



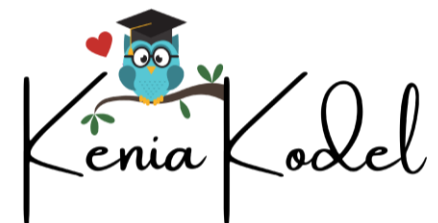
UNIVERSIDADE
FEDERAL DE
SERGIPE



Estruturas de Repetição

WHILE e DO-WHILE

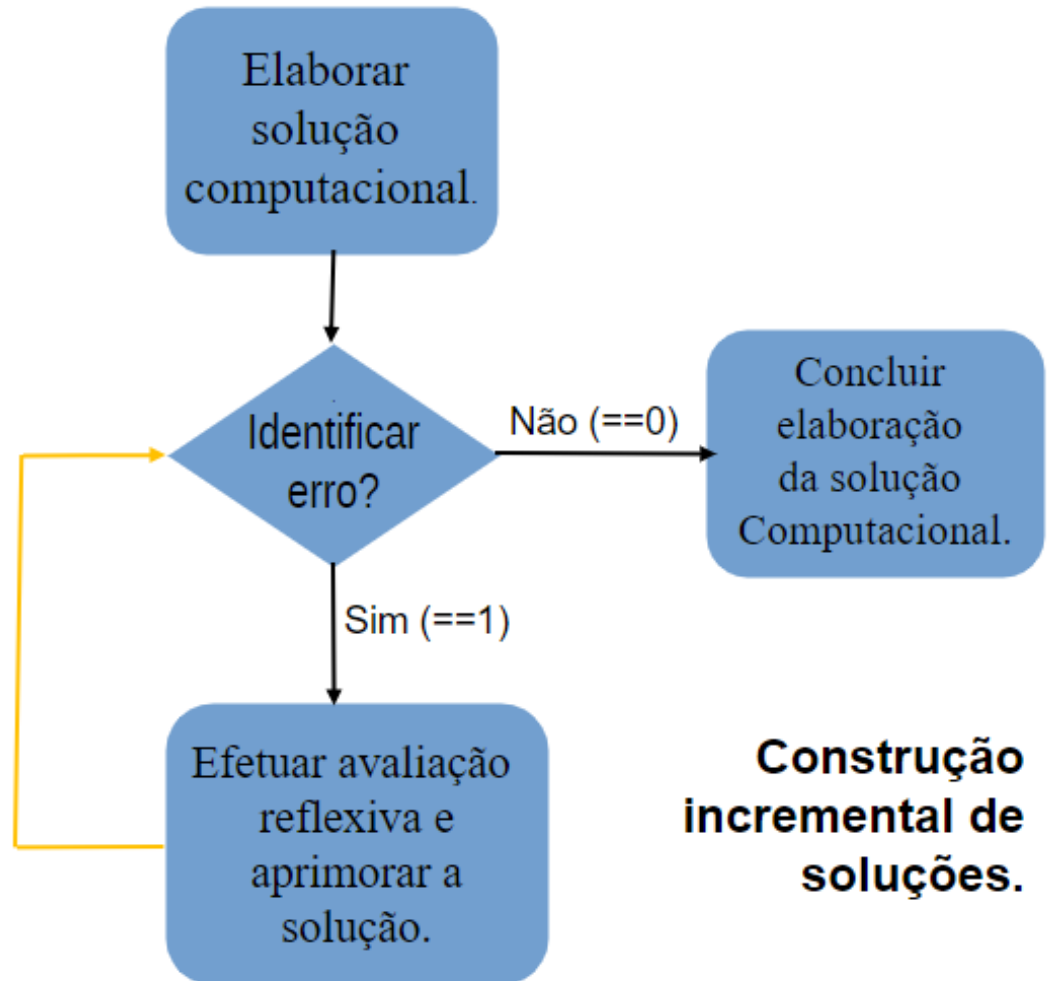
PROGRAMAÇÃO IMPERATIVA



Estruturas de Repetição

DEFINIÇÃO

Possibilitam que um comando, ou bloco de comandos, seja executado repetidas vezes.



