

# Programação e Desenvolvimento de Software I

## Repetição

---

Prof. Héctor Azpúrua  
(slides adaptados do Prof. Pedro Olmo)

# Comandos **break** e **continue**

- Em alguns programas, durante um processamento iterativo, pode ser necessário:
  - **Encerrar o processamento iterativo** independentemente do valor da condição do laço
  - **Executar apenas parcialmente uma iteração**, ou seja, executar somente algumas das instruções do laço da repetição
- Para encerrar um processamento iterativo, independentemente do valor da condição do laço, deve-se usar o comando **break**

# Comandos **break** e **continue**

- Exemplo:
  - Dados os valores  $N$  (int) e  $A$  (float), determine a partir de qual termo o valor de:

$$s = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{N}$$

é maior do que  $A$

- Suponha  $N = 10$  e  $A = 2.0$

Instante	Valor de $S$
1º termo	1.000000
2º termo	1.500000
3º termo	1.833333
4º termo	2.083333

↑  
A partir do quarto  
termo  $s > A$

# Comandos **break** e **continue**

- Um programa para resolver este problema pode ser escrito como:

```
i = 1;
s = 0;
while (i <= N) {
    s = s + 1.0 / i;
    if (s > a) {
        printf("Numero de termos = %d", i);
        break;
    }
    i++;
}
```

- Neste caso, a **condição do laço** controla apenas o numero de termos do somatório
- O laço pode ser encerrado quando  $s > a$ , usando-se o comando **break**

# Comandos **break** e **continue**

- Para executar somente algumas das instruções do laço, mas sem encerrar a repetição:
  - Comando **continue**
- **Exemplo:**
  - Ler a idade e o peso de **N** pessoas e determinar a soma dos pesos das pessoas com mais de 30 anos

```
printf("Valor de N: ");
scanf("%d", &N);
s = 0;
i = 1;
while (i <= N) {
    printf("Idade e peso da pessoa %d:", i);
    scanf("%d %f", &idade, &peso);
    if (idade <= 30)
        continue;

    s = s + peso;
    i++;
}
printf("peso total = %.2f\n", s);
```

# Comandos **break** e **continue**

```
./break1
Valor de N: 3
Idade e peso da pessoa 1:20 171
Idade e peso da pessoa 1:19 150
Idade e peso da pessoa 1:28 182
Idade e peso da pessoa 1:32 171
Idade e peso da pessoa 2:45 190
Idade e peso da pessoa 3:80 145
peso total = 506.00
```

Só são contabilizadas entradas  
que tenham a idade correta!

```
printf("Valor de N: ");
scanf("%d", &N);
s = 0;
i = 1;
while (i <= N) {
    printf("Idade e peso da pessoa %d:", i);
    scanf("%d %f", &idade, &peso);
    if (idade <= 30)
        continue;

    s = s + peso;
    i++;
}
printf("peso total = %.2f\n", s);
```

# Comando For

## Problema: Teste de primalidade

### ■ Definição:

- Um número é primo se ele SOMENTE for divisível por 1 e por ele mesmo
- Para identificar um número  $P$  como primo, desta forma, devemos verificar se os números de 2 a  $P-1$  não geram resto na divisão

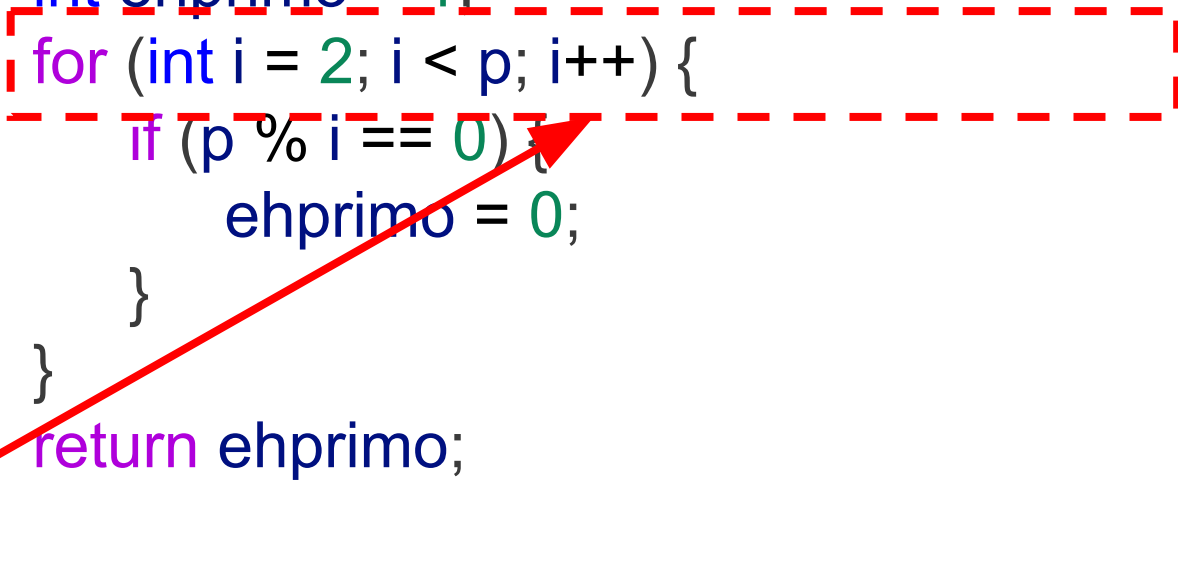
### ■ Algoritmo:

