

Introdução à Linguagem C

O que é

- Criada na década de 70 por Dennis Ritchie e Ken Thompson
- Linguagem Imperativa estruturada
 - Baseada em comandos que alteram estados de variáveis
- Compilada
 - Diferente de Python que era interpretada
- Fortemente tipada e case sensitive
 - Cada variável tem um tipo específico, esse domínio, é mantido durante a execução do programa
- Possui características de alto nível e baixo nível
 - Aceita inclusive que você execute códigos em Assembly em seu interior
- Utilizada para os mais variados propósitos
 - Utilizada na implementação do Kernel do Linux

Sumário

- **Primeiro Programa**
- Entrada e Saída
- Condições
- Repetições

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Meu primeiro código em C! \n");
6      return 0;
7  }
```

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Meu primeiro código em C! \n");
6      return 0;
7  }
```

Importa uma
biblioteca

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Meu primeiro código em C! \n");
6      return 0;
7  }
```

Bloco principal

Primeiro Programa

```
1  #include <stdio.h>
```

```
2
```

```
3  int main()
```

```
4  {
```

```
5      printf("C! \n");
```

```
6      return 0;
```

```
7  }
```

Delimitadores de bloco.

Em Python, essa delimitação era feita com indentação.

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Meu primeiro código em C! \n");
6      return 0;
7  }
```

Linhas que executam ação
terminam em ';'

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Meu primeiro código em C! \n");
6      return 0;
7  }
```

- Como compilar e rodar?

compila

```
gschreiner@nasgul:~$ gcc hellow.c -o prog
```

```
gschreiner@nasgul:~$ ./prog
```

executa

```
Meu primeiro código em C!
```

```
gschreiner@nasgul:~$
```

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Meu primeiro código em C!");
6      return 0;
7  }
```

Testar com valores pré-definidos

```
gschreiner@nasgul:~$ ./prog < arq.in > arq.out
gschreiner@nasgul:~$ diff arq.out arq.res
gschreiner@nasgul:~$
```

- Como compilar e rodar?

```
gschreiner@nasgul:~$ gcc hellow.c -o prog
gschreiner@nasgul:~$ ./prog
Meu primeiro código em C!
gschreiner@nasgul:~$
```

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Meu primeiro código em C! \n");
6      return 0;
7  }
```

Podemos usar qualquer uma das duas formas para o **main**

`int main() {`

`int main(int argc, char const *argv[]){`

Permite passar parâmetros no momento da execução do programa

Sumário

- Primeiro Programa
- **Entrada e Saída**
- Condições
- Repetições

Entrada de dados

```
1  #include <stdio.h>
2
3  int
4  {
5      int i;
6      printf("Digite um inteiro: \n");
7      scanf("%d", &i);
8
9      printf("Você digitou o número: %d \n", i);
10     return 0;
11 }
```

Tipo da variável

Entrada de dados

```
1  #include <stdio.h>
2
3  int
4  {
5      int i;
6      printf("Digite u
7      scanf("%d", &i);
8
9      printf("Você dig
10     return 0;
11 }
```

Tipo da variável

Tipo	Bytes	Escala
char	1	-128 a 127
int	4	-2.147.483.648 a 2.147.483.647
short	2	-32.765 a 32.767
long	4	-2.147.483.648 a 2.147.483.647
unsigned char	1	0 a 255
unsigned	4	0 a 4.294.967.295
unsigned long	4	0 a 4.294.967.295
unsigned short	2	0 a 65.535
float	4	$3,4 \times 10^{-38}$ a $3,4 \times 10^{38}$
double	8	$1,7 \times 10^{-308}$ a $3,4 \times 10^{308}$
long double	10	$3,4 \times 10^{-4932}$ a $3,4 \times 10^{4932}$
void	0	nenhum valor

Entrada de dados

```
1  #include <stdio.h>
2
3  int main
4  {
5      int i;
6      printf("Digite um inteiro: \n");
7      scanf("%d", &i);
8
9      printf("Você digitou o número: %d \n", i);
10     return 0;
11 }
```

Nome da variável

Entrada de dados

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6      printf("Digite um inteiro");
7      scanf("%d", &i);
8
9      printf("Você digitou o número: %d \n", i);
10     return 0;
11 }
```