Estruturas de Repetição

APLICAÇÃO

```
1 // Objetivo didático: testa switch-case, char, funções.
2
3 #include <stdio.h>
4 #include <ctype.h>
5
6 char menu(){
7     char Opcao;
8     printf("Qual operacao realizar? \n");
9     printf("A - Adicao \n");
10    printf("S - Subtracao \n");
11    printf("M - Multiplicacao \n");
12    printf("D - Divisao \n");
13    scanf(" %c", &Opcao);
14    return Opcao;}
15
16 void somar(float V1, float V2){
17     printf("A soma eh: %.1f", V1+V2);}
18
19 void subtrair(float V1, float V2){
20     printf("A subtração eh: %.1f", V1-V2);}
21
22 void multiplicar(float V1, float V2){
23     printf("A multiplicação eh: %.1f", V1*V2);}
24
25 void dividir(float V1, float V2){
26     if (V2!=0)
27         printf("A divisao eh: %.1f", V1/V2);
28     else
29         printf("ERRO: Tentativa de divisao por zero.");}
```

```
30
31 int main(){
32     float Num1, Num2;
33     printf("Qual o 1o valor? ");
34     scanf("%f", &Num1);
35     printf("Qual o 2o valor? ");
36     scanf("%f", &Num2);
37     char Operador = menu();
38     Operador=toupper(Operador);
39     switch (Operador){
40         case 'A': somar(Num1,Num2); break;
41         case 'S': subtrair(Num1,Num2); break;
42         case 'M': multiplicar(Num1,Num2); break;
43         case 'D': dividir(Num1,Num2); break;
44         default: printf("Opcao Invalida!!!")
45     }
46     return 0;}
```

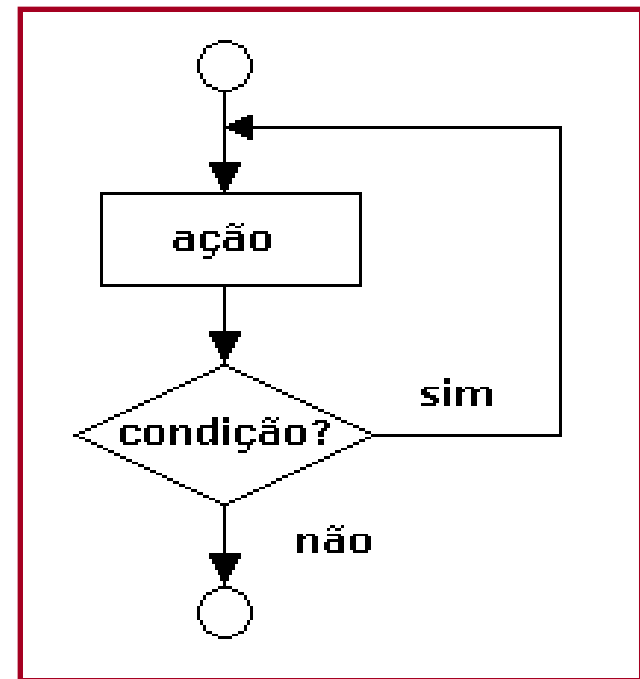
Para este funcionar para diversas entradas, faz-se necessário aplicar **estruturas de repetição**.

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO EM C

São estruturas de repetição em C:

1. `while`
2. `do - while`
3. `for`



Repetição em Fluxograma

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO `WHILE` E `DO-WHILE`

As estruturas de repetição (laço, loop ou enlaçamento) – ENQUANTO (**while**) ou FAÇA...ENQUANTO (**do...while**) – são **condicionadas** – e suas condições de parada não dependem necessariamente de um contador.

