTRUTURA BÁSICA

```
1 | while(condição){  
2 |     //faça isso  
3 | }
```

Ou seja, enquanto a condição for verdadeira, faça o que estiver entre as chaves.



Universidade
Federal de
Uberlândia



PROEXC
diretoria de extensão



direc
diretoria de extensão



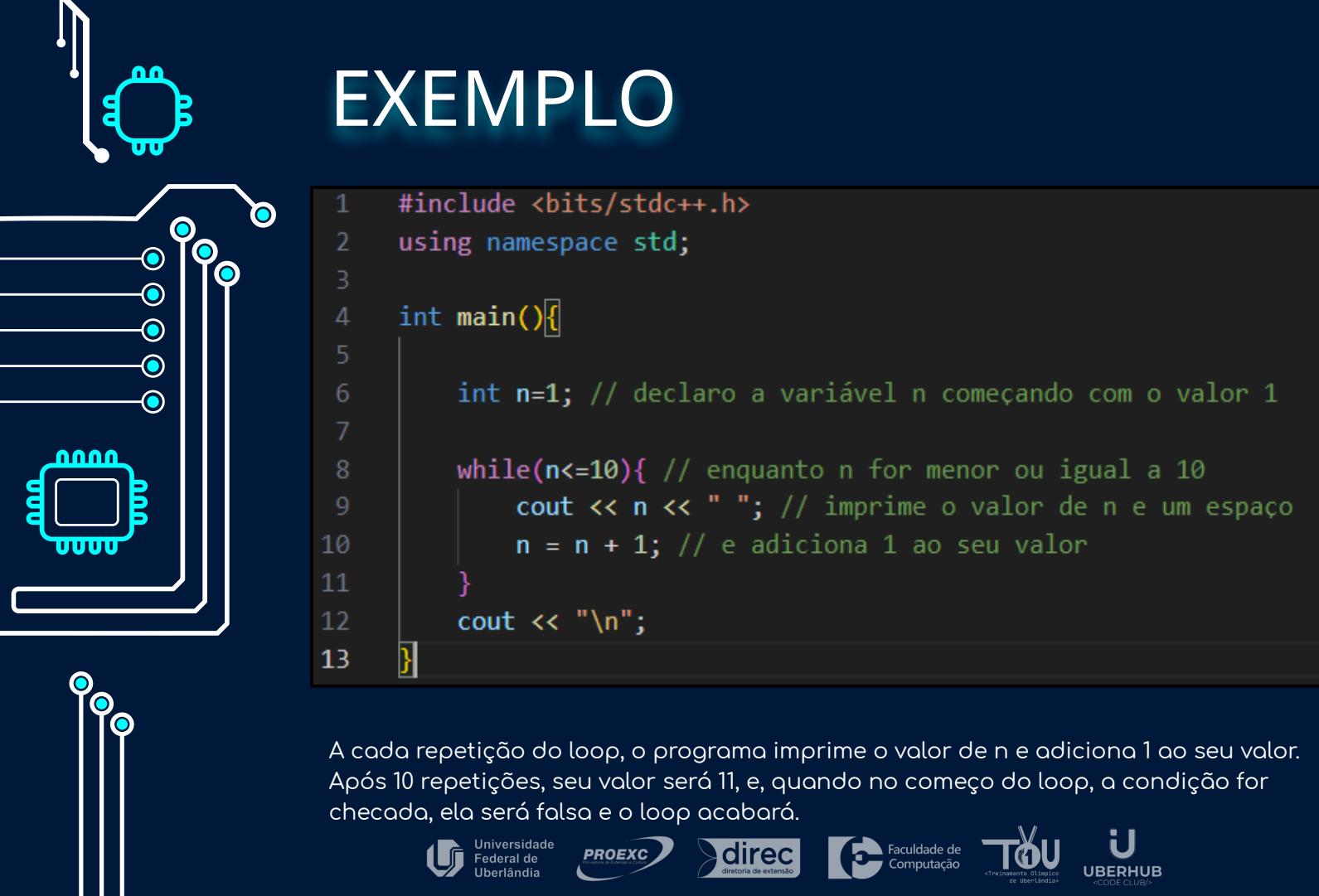
Faculdade de
Computação



TOU
Tecnológico
de Uberlândia



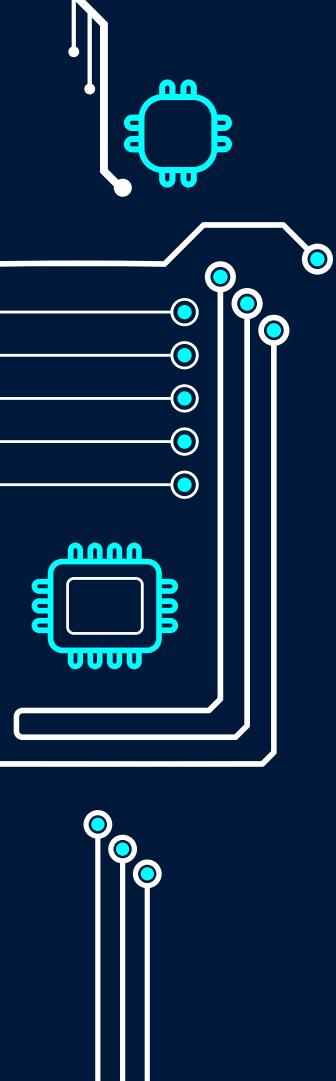
UBERHUB
CODE CLUB



EXEMPLO

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5
6     int n=1; // declaro a variável n começando com o valor 1
7
8     while(n<=10){ // enquanto n for menor ou igual a 10
9         cout << n << " "; // imprime o valor de n e um espaço
10        n = n + 1; // e adiciona 1 ao seu valor
11    }
12    cout << "\n";
13 }
```

A cada repetição do loop, o programa imprime o valor de n e adiciona 1 ao seu valor. Após 10 repetições, seu valor será 11, e, quando no começo do loop, a condição for checada, ela será falsa e o loop acabará.



FOR

- É separado em 3 partes distintas:
 - Inicialização - inicializa variáveis (contador);
 - Condição de parada - determina o fim da repetição;
 - Incremento - incrementa o valor do contador.
