

C Básico

Prof. Denio Duarte

duarte@uffs.edu.br

Prof. Claunir Pavan

claunir.pavan@uffs.edu.br

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 é interpretada
 - Compilador sugerido: **gcc** (**g**nu **c**ompiler **c**ollection)
- Tipada e case sensitive
 - Cada variável tem um tipo específico, esse domínio, é mantido durante a execução do programa
- Utilizada para os mais variados propósitos
 - Utilizada na implementação do Kernel do Linux

Before start

- Os primeiros programas eram implementados em linguagem de máquina
- Era muito difícil programar (décadas de 1940 e 1950)
- Por exemplo, um programa que verificasse a possibilidade de votar:

Informe a sua idade: 20

Pode votar

Before start

- Em linguagem de baixo nível (máquina) ficaria assim:

Informe a sua idade: 20

Pode votar

Início

```
.LC0:
.string "Informe a sua idade: "
.LC1:
.string "%d"
.LC2:
.string "Pode votar"
.LC3:
.string "Não pode votar"
.text
.globl main
.type main,@function
```

```
main:
.LFB0:
.cfi_startproc
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq %rsp, %rbp
.cfi_def_cfa_register 6
subq $16, %rsp
movq %fs:40, %rax
movq %rax, -8(%rbp)
xorl %eax, %eax
```

```
leaq .LC0(%rip), %rdi
movl $0, %eax
call printf@PLT
leaq -12(%rbp), %rax
movq %rax, %rsi
leaq .LC1(%rip), %rdi
movl $0, %eax
call __isoc99_scanf@PLT
movl -12(%rbp), %eax
cmpl $16, %eax
jle .L2
leaq .LC2(%rip), %rdi
```

```
movl $0, %eax
call printf@PLT
jmp .L3
.L2:
leaq .LC3(%rip), %rdi
movl $0, %eax
call printf@PLT
.L3:
movl $0, %eax
movq -8(%rbp), %rdx
xorq %fs:40, %rdx
je .L5
```

```
call __stack_chk_fail@PLT
.L5:
leave
.cfi_def_cfa 7, 8
ret
.cfi_endproc
.LFE0:
.size main,.-main
.ident "GCC: (Ubuntu 7.4.0-1ubuntu1~18.04.1) "
.section .note.GNU-stack,"",@progbits
```

Fim

Before start

- Em linguagem de alto nível ficaria assim:

Informe a sua idade: 20
Você pode votar

//C

```
#include <stdio.h>
int main(void)
{
    int id;
    printf("Informe a sua idade: ");
    scanf("%d",&id);
    if (id>=17)
        printf("Pode votar\n");
    else
        printf("Não pode votar\n");
    return 0;
}
```

#python

```
id=input('Informe a sua idade: ')
id=int(id)
if id>=17:
    print('Pode votar')
else:
    print('Não pode votar')
```

{Pascal}

```
program vota;
var
    id : integer;
begin
    write('Informe a sua idade: ');
    readln(id);
    if id>=17
    then
        writeln('Pode votar')
    else
        writeln('Não pode votar')
end.
```

Before start

- Executando Programas

- Como um programa escrito em linguagem de alto nível é **compreendido** pelo computador?

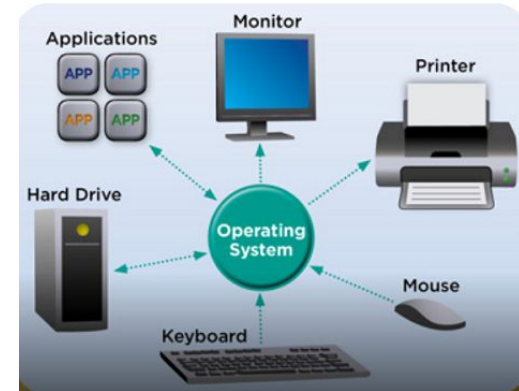
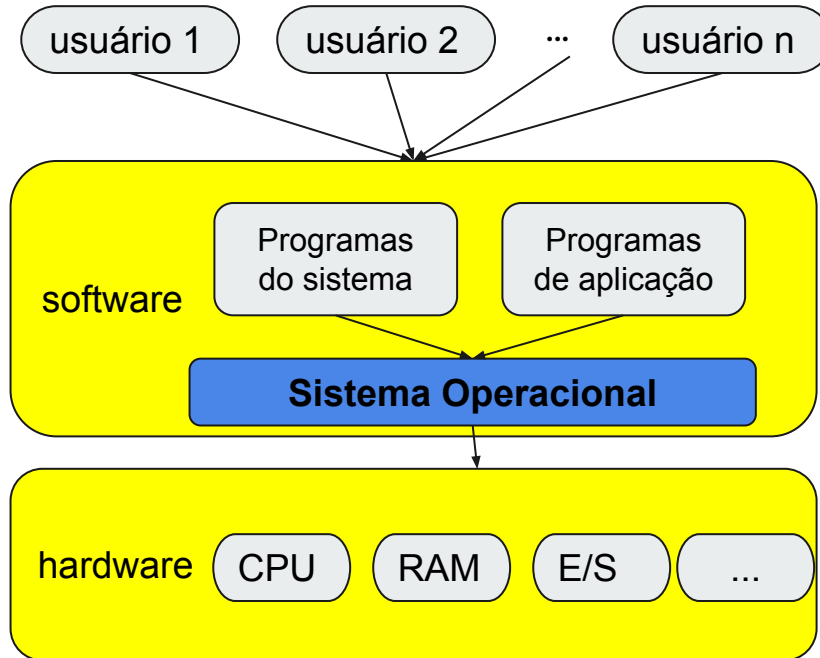
- Existe **um programa mestre** (sistema) que torna os computadores mais acessíveis pelos humanos
- Faz as transformações de nossas requisições de entrada (teclado, voz, mouse) para o computador
- Transforma as respostas para um formato compreensível por nós