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO WHILE

```
while (condição)
    <instrução>; ...
```

Na execução deste, repete-se a <instrução> enquanto (condição) dada for verdadeira.

Em outras palavras, enquanto a (condição) for avaliada como verdadeira, a repetição se mantém.

Estruturas de Repetição

EXERCÍCIO EXEMPLO

JK tem um limite L em seu cartão de crédito e todos os meses ocorre de "estourar" L, ou seja, tentar efetuar uma compra e ser negada por falta de limite. Para evitar esta situação, JK deseja ter um programa para antes de efetuar qualquer débito D, checar se tem limite suficiente; dados L e sequencia de D. *Vale esclarecer que, no dia de compras, JK somente para de comprar quando o limite é zerado.*

```
Debito: 1500
Esta compra "estourarah" o cartao!
Limite restante: 1000.00

Debito: 300
Compra possivel!
Limite restante: 700.00

Debito: 400
Compra possivel!
Limite restante: 300.00

Debito: 500
Esta compra "estourarah" o cartao!
Limite restante: 300.00

Debito: 450
Esta compra "estourarah" o cartao!
Limite restante: 300.00

Debito: 300
Compra possivel!
Limite restante: 0.00

Seu limite de compras acabou!
-----
```

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

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

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao! \n");  
    printf("Limite restante: %.2f \n\n",*L);}
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```

```
Debito: 1500  
Esta compra "estourarah" o cartao!  
Limite restante: 1000.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 700.00
```

```
Debito: 400  
Compra possivel!  
Limite restante: 300.00
```

```
Debito: 500  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 450  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 0.00
```

```
Seu limite de compras acabou!
```

```
-----
```


Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

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

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao! \n");  
    printf("Limite restante: %.2f \n\n",*L);}
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```

```
Debito: 1500  
Esta compra "estourarah" o cartao!  
Limite restante: 1000.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 700.00
```

```
Debito: 400  
Compra possivel!  
Limite restante: 300.00
```

```
Debito: 500  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 450  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 0.00
```

```
Seu limite de compras acabou!
```



A condição deve ser avaliada antes de iniciar o loop?

Estruturas de Repetição

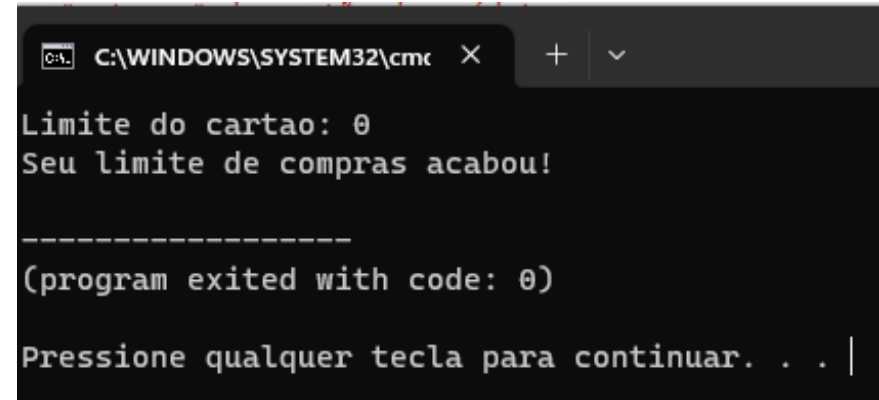
EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

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

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao! \n");  
    printf("Limite restante: %.2f \n\n",*L);}
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```



```
C:\WINDOWS\SYSTEM32\cmd X + v  
Limite do cartao: 0  
Seu limite de compras acabou!  
-----  
(program exited with code: 0)  
Pressione qualquer tecla para continuar. . . |
```

O limite de compras sendo 0, uma vez que a condição fica no início do laço, não é perguntado sobre débitos.