Lê um inteiro

Entrada de dados

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6      printf("Digite um inteiro");
7      scanf("%d", &i);
8
9      printf("Você digitou o número: %d \n", i);
10     return 0;
11 }
```

Formato de leitura

Lê um inteiro

Entrada de dados

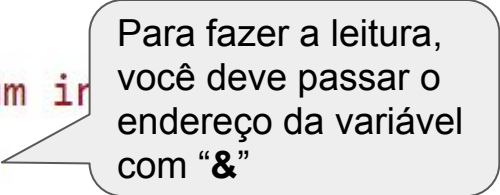
```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6      printf("Digite um inteiro: ");
7      scanf("%d", &i);
8
9      printf("Você digitou: %d\n", i);
10     return 0;
11 }
```

Formato de leitura

Código	Função
<u>%c</u>	Lê um único caractere
<u>%i</u> ou <u>%d</u>	Lê um inteiro decimal
<u>%e</u>	Lê um número em notação científica
<u>%f</u>	Lê um número em ponto flutuante
<u>%o</u>	Lê um número octal
<u>%s</u>	Lê uma <u>string</u>
<u>%x</u>	Lê um número hexadecimal
<u>%u</u>	Lê um decimal sem sinal
<u>%li</u> ou <u>%ld</u>	Lê um inteiro longo
<u>%lf</u>	Lê um double
<u>%Lf</u>	Lê um long double

Entrada de dados

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6      printf("Digite um inteiro: ");
7      scanf("%d", &i);
8
9      printf("Você digitou o número: %d \n", i);
10     return 0;
11 }
```



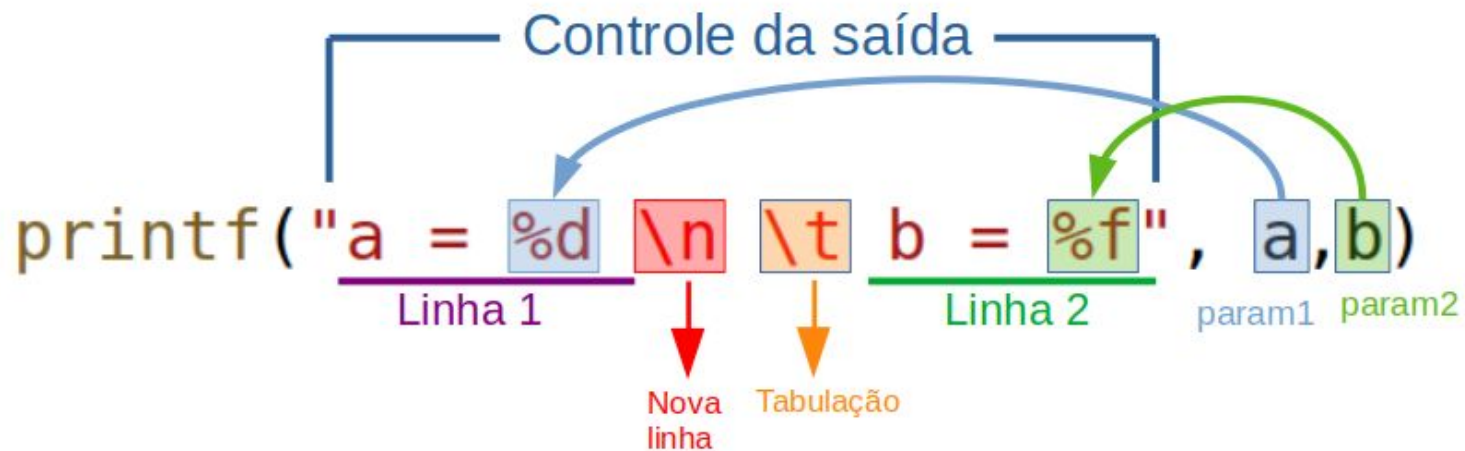
Para fazer a leitura, você deve passar o endereço da variável com "&"

Saída

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6      printf("Digite um inteiro: \n");
7      scanf("%d", &i);
8
9      printf("Você digitou o número: %d \n", i);
10     return 0;
11 }
```

Exemplo de print
com um inteiro.

Saída



Não esqueça...

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i;
6      printf("Digite um inteiro: \n");
7      scanf("%d", &i);
8
9      printf("Você digitou o número: %d \n", i);
10     return 0;
11 }
```



Equivalência de códigos

- Exemplo leitura de dois valores inteiros (um em cada linha):

Python

```
1  # pegando entradas
2  A = int(input())
3  B = int(input())
4
5  # lógica
6  X = A + B
7
8  # saída
9  print("X = {}".format(X))
```

C

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a, b, x;
6      scanf("%d", &a);
7      scanf("%d", &b);
8
9      x = a + b;
10
11     printf("X = %d \n", x);
12
13     return 0;
14 }
```

Equivalência de códigos

- Exemplo leitura de dois valores inteiros (um em cada linha):

Python

```
1  # pegando entradas
2  A = int(input())
3  B = int(input())
4
5  # lógica
6  X = A + B
7
8  # saída
9  print("X = {}".format(X))
```

C

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a, b, x;
6      scanf("%d", &a);
7      scanf("%d", &b);
8
9      x = a + b;
10
11     printf("X = %d \n", x);
12
13     return 0;
14 }
```