- Executa um conjunto de comandos até que a condição seja falsa.

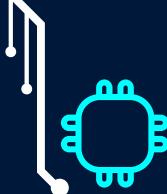


ESTRUTURA BÁSICA

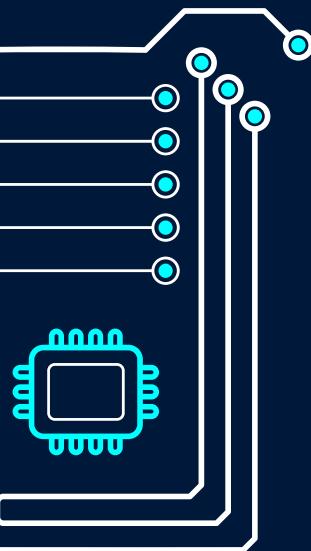
```
1 | for(inicialização ; condição_de_parada ; incremento){  
2 |     //comandos  
3 | }
```

Cada parte do for é separada por um ; (ponto e vírgula).



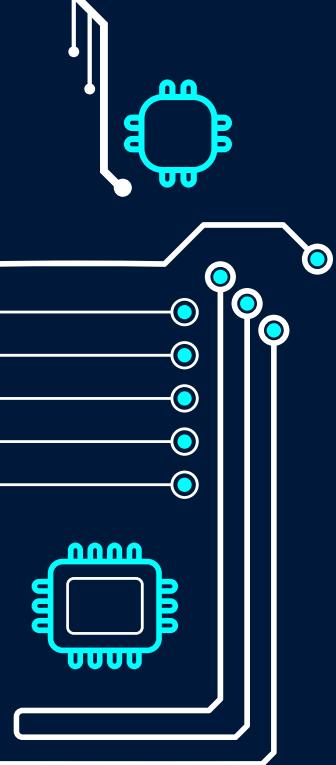


EXEMPLO



```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5
6     for(int i=1;i<100;i++){ // enquanto i for menor que 100
7         cout << i << " "; // imprime o valor de i e um espaço
8     }
9
10    cout << "\n";
11 }
```

Inicializa uma variável `i` com valor 1 e enquanto `i` for menor que 100, iremos imprimir o valor de `i` na tela. Após acabar todos os comandos dentro do `for`, a variável `i` é incrementada em uma unidade e passa novamente pela checagem da condição de parada. Caso a condição continue verdadeira (`i` sendo menor que 100), os comandos dentro do `for` executarão novamente. Mas assim que a condição `for` falsa (`i` sendo 100), o laço de repetição se encerrará.



LOOP INFINITO

Seja no for ou while, devemos SEMPRE estipular uma condição de parada para o nosso laço de repetição, se não, o código irá rodar infinitamente.