- Iremos iterar sobre os valores de 2 a  $p-1$
- Precisamos guardar, de alguma forma, que o número é primo ou não
  - variável booleana – verdadeiro se é primo, falso senão
- Se o resto der zero, marcar como falso.

# Comando For

Problema: Teste de primalidade

```
int primo(int p) {  
    int ehprimo = 1;  
    for (int i = 2; i < p; i++) {  
        if (p % i == 0) {  
            ehprimo = 0;  
        }  
    }  
    return ehprimo;  
}
```

A red dashed rectangle encloses the for loop and the if statement. A red arrow points from the text 'Novo comando!' to the if statement.

Novo comando!



# Comando For

- A diferença do **for** com o `while` e `do-while`:
  - O comando `for` consegue determinar uma:
    - **condição de inicio,**
    - **parada e**
    - **incremento**
  - Variáveis diferentes teriam que ser criadas manualmente no `while`

O comando **for** permite que um conjunto de instruções seja executado de forma cíclica, usando uma variável para controlar o fluxo do laço e o seu incremento

# Comando For

Problema: Teste de primalidade


```
for (condição inicial; condição parada; incremento) {  
    // fazer alguma coisa  
}
```



```
for (int cont = 0; cont < 10; cont = cont + 1) {  
    printf("%d\n", cont);  
}
```

```
for (int cont = 0; cont < 10; cont++) {  
    printf("%d\n", cont);  
}
```

Forma mais comum  
de incremento!



# Comando For

Problema: Teste de primalidade

- Qual é o escopo de `i`?

```
int primo(int p) {  
    int ehprimo = 1;  
    for (int i = 2; i < p; i++) {  
        if (p % i == 0) {  
            ehprimo = 0;  
        }  
    }  
    return ehprimo;  
}
```

# Comando For

## Problema: Teste de primalidade

- Qual é o escopo de `i`?
  - Só existe dentro do `for`!
- Esse código tem algum problema?
  - Ele vai ter que percorrer todos os números de 2 até `p`, mesmo que o número não seja primo...
  - Como melhorar?
    - Break!

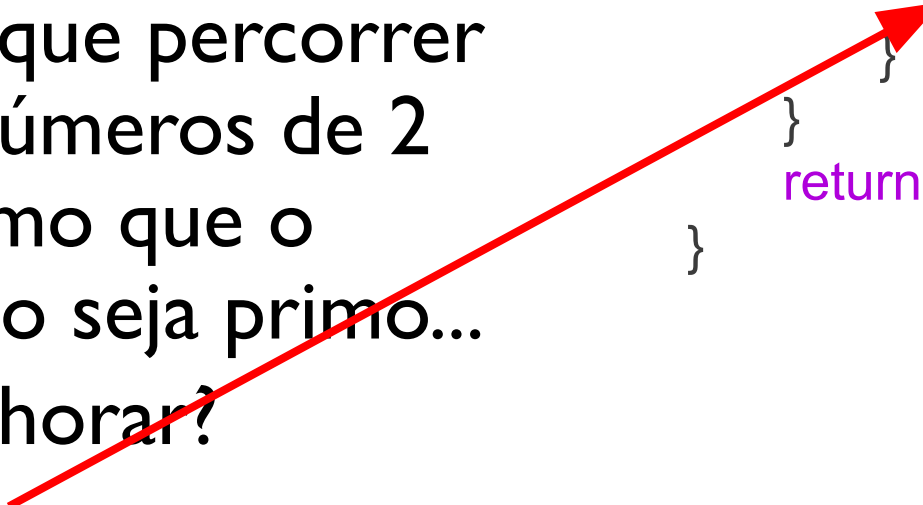
```
int primo(int p) {  
    int ehprimo = 1;  
    for (int i = 2; i < p; i++) {  
        if (p % i == 0) {  
            ehprimo = 0;  
        }  
    }  
    return ehprimo;  
}
```