Equivalência de códigos

- Exemplo leitura de dois valores inteiros (um em cada linha):

Python

```
1  # pegando entradas
2  A = int(input())
3  B = int(input())
4
5  # lógica
6  X = A + B
7
8  # saída
9  print("X = {}".format(X))
```

C

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a, b, x;
6      scanf("%d", &a);
7      scanf("%d", &b);
8
9      x = a + b;
10
11     printf("X = %d \n", x);
12
13     return 0;
14 }
```

Equivalência de códigos

- Exemplo leitura de dois valores inteiros (um em cada linha):

Python

```
1  # pegando entradas
2  A = int(input())
3  B = int(input())
4
5  # lógica
6  X = A + B
7
8  # saída
9  print("X = {}".format(X))
```

C

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int a, b, x;
6      scanf("%d", &a);
7      scanf("%d", &b);
8
9      x = a + b;
10
11     printf("X = %d \n", x);
12
13     return 0;
14 }
```

Equivalência de códigos

Exemplos de Entrada

2.00

```
1  # entrada
2  raio = float(input())
```

```
float raio;
scanf("%f", &raio);
```

Exemplos de Saída

A=12.5664

```
1  # saída
2  print("A={:.4f}".format(area))
```

```
printf("A = %.4f \n",raio);
```

Equivalência de códigos

- Exemplo leitura de mais de um valor por linha:

Exemplos de Entrada

7	14	106
---	----	-----

```
1 # entrada
2 A, B, C = map(int, input().split(" "))
```

```
int a,b,c;
scanf("%d %d %d", &a, &b, &c);
```

Sumário

- Primeiro Programa
- Entrada e Saída
- **Condições**
- Repetições

Condições

- Condições em Python:

```
1  if condicao:
2      bloco
3  else:
4      bloco
```

```
1  if condicao:
2      bloco
3  elif condicao:
4      bloco
5  else:
6      bloco
```

- Condições em C:

```
1  if (condicao){
2      bloco;
3  } else {
4      bloco;
5  }
```

```
1  if (condicao){
2      bloco;
3  } else if (condicao) {
4      bloco;
5  } else {
6      bloco;
7  }
```

Condições

- Condição em Python

```
1 idade = 18
2 if idade >= 18:
3     print('maior de idade')
4 else:
5     print('menor de idade')
```

- Condição em C

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[]){
4      int idade = 18;
5
6      if (idade >= 18){
7          printf("Maior de idade");
8      } else {
9          printf("Menor de idade");
10     }
11
12     return 0;
13 }
```

Condições

- Condição em Python

```
1 idade = 18
2 if idade >= 18:
3     print('maior de idade')
4 else:
5     print('menor de idade')
```

- Condição em C

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[]){
4      int idade = 18;
5
6      if (idade >= 18){
7          printf("Maior de idade");
8      } else {
9          printf("Menor de idade");
10     }
11
12     return 0;
13 }
```


Condições

- Condição em C

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[]){
4      int val = 0;
5
6      if (val){
7          printf("verdadeiro");
8      } else {
9          printf("falso");
10     }
11
12     return 0;
13 }
```

Condições

- Condição em C

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[])
4  {
5      int val = 0;
6
7      if (val){
8          printf("verdadeiro");
9      } else {
10         printf("falso");
11     }
12
13     return 0;
14 }
```

Qualquer valor
!= de 0 é verdadeiro

Condições

- Switch / case

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int x;
6      scanf("%d", &x);
7      switch (x){
8          case 1:
9              bloco; // executa caso x == 1
10             break;
11         case 2:
12             bloco; // executa caso x == 2
13             break;
14         default:
15             bloco; // nenhuma das opções anteriores
16     }
17     return 0;
18 }
```

Sumário

- Primeiro Programa
- Entrada e Saída
- Condições
- **Repetições**

Repetições

- For em Python:

```
1 for x in range(6):  
2     print(x)  
3 else:  
4     print("Finally finished!")
```

- For em C:

```
1  #include <stdio.h>  
2  
3  int main(int argc, char const *argv[]){  
4      int i;  
5  
6      for ( i=0 ; i < 6; i++ ){  
7          printf("%d",i);  
8      }  
9      printf("Finally Finished!");  
10  
11     return 0;  
12 }
```

Repetições

- For em Python:

```
1 for x in range(6):  
2     print(x)  
3 else:  
4     print("Finally finished!")
```

- For em C:

```
1  #include <stdio.h>  
2  
3  int main(int argc, char const *argv[]){  
4      int i;  
5  
6      for ( i=0 ; i < 6; i++ ){  
7          printf("%d",i);  
8      }  
9      printf("Finally Finished!");  
10  
11     return 0;  
12 }
```

Repetições

- For em Python