```
int n=1;  
  
while(n<=10){  
    cout << n << " ";  
}  
cout << "\n";
```

```
for(int i=1;i<=10;){  
    cout << i << " ";  
}
```

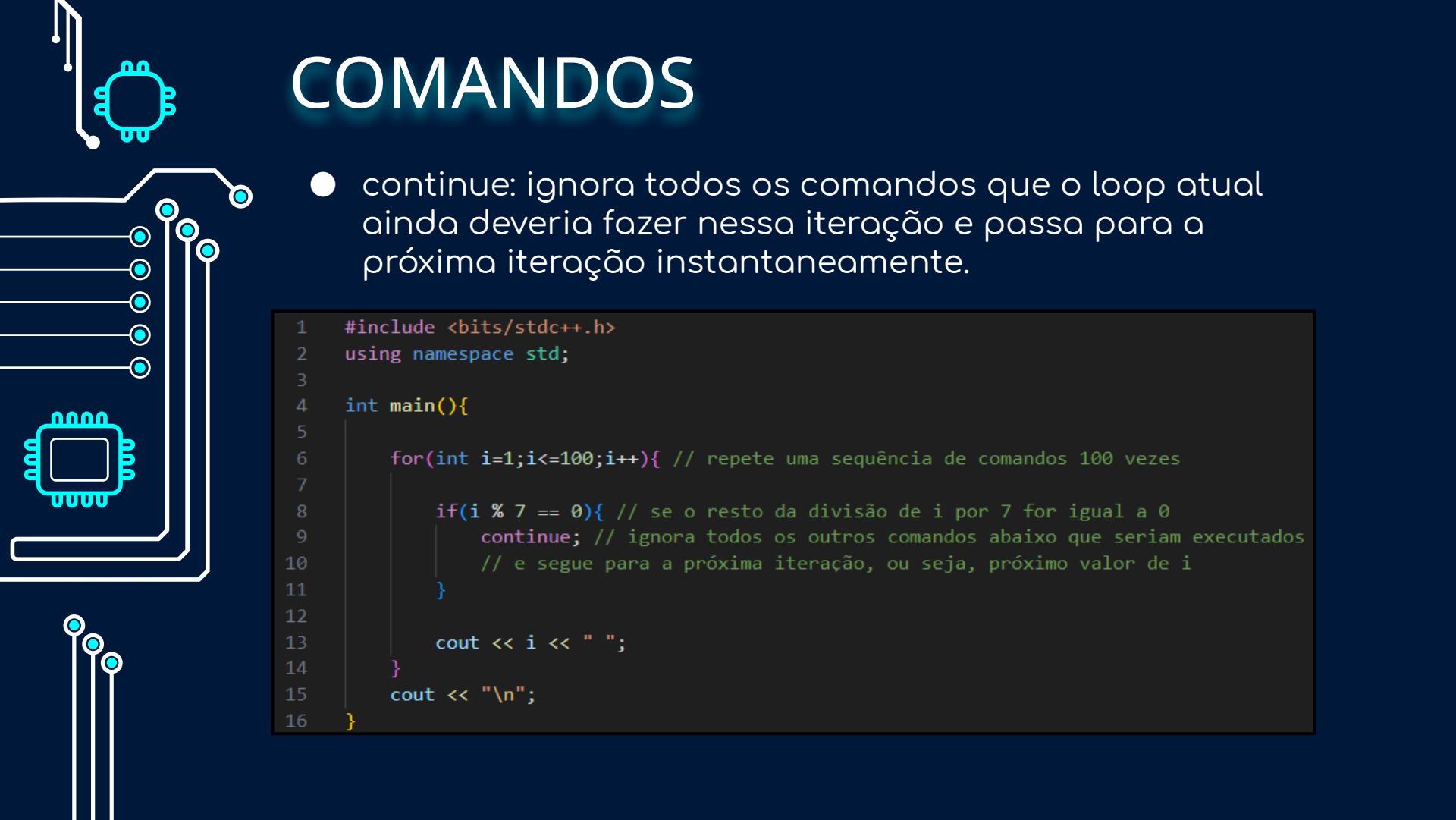
Aqui, em ambos os casos, não existe um contador, logo, será impresso 1 na tela infinitamente.



COMANDOS

- break: para o loop antes do critério de parada original.

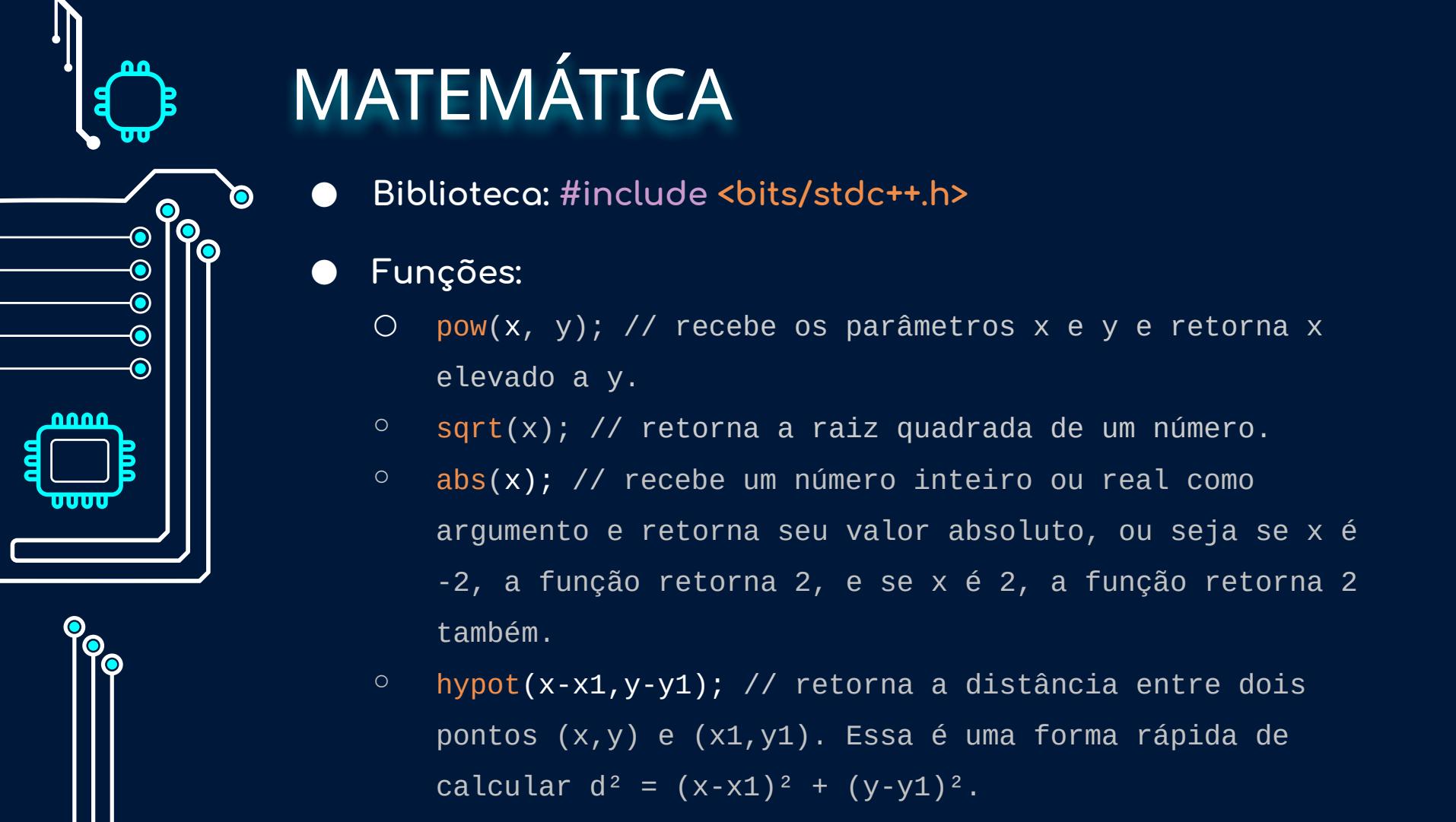
```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5
6     int var;
7