# Comando For

## Problema: Teste de primalidade

- Qual é o escopo de `i`?
  - Só existe dentro do `for`!
- Esse código tem algum problema?
  - Ele vai ter que percorrer todos os números de 2 até `p`, mesmo que o número não seja primo...
  - Como melhorar?
    - Break!

```
int primo(int p) {  
    int ehprimo = 1;  
    for (int i = 2; i < p; i++) {  
        if (p % i == 0) {  
            ehprimo = 0;  
            break;  
        }  
    }  
    return ehprimo;  
}
```



# Comando For

Problema: Teste de primalidade

## ■ Outra forma de fazer?

```
int primo2(int p) {  
    int ehprimo = 1;  
    for (int i = 2; i < p && ehprimo == 1; i++) {  
        if (p % i == 0) {  
            ehprimo = 0;  
        }  
    }  
    return ehprimo;  
}
```

Condição de parada  
diretamente no loop!

# Laços aninhados

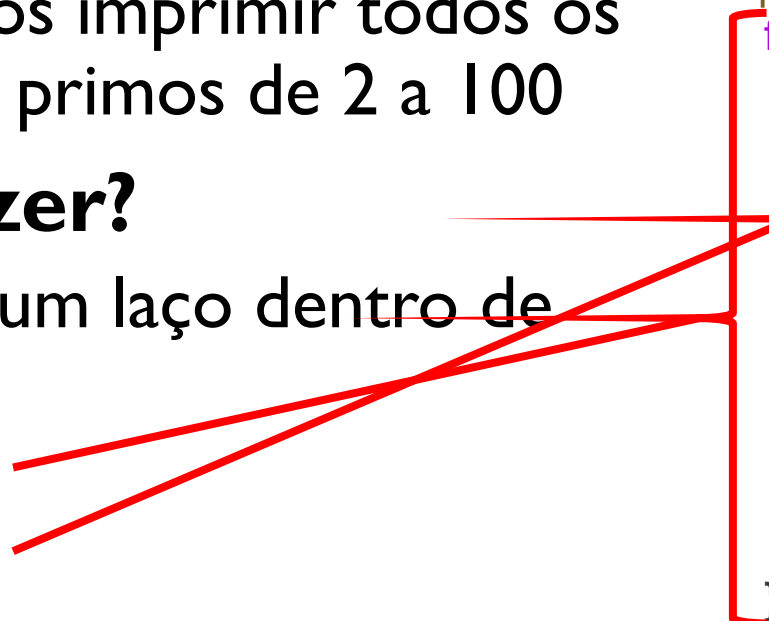
## ■ Problema:

- Queremos imprimir todos os números primos de 2 a 100

## ■ Como fazer?

- Colocar um laço dentro de outro!
  - Laço 1
  - Laço 2

```
void primos() {  
    for (int p = 2; p <= 100; p++) {  
        int ehprimo = 1;  
        for (int i = 2; i < p && ehprimo == 1; i++) {  
            if (p % i == 0)  
                ehprimo = 0;  
        }  
        if (ehprimo == 0)  
            printf("%d nao eh primo\n", p);  
        else  
            printf("%d eh primo\n", p);  
    }  
}
```



# Considerações finais

- Existem 3 tipos de laços no C:

1. `while`

- **Não se sabe com anterioridade** quantas vezes vai ser executado o laço

2. `do-while`

- Similar ao `while` mas é garantido que **pelo menos uma vez** o laço vai ser executado

3. `for`

- **Você sabe** quantas vezes vai rodar o laço



# Perguntas?

- E-mail:
  - [hector@dcc.ufmg.br](mailto:hector@dcc.ufmg.br)
- Material da disciplina:
  - <https://pedroolmo.github.io/teaching/pdsI.html>
- Github:
  - <https://github.com/h3ct0r>



**Héctor Azpúrua**  
h3ct0r