Estruturas de Repetição

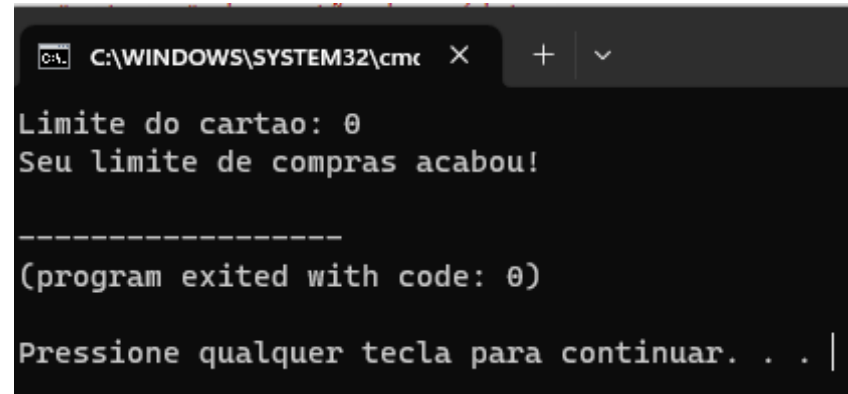
EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

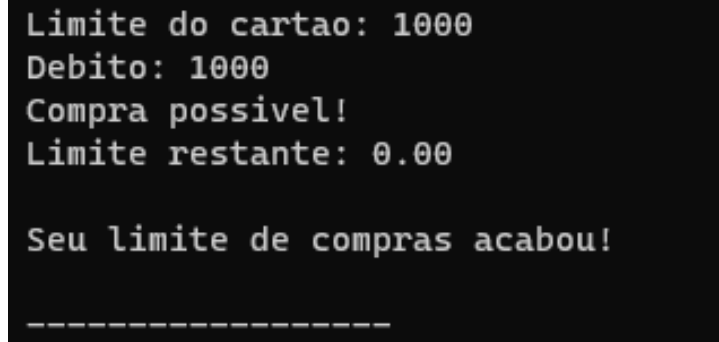
```
#include <stdio.h>
```

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao! \n");  
    printf("Limite restante: %.2f \n\n",*L);}
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```



```
C:\WINDOWS\SYSTEM32\cmd X + v  
Limite do cartao: 0  
Seu limite de compras acabou!  
-----  
(program exited with code: 0)  
Pressione qualquer tecla para continuar. . . |
```



```
Limite do cartao: 1000  
Debito: 1000  
Compra possivel!  
Limite restante: 0.00  
  
Seu limite de compras acabou!  
-----
```

A condição deve ser avaliada antes de iniciar o loop (laço, repetição) !

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito

#include <stdio.h>

void ValidaCompra(float *L) {
    float D;
    printf("Debito: ");
    scanf("%f",&D);
    if (*L - D >= 0){
        printf("Compra possivel! \n");
        *L = *L - D;}
    else
        printf("Compra \"estourarah\" o cartao! \n");
    printf("Limite restante: %.2f \n\n",*L);}

int main(){
    float Limite;
    printf("Limite do cartao: ");
    scanf("%f",&Limite);
    while (Limite>0)
        ValidaCompra(&Limite);
    printf("Seu limite de compras acabou!");
    return 0;}
```

```
Debito: 1500
Esta compra "estourarah" o cartao!
Limite restante: 1000.00

Debito: 300
Compra possivel!
Limite restante: 700.00

Debito: 400
Compra possivel!
Limite restante: 300.00

Debito: 500
Esta compra "estourarah" o cartao!
Limite restante: 300.00

Debito: 450
Esta compra "estourarah" o cartao!
Limite restante: 300.00

Debito: 300
Compra possivel!
Limite restante: 0.00

Seu limite de compras acabou!
```



Por que o parâmetro L é de saída?

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

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

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao! \n");  
    printf("Limite restante: %.2f \n\n",*L);}
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```

```
Debito: 1500  
Esta compra "estourarah" o cartao!  
Limite restante: 1000.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 700.00
```