8     for(int i=1;i<=10;i++){ // repete uma sequência de comandos 10 vezes
9         cin >> var; // ler um valor que será armazenado em var
10
11        if(var == 0){ // se o valor for igual a 0
12            break; // encerre o for
13        }
14
15        cout << 2*var << "\n";
16    }
17 }
```



COMANDOS

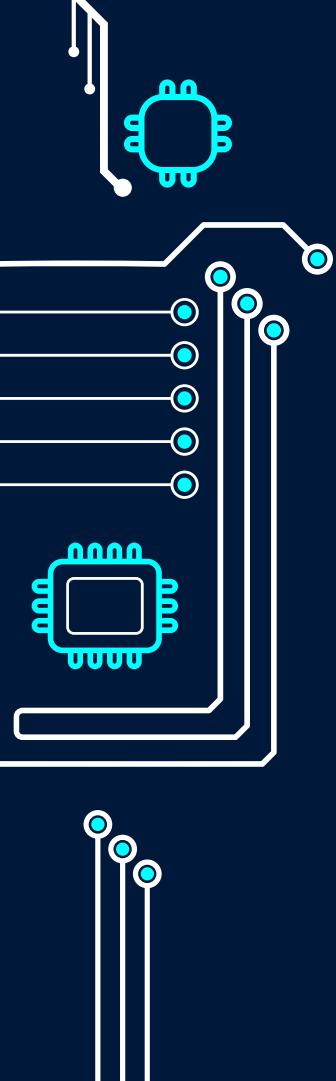
- continue: ignora todos os comandos que o loop atual ainda deveria fazer nessa iteração e passa para a próxima iteração instantaneamente.

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main(){
5
6     for(int i=1;i<=100;i++){ // repete uma sequência de comandos 100 vezes
7
8         if(i % 7 == 0){ // se o resto da divisão de i por 7 for igual a 0
9             continue; // ignora todos os outros comandos abaixo que seriam executados
10            // e segue para a próxima iteração, ou seja, próximo valor de i
11        }
12
13        cout << i << " ";
14    }
15    cout << "\n";
16 }
```