- Sistema Operacional (SO)

- Gerencia os recursos de hardware e software do computador, e oferece serviços comuns para os programas serem executados

Before start

- Sistema Operacional (SO)



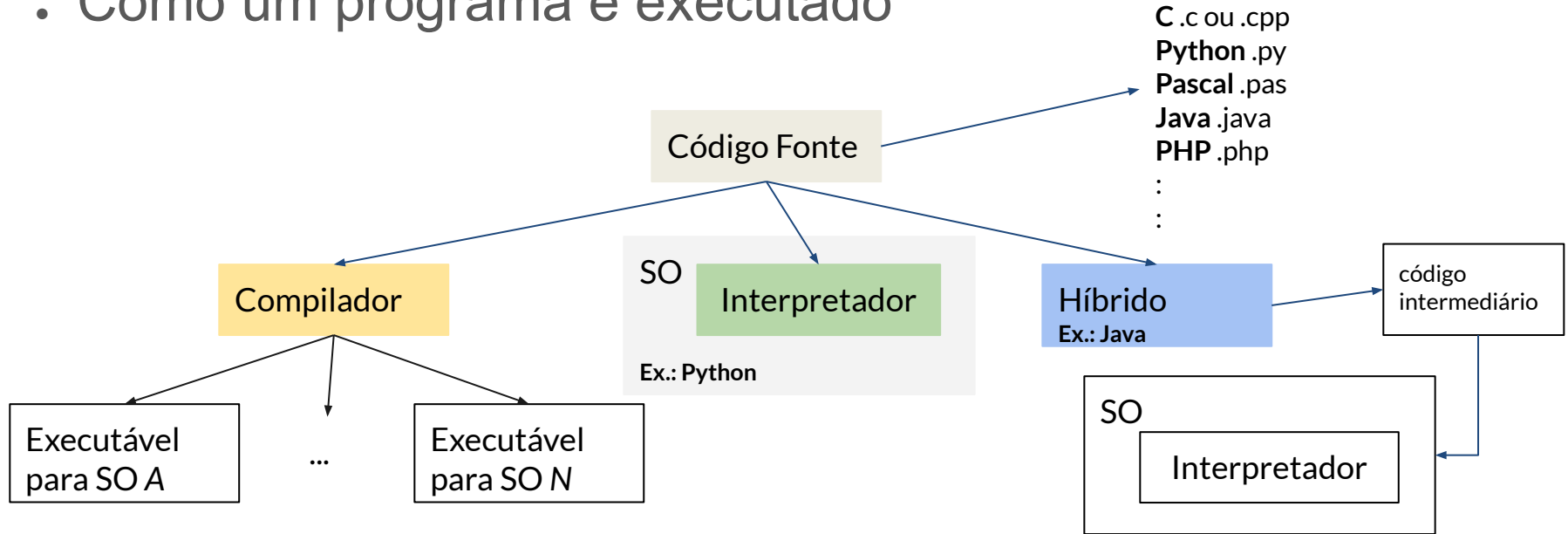
Before start

- Sistema Operacional (SO)



Before start

- Como um programa é executado



Now, Ladies and Gentlemen
C

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  }
```

```
#Python
print('Meu primeiro código em C!')
```

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 exibe 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  }
```

- Como compilar e rodar?

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

Entrada de dados

```
1  #include<stdio.h>
2
3  int c, 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

```
#Python
i=int(input('Digite um inteiro'))
print('Você digitou o número: ',i)
```

Entrada de dados

```
1  #include<stdio.h>
2
3  int i;
4  {
5      printf("Digite ");
6      scanf("%d", &i);
7
8      printf("Você digitou: ");
9
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

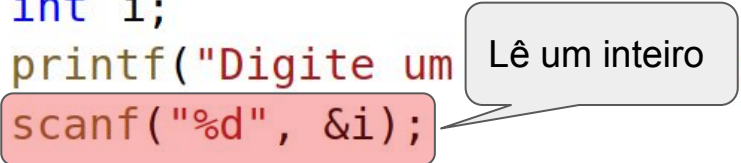
Entrada de dados

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



A diagram illustrating the execution of the `scanf` function. A light gray speech bubble containing the text "Lê um inteiro" (Reads an integer) points to the `scanf` function call on line 7. A light red rectangular box highlights the `scanf("%d", &i);` line.