```
Debito: 400  
Compra possivel!  
Limite restante: 300.00
```

```
Debito: 500  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 450  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 0.00
```

```
Seu limite de compras acabou!
```



Quais chaves
delimitam as instruções
subordinadas ao `while`?

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

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

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao!  
        \n");  
    printf("Limite restante: %.2f \n\n",*L);} 
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```

```
Debito: 1500  
Esta compra "estourarah" o cartao!  
Limite restante: 1000.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 700.00
```

```
Debito: 400  
Compra possivel!  
Limite restante: 300.00
```

```
Debito: 500  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 450  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 0.00
```

```
Seu limite de compras acabou!
```



Para cumprir o mesmo papel de L, poderia ser usada uma variável global?

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

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

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao! \n");  
    printf("Limite restante: %.2f \n\n",*L);}
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```

```
Debito: 1500  
Esta compra "estourarah" o cartao!  
Limite restante: 1000.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 700.00
```

```
Debito: 400  
Compra possivel!  
Limite restante: 300.00
```

```
Debito: 500  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 450  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 0.00
```

```
Seu limite de compras acabou!
```



Por que há duas *
delimitando a palavra
"estourará"?

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

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

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao! \n");  
    printf("Limite restante: %.2f \n\n",*L);}
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```

```
Debito: 1500  
Esta compra "estourarah" o cartao!  
Limite restante: 1000.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 700.00
```

```
Debito: 400  
Compra possivel!  
Limite restante: 300.00
```

```
Debito: 500  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 450  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 0.00
```

```
Seu limite de compras acabou!
```



A construção da função
ValidaCompra é justificável
/ correta, necessária?

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

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

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao! \n");  
    printf("Limite restante: %.2f \n\n",*L);}
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```

```
C:\WINDOWS\SYSTEM32\cmd  X + v  
  
Limite do cartao: 0  
Seu limite de compras acabou!  
  
-----  
(program exited with code: 0)  
  
Pressione qualquer tecla para continuar. . . |
```



Os comandos subordinados
ao `while` podem não ser
executados nem uma
única vez?

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
// JK evitar "estouro" de cartão de crédito
```

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

```
void ValidaCompra(float *L) {  
    float D;  
    printf("Debito: ");  
    scanf("%f",&D);  
    if (*L - D >= 0){  
        printf("Compra possivel! \n");  
        *L = *L - D;}  
    else  
        printf("Compra \"estourarah\" o cartao! \n");  
    printf("Limite restante: %.2f \n\n",*L);}
```

```
int main(){  
    float Limite;  
    printf("Limite do cartao: ");  
    scanf("%f",&Limite);  
    ValidaCompra(&Limite);  
    while (Limite>0)  
        ValidaCompra(&Limite);  
    printf("Seu limite de compras acabou!");  
    return 0;}
```

```
Debito: 1500  
Esta compra "estourarah" o cartao!  
Limite restante: 1000.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 700.00
```

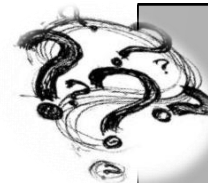
```
Debito: 400  
Compra possivel!  
Limite restante: 300.00
```

```
Debito: 500  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 450  
Esta compra "estourarah" o cartao!  
Limite restante: 300.00
```

```
Debito: 300  
Compra possivel!  
Limite restante: 0.00
```

```
Seu limite de compras acabou!
```



Por que não há ponto
e vírgula após a
condição do `while`?

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO WHILE

```
// JK evitar "estouro" de cartão de crédito

#include <stdio.h>

void ValidaCompra(float *L) {
    float D;
    printf("Debito: ");
    scanf("%f", &D);
    if (*L - D >= 0) {
        printf("Compra possivel! \n");
        *L = *L - D;
    }
    else
        printf("Compra \"estourarah\" o cartao! \n");
    printf("Limite restante: %.2f \n\n", *L);
}

int main() {
    float Limite;
    printf("Limite do cartao: ");
    scanf("%f", &Limite);
    while (Limite > 0)
        ValidaCompra(&Limite);
    printf("Seu limite de compras acabou!");
    return 0;
}
```

Quando na execução do programa, a estrutura ENQUANTO é alcançada, a condição é testada. Se o resultado for falso, os comandos subordinados a esta são ignorados, caso contrário são executados repetidamente enquanto a condição for verdadeira.