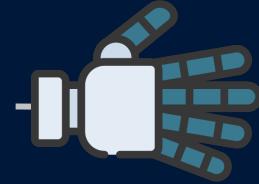


MATEMÁTICA

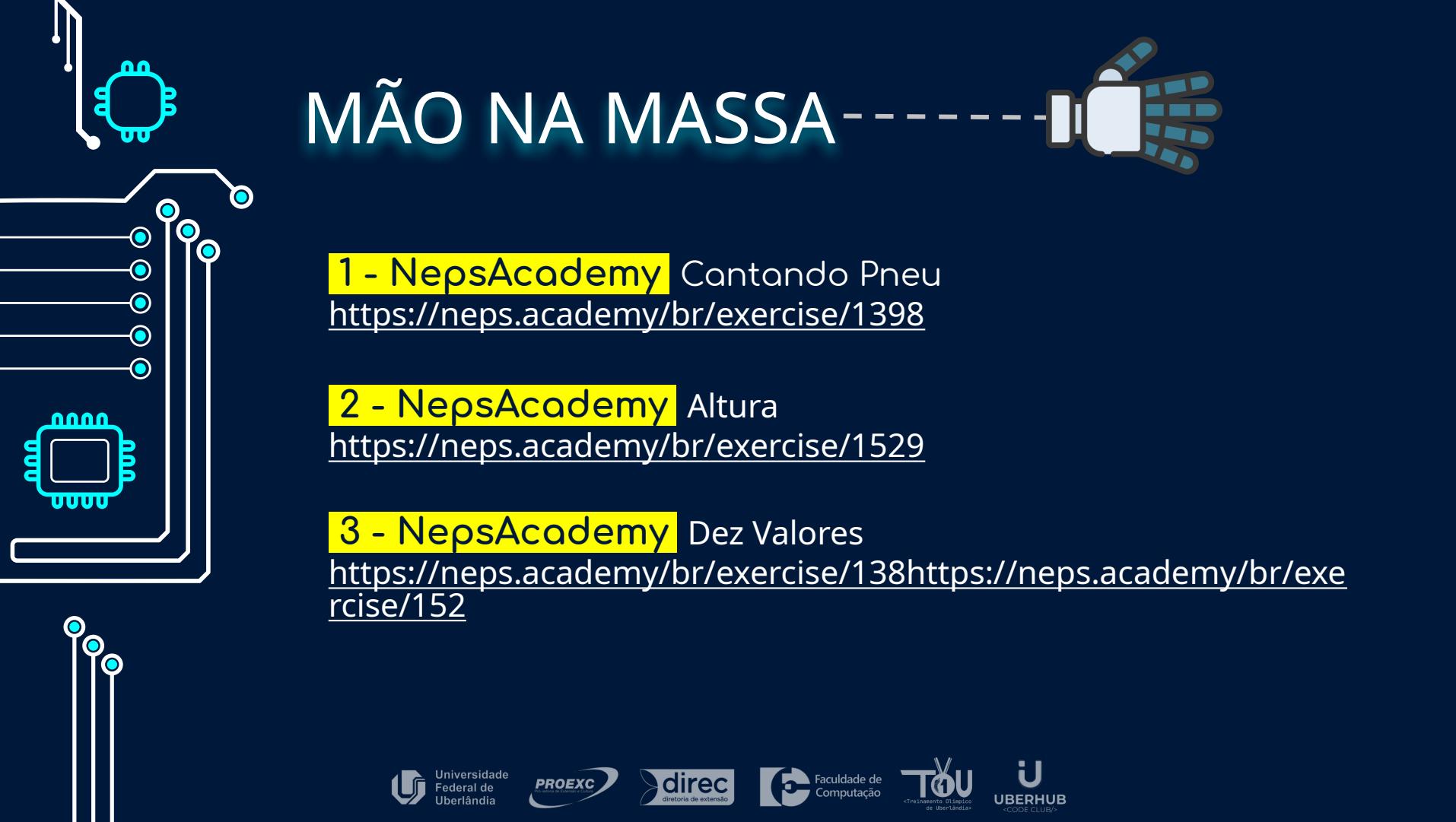
- Biblioteca: `#include <bits/stdc++.h>`
- Funções:
 - `pow(x, y);` // recebe os parâmetros x e y e retorna x elevado a y.
 - `sqrt(x);` // retorna a raiz quadrada de um número.
 - `abs(x);` // recebe um número inteiro ou real como argumento e retorna seu valor absoluto, ou seja se x é -2, a função retorna 2, e se x é 2, a função retorna 2 também.
 - `hypot(x-x1,y-y1);` // retorna a distância entre dois pontos (x,y) e (x1,y1). Essa é uma forma rápida de calcular $d^2 = (x-x1)^2 + (y-y1)^2$.