```
1 for x in range(10):
2     print(x)
3 else:
4     print("Fim")
```

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[]){
4
5      const *argv[]){
6
7      for ( i=0 ; i < 6; i++ ){
8          printf("%d\n", i);
9      }
10     printf("Finally finished!");
11
12     return 0;
13 }
```

De onde começa

Como vai

Até quando vai

Repetições

- While em Python:

```
1 i = 1
2 while i < 6:
3     print(i)
4     i += 1
```

- While em C:

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[]){
4      int i = 1;
5      while (i < 6)
6      {
7          printf("%d",i);
8          i++; //mesma coisa que i+=1
9      }
10
11     return 0;
12 }
```


Repetições

- Do While em C:

```
1  #include <stdio.h>
2
3  int main() {
4      int i = 1;
5      do {
6          printf("%d, ", i);
7          i++;
8      } while (i < 6);
9
10     return 0;
11
12 }
```

Repetições

- While em C:

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[]){
4      int i = 1;
5      while (i < 6)
6      {
7          printf("%d",i);
8          i++; //mesma coisa que i+=1
9      }
10
11     return 0;
12 }
```

Condição é testada antes

- Do While em C:

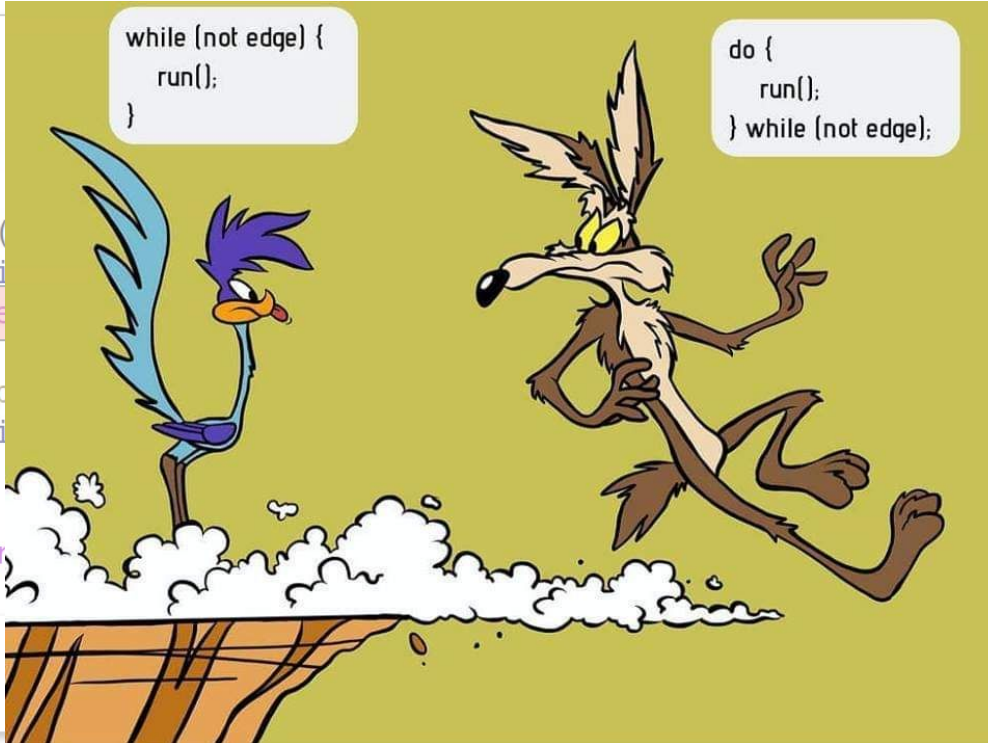
```
1  #include <stdio.h>
2
3  int main() {
4      int i = 1;
5      do {
6          printf("%d, ", i);
7          i++;
8      } while (i < 6);
9
10     return 0;
11
12 }
```

Condição é testada depois

Repetições

- While em

```
1  #include
2
3  int main()
4  {
5      int i;
6      while (i < 6)
7      {
8          printf("%d\n", i);
9          i++;
10     }
11     return 0;
12 }
```



m C

```
<stdio.h>

() {
i = 1;
printf("%d\n", i);
i++;
while (i < 6);
return 0;
```

Repetição + Leitura de dados

- Leitura de dado até o fim do arquivo

- Python:

```
1 while True:
2     try:
3
4         # entrada
5         # lógica
6         # saída pode vir aqui
7
8     except EOFError:
9         break
10 # saída pode vir aqui
```

- C:

```
1 #include <stdio.h>
2
3 int main() {
4     int i;
5
6     while (scanf("%d", &i) != EOF){
7         //Logica que eu quiser
8     }
9
10    return 0;
11 }
```