Estruturas de Repetição

EXERCÍCIO 01

```
// JK evitar "estouro" de cartão de crédito

#include <stdio.h>

void ValidaCompra(float *L) {
    float D;
    printf("Debito: ");
    scanf("%f",&D);
    if (*L - D >= 0){
        printf("Compra possivel! \n");
        *L = *L - D;}
    else
        printf("Compra \"estourarah\" o cartao! \n");
    printf("Limite restante: %.2f \n\n",*L);}

int main(){
    float Limite;
    printf("Limite do cartao: ");
    scanf("%f",&Limite);
    while (Limite>0)
        ValidaCompra(&Limite);
    printf("Seu limite de compras acabou!");
    return 0;}
```

```
Debito: 1500
Esta compra "estourarah" o cartao!
Limite restante: 1000.00

Debito: 300
Compra possivel!
Limite restante: 700.00

Debito: 400
Compra possivel!
Limite restante: 300.00

Debito: 500
Esta compra "estourarah" o cartao!
Limite restante: 300.00

Debito: 450
Esta compra "estourarah" o cartao!
Limite restante: 300.00

Debito: 300
Compra possivel!
Limite restante: 0.00

Seu limite de compras acabou!
```

Ajustar o código dado de forma que seja exibida o número da compra.

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO DO-WHILE

```
do  
    <instrução>; ...  
while (condição);
```

Na execução deste, repete-se a <instrução> enquanto (condição) dada for verdadeira.

Em outras palavras, enquanto a (condição) for avaliada como verdadeira, a repetição se mantém.

Estruturas de Repetição

EXERCÍCIO EXEMPLO

JK precisa fazer vaquinhas para alcançar valores V visando ajudar em campanhas de proteção animal. Ele precisa de um programa para somar S as diversas doações até que S seja maior ou igual a V .

```
Meta da vaquinha: 2500
Valor da doacao: 25
Valor da doacao: 500
Valor da doacao: 75
Valor da doacao: 1000
Valor da doacao: 300
Valor da doacao: 300
Valor da doacao: 300
A meta da vaquinha foi alcancada!
```

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
1 // JK faz vaquinha para proteção animal
2
3 #include <stdio.h>
4
5 int main(){
6     float Meta, Doacao, Soma = 0;
7     printf("Meta da vaquinha: ");
8     scanf("%f",&Meta);
9     do{
10         printf("Valor da doacao: ");
11         scanf("%f",&Doacao);
12         Soma = Soma + Doacao;}
13     while (Soma<Meta);
14     printf("A meta da vaquinha foi alcançada!");
15     return 0;
16 }
```

```
Meta da vaquinha: 2500
Valor da doacao: 25
Valor da doacao: 500
Valor da doacao: 75
Valor da doacao: 1000
Valor da doacao: 300
Valor da doacao: 300
Valor da doacao: 300
A meta da vaquinha foi alcançada!
```

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
1 // JK faz vaquinha para proteção animal
2
3 #include <stdio.h>
4
5 int main(){
6     float Meta, Doacao, Soma = 0;
7     printf("Meta da vaquinha: ");
8     scanf("%f",&Meta);
9     do{
10         printf("Valor da doacao: ");
11         scanf("%f",&Doacao);
12         Soma = Soma + Doacao;}
13 while (Soma<Meta);
14 printf("A meta da vaquinha foi alcançada!");
15 return 0;
16 }
```

```
Meta da vaquinha: 2500
Valor da doacao: 25
Valor da doacao: 500
Valor da doacao: 75
Valor da doacao: 1000
Valor da doacao: 300
Valor da doacao: 300
Valor da doacao: 300
A meta da vaquinha foi alcançada!
-----
```



A condição deve ser avaliada
antes de iniciar o loop?

