

Introdução a C

Prof. Denio Duarte

Prof. Geomar Schreiner

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
 - Compilador sugerido: **gcc** (**g**nu **c**ompiler **c**ollection)
- 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

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[])
4  {
5
6      printf("Meu primeiro código em C! \n"); //mesma ideia do print em Python
7
8      return 0;
9  }
```

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[])
4  {
5
6      printf("Meu primeiro código em C! \n"); //mesma ideia do print em Python
7
8      return 0;
9  }
```

Importa uma biblioteca para utilizar
funções de entrada e saída:
standard input output (.h → header)

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[])
4  {
5
6
7
8      return 0;
9  }
```

Bloco principal

Tudo que estiver
aqui é executado.

O que estiver fora é
apenas compilado

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[])
4  {
5
6      printf("Olá mundo em C! \n"); //mesma ideia do print em Python
7
8      return 0;
9  }
```

Delimitadores de bloco.
Em Python essa delimitação era feita com indentação.

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[])
4  {
5
6      printf("Meu primeiro código em C! \n"); //mesma ideia do print em Python
7
8      return 0;
9  }
```

Linhas que executam ação terminam em ';'

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[])
4  {
5
6      printf("Meu primeiro código em C! \n"); //mesma ideia do print em Python
7
8      return 0;
9  }
```

- Como compilar e rodar?

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[])
4  {
5
6      printf("Meu primeiro código em C! \n"); //mesma ideia do print em Python
7
8      return 0;
9  }
```

- Como compilar e rodar?

home:~\$ gcc -Wall hello.c -o hello
home:~\$./hello
Meu primeiro código em C!
home:~\$

Diretivas para o compilador:
-Wall informa todos os possíveis problemas no programa, por exemplo, variável criada e não utilizada.
-o nome do programa executável

Primeiro Programa

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[])
4  {
5
6      printf("Meu primeiro código em C! \n"); //mesma ideia do print em Python
7
8      return 0;
9  }
```

Direciona o conteúdo do testes.in para a entrada de hello

Direciona a saída de hello para saida.out

- Como compilar e rodar?

```
home:~$ gcc -Wall hello.c -o hello
home:~$ ./hello
Meu primeiro código em C!
home:~$
```

```
home:~$ ./hello < testes.in > saida.out
home:~$ diff saida.out correto.out
home:~$
```

diff verifica se os dois arquivos são iguais, neste caso saida.out (gerado pelo programa) e correto.out (saída correta para as entradas de teste.in)

Entrada de dados

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

Tipo da variável

Entrada de dados

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

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

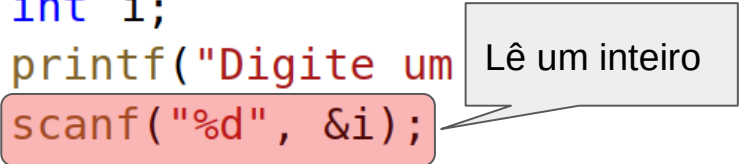
Entrada de dados

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

Nome da variável

Entrada de dados

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



Entrada de dados

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

Formato esperado de leitura

Lê um inteiro

Entrada de dados

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

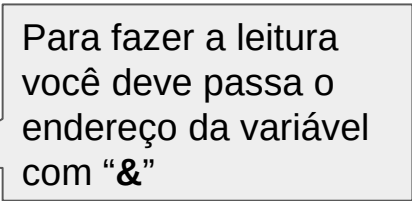
Formato de leitura

scanf("%d", &i);

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 <u>haxadecimal</u>
<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(int argc, char const *argv[])
4  {
5      int i;
6      printf("Digite um
7      scanf("%d", &i),
8
9      printf("Você digitou o número: %d", i);
10     return 0;
11 }
```



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

Entrada de dados

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

Exemplo de print
com um inteiro.

Entrada de dados

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

Controle da saída

Linha 1

Linha 2

Nova linha

Tabulação

param1 param2

Entrada de dados

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("a = %d\n\t b = %f", a, b);
6
7      scanf("%d", &i);
8
9      printf("Você digitou o número: %d", i);
10     printf("VOLUME = %.3f", s);
11 }
```