MÃO NA MASSA

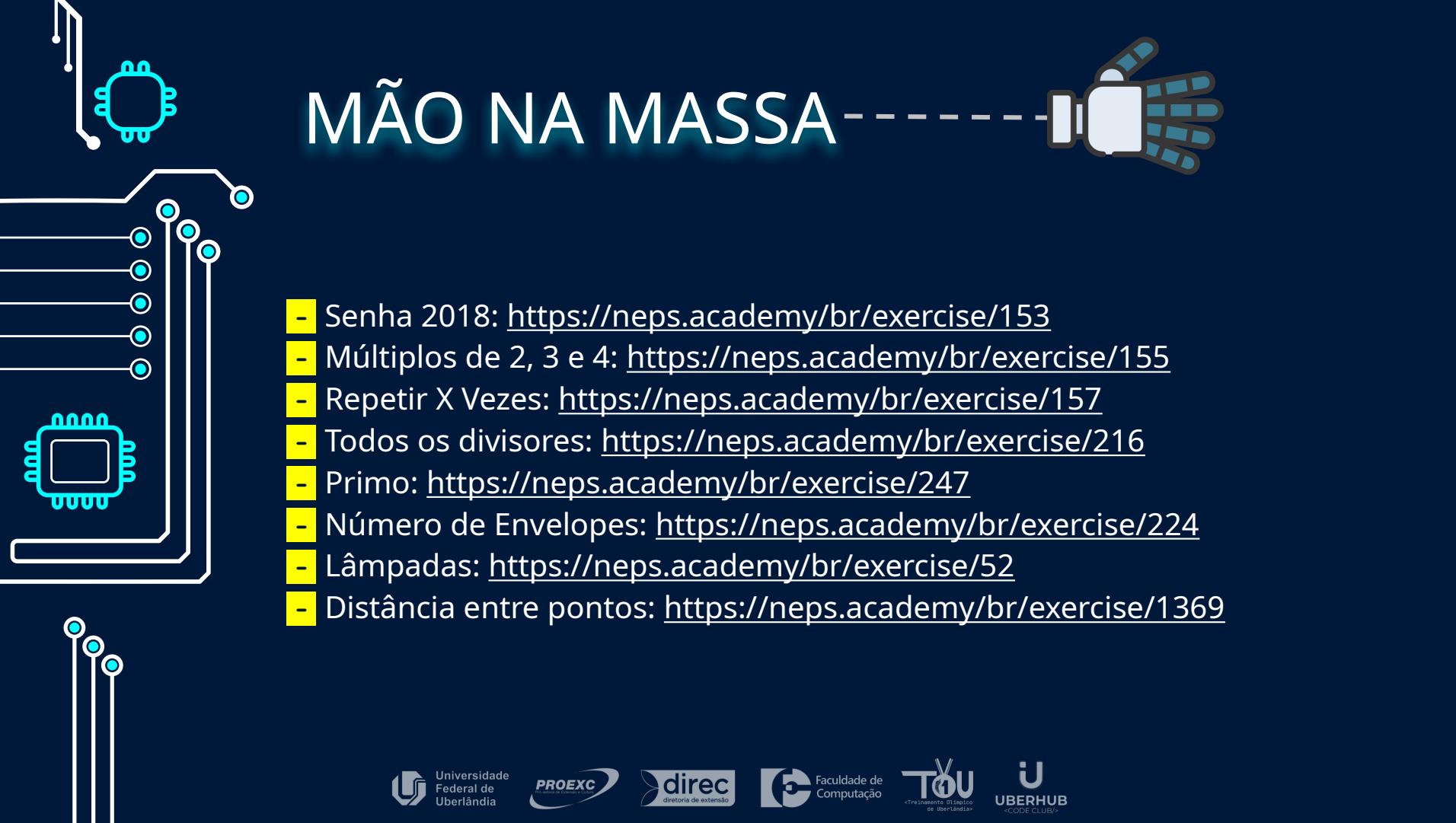


- 1** Faça um algoritmo em C++, utilizando while, que printe os números de 1 a 100.
- 2** Faça um algoritmo em C++ utilizando for, que printe os números de 1 a 100.
- 3** Faça um algoritmo em C++ para calcular a média de N números, o valor de N é dado pelo usuário.
- 4** Faça um algoritmo em C++ que leia um número positivo e imprima seus divisores.





VAMOS PRATICAR



- Senha 2018: <https://neps.academy/br/exercise/153>
- Múltiplos de 2, 3 e 4: <https://neps.academy/br/exercise/155>
- Repetir X Vezes: <https://neps.academy/br/exercise/157>
- Todos os divisores: <https://neps.academy/br/exercise/216>
- Primo: <https://neps.academy/br/exercise/247>
- Número de Envelopes: <https://neps.academy/br/exercise/224>
- Lâmpadas: <https://neps.academy/br/exercise/52>
- Distância entre pontos: <https://neps.academy/br/exercise/1369>



OBRIGADO



@projetotouuberlandia



Universidade
Federal de
Uberlândia



PROEXC
diretoria de extensão



direc
diretoria de extensão



Faculdade de
Computação



TOU
Tecnológico da U



UBERHUB
<CODE CLUB/>



CREDITS: This presentation template was created by [Slidesgo](#), and includes icons by [Flaticon](#), and infographics & images by [Freepik](#)