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
1 // JK faz vaquinha para proteção animal
2
3 #include <stdio.h>
4
5 int main(){
6     float Meta, Doacao, Soma = 0;
7     printf("Meta da vaquinha: ");
8     scanf("%f",&Meta);
9     do{
10         printf("Valor da doacao: ");
11         scanf("%f",&Doacao);
12         Soma = Soma + Doacao;}
13     while (Soma<Meta);
14     printf("A meta da vaquinha foi alcançada!");
15     return 0;
16 }
```

```
Meta da vaquinha: 2500
Valor da doacao: 25
Valor da doacao: 500
Valor da doacao: 75
Valor da doacao: 1000
Valor da doacao: 300
Valor da doacao: 300
Valor da doacao: 300
A meta da vaquinha foi alcançada!
-----
```



A condição deve ser avaliada antes de iniciar o loop?

Em termos lógicos, só faz sentido avaliar se a meta já foi alcançada a partir da ocorrência da 1a doação.

Estruturas de Repetição

EXERCÍCIO EXEMPLO

```
1 // JK faz vaquinha para proteção animal
2
3 #include <stdio.h>
4
5 int main(){
6     float Meta, Doacao, Soma = 0;
7     printf("Meta da vaquinha: ");
8     scanf("%f",&Meta);
9     do{
10         printf("Valor da doacao: ");
11         scanf("%f",&Doacao);
12         Soma = Soma + Doacao;}
13 while (Soma<Meta);
14 printf("A meta da vaquinha foi alcançada!");
15 return 0;
16 }
```

```
Meta da vaquinha: 2500
Valor da doacao: 25
Valor da doacao: 500
Valor da doacao: 75
Valor da doacao: 1000
Valor da doacao: 300
Valor da doacao: 300
Valor da doacao: 300
A meta da vaquinha foi alcançada!
-----
```



A variável Soma deve ser inicializada?

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO DO-WHILE

Caso a lógica dos comandos não provoque a alteração do estado da condição, ocorrerá um loop infinito.

do

<instrução>...

while (condição);

```
1 // JK faz vaquinha para proteção animal
2
3 #include <stdio.h>
4
5 int main(){
6     float Meta, Doacao, Soma = 0;
7     printf("Meta da vaquinha: ");
8     scanf("%f",&Meta);
9     do{
10         printf("Valor da doacao: ");
11         scanf("%f",&Doacao);
12         Soma = Soma + Doacao;}
13     while (Soma<Meta);
14     printf("A meta da vaquinha foi alcançada!");
15     return 0;
16 }
```



Qual instrução garante que o loop não será infinito?

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO DO-WHILE

Caso a lógica dos comandos não provoque a alteração do estado da condição, ocorrerá um loop infinito.

do

<instrução>...

while (condição);

```
1 // JK faz vaquinha para proteção animal
2
3 #include <stdio.h>
4
5 int main(){
6     float Meta, Doacao, Soma = 0;
7     printf("Meta da vaquinha: ");
8     scanf("%f",&Meta);
9     do{
10         printf("Valor da doacao: ");
11         scanf("%f",&Doacao);
12         Soma = Soma + Doacao;}
13     while (Soma<Meta);
14     printf("A meta da vaquinha foi alcancada!");
15     return 0;
16 }
```



As instruções subordinadas ao do-while podem não ser executadas nenhuma vez?

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO DO-WHILE

Os comandos subordinados ao DO...WHILE são executados pelo menos uma vez, já que a condição (de parada) somente é testada no final da estrutura.

do

<instrução>...

while (condição);