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 }
```



The diagram consists of two callout boxes with arrows pointing to specific parts of the code. The first box, labeled "Formato esperado de leitura", points to the "%d" format specifier in the scanf function on line 7. The second box, labeled "Lê um inteiro", points to the variable &i in the same scanf function call.

Entrada de dados

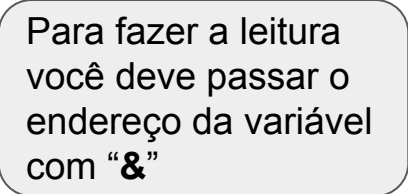
```
1  #include<stdio.h>
2
3  int main(int argc, char const* argv[])
4  {
5      int i;
6      printf("Digite um inteiro: ");
7      scanf("%d", &i);
8
9      printf("Você digitou o número %d", i);
10     return 0;
11 }
```

Formato de leitura

Descritor	Função
%c	lê um único caractere
%s	lê uma string
%d	lê um inteiro decimal (base 10)
%ld	lê um inteiro longo
%i	lê um inteiro (detecta a base automaticamente)
%li	lê um inteiro longo (detecta a base)
%hi	lê um inteiro curto (short int)
%hu	lê um inteiro curto sem sinal (unsigned int)
%f	lê um número em ponto flutuante
%lf	lê um double
%Lf	lê um long double
%o	lê um número octal (base 8)
%x	lê um número hexadecimal (base 16)
%e	lê um número em notação científica

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 passar 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 i;
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

param1 param2

Nova linha

Tabulação

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

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

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      }
9  }
```

Condições

- Como são as condições em Python

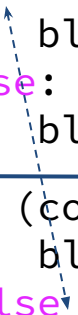
```
1  if condição:
2      bloco
3  else:
4      bloco
```

```
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      }
9  }
```



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 }
```

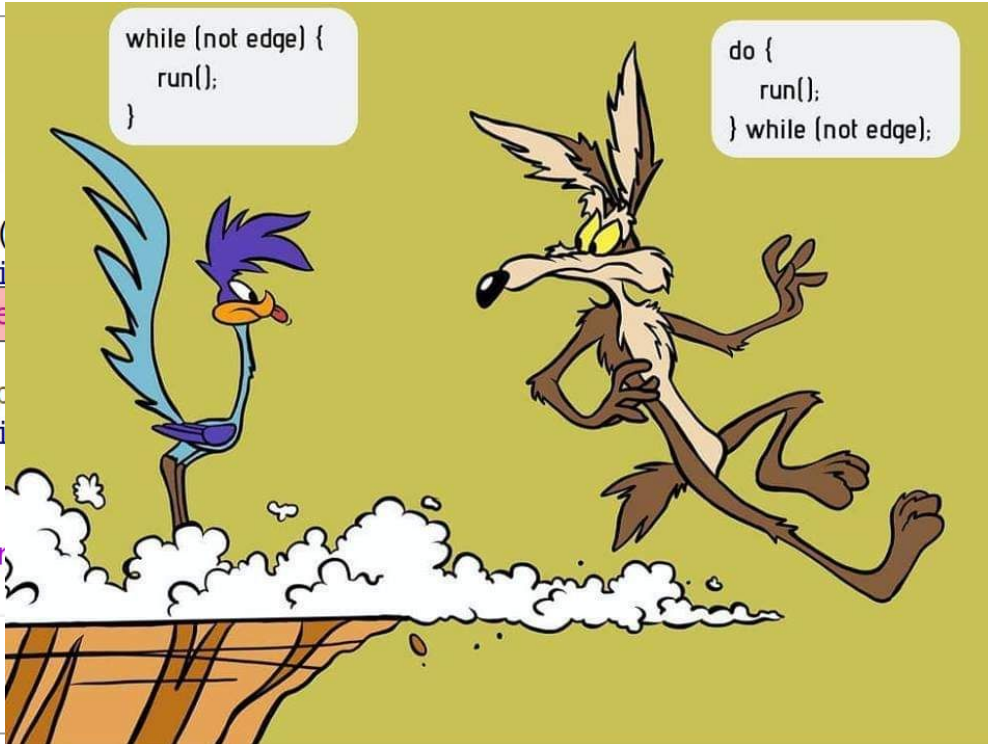
- Do While em C

```
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 }
```



m C

```
<stdio.h>

() {
i = 1;

printf("%d, ", i);
i++;
while (i < 6);

return 0;
```

Repetição

- While em C

Podemos usar qualquer uma das duas opções.

```
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 }
```

- 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 + Leitura de dados

- Leitura de dado até o fim do arquivo

- Leitura em 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
```

- Leitura em 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 }
```