Controle da saída

Linha 1

Nova linha

Tabulação

Linha 2

param1 param2

Entrada de dados

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



Entrada de dados - Conversão de Códigos

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

Entrada de dados - Conversão de Códigos

C

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))
```

```
1  #include <stdio.h>
2
3  int main(int argc, char const *argv[])
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 }
```

Entrada de dados - Conversão de Códigos

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


Entrada de dados - Conversão de Códigos

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

Entrada de dados - Conversão de Códigos

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

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);
```

Entrada de dados - Conversão 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);
```

Condições

- Como são as condições em Python

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

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

- Condição em C

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


```
1  if (condição){
2      bloco;
3  }else
4      if (condição){
5          bloco;
6      }else {
7          bloco;
8      }
```

Condições

- Como são as condições em Python

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

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



- Condição em C

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

```
1  if (condição){  
2      |    bloco;  
3  } else  
4      if (condição):{  
5          bloco;  
6      } else {  
7          bloco;  
8      }
```

Condições

Python

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

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

Python

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

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

```
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

```
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

Repetição

- For em Python

```
1 for x in range(6):  
2     print(x)  
3 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 }
```

Repetição

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

De onde começa

Como vai

Até quando vai

Repetição

- 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 }
```

Repetição

- 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 }
```

Repetição

- 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 }
```

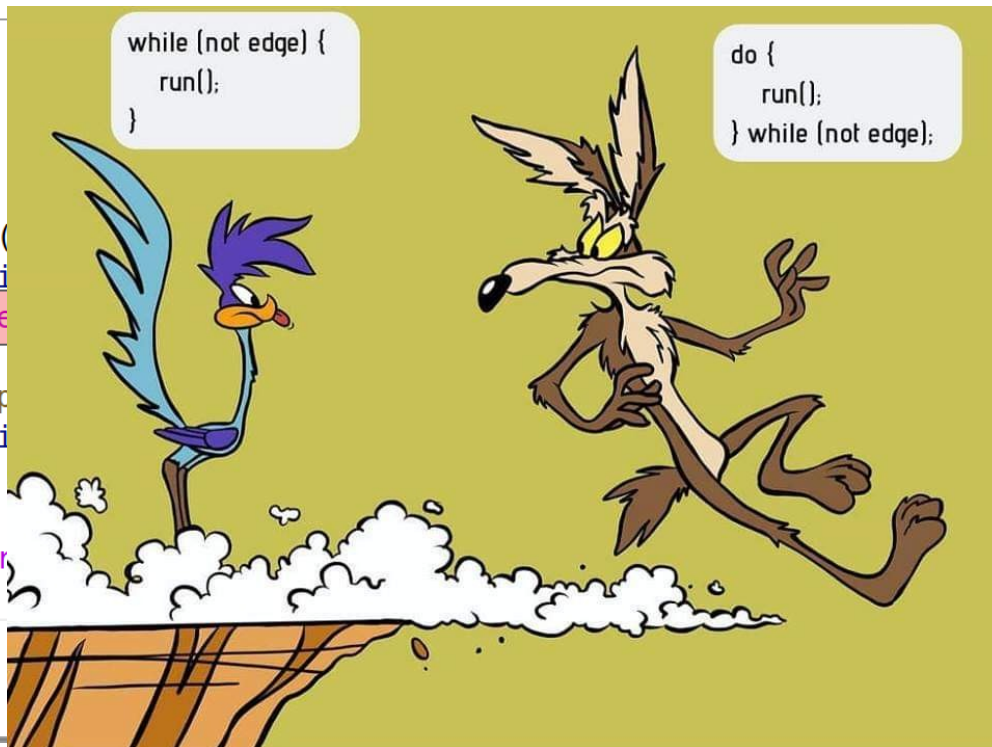
- Do While em C

```
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 }
```

Repetição

- 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 }
```



```
do {
    run();
} while (not edge);
```

m C

```
<stdio.h>

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

Repetição

- 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 }
```

Podemos usar qualquer uma das duas opções.

- 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 }
```