```
1 // JK faz vaquinha para proteção animal
2
3 #include <stdio.h>
4
5 int main(){
6     float Meta, Doacao, Soma = 0;
7     printf("Meta da vaquinha: ");
8     scanf("%f",&Meta);
9     do{
10         printf("Valor da doacao: ");
11         scanf("%f",&Doacao);
12         Soma = Soma + Doacao;}
13     while (Soma<Meta);
14     printf("A meta da vaquinha foi alcançada!");
15     return 0;
16 }
```

Estruturas de Repetição

EXERCÍCIO 02

Ajustar o código dado de forma que seja identificado também quando a meta da vaquinha foi ultrapassada, e para, sempre que houver uma doação, exibir o valor parcial alcançado.

```
1 // JK faz vaquinha para proteção animal
2
3 #include <stdio.h>
4
5 int main(){
6     float Meta, Doacao, Soma = 0;
7     printf("Meta da vaquinha: ");
8     scanf("%f",&Meta);
9     do{
10         printf("Valor da doacao: ");
11         scanf("%f",&Doacao);
12         Soma = Soma + Doacao;}
13     while (Soma<Meta);
14     printf("A meta da vaquinha foi alcancada!");
15     return 0;
16 }
```

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO DO-WHILE



Qual o objetivo alcançado a partir desse trecho de código?

...

do

```
{  
    printf("Digite um valor: ");  
    scanf("%d", &N1);  
    printf("Digite outro valor: ");  
    scanf("%d", &N2);  
    printf("A soma eh: %d. \n", N1+N2);  
    printf("Outra soma? 1- Sim 2 - Nao");  
    scanf("%d", &Opcao);  
}
```

while _____;

...

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO DO-WHILE

...

do

```
{  
    printf("Digite um valor: ");  
    scanf("%d", &N1);  
    printf("Digite outro valor: ");  
    scanf("%d", &N2);  
    printf("A soma eh: %d. \n", N1+N2);  
    printf("Outra soma? 1- Sim 2 - Nao");  
    scanf("%d", &Opcao);  
}
```

while _____;

...



Como fica o **while** para garantir iterações e evitar loops infinitos?

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO DO-WHILE

...

do

```
{  
    printf("Digite um valor: ");  
    scanf("%d", &N1);  
    printf("Digite outro valor: ");  
    scanf("%d", &N2);  
    printf("A soma eh: %d. \n", N1+N2);  
    printf("Outra soma? 1- Sim 2 - Nao");  
    scanf("%d", &Opcao);  
}
```

while (Opcao==1) ;

...

Quais instruções são repetidas na execução desse?



Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO DO-WHILE

...

do

{

```
printf("Digite um valor: ");
```

```
scanf("%d", &N1);
```

```
printf("Digite outro valor: ");
```

```
scanf("%d", &N2);
```

```
printf("A soma eh: %d. \n", N1+N2);
```

```
printf("Outra soma? 1- Sim 2 - Nao");
```

```
scanf("%d", &Opcao);
```

}

while (Opcao==1) ;

...



Que linha(s)
garante(m) que não
haverá loop infinito ?

Estruturas de Repetição

EXERCÍCIO EXEMPLO

Escrever programa para ler um número inteiro qualquer, e exibir o sucessor deste. Em seguida, perguntar ao usuário se desejar continuar. Em caso afirmativo (S), apresentar o sucessor do número exibido. E novamente perguntar o desejo do usuário, e continuar até que ele responda não (N).

Estruturas de Repetição

EXERCÍCIO EXEMPLO

Sucessor! Correto? Escrever programa para ler um número inteiro qualquer, e exibir o sucessor deste. Em seguida, perguntar ao usuário se desejar continuar. Em caso afirmativo (S), apresentar o sucessor do número exibido. E novamente perguntar o desejo do usuário, e continuar até que ele responda não (N).

```
leia(N) ;  
faça  
    N ← N+1 ;  
    escreva('Sucessor: ', N) ;  
    escreva('Outra vez? S|N') ;  
    leia(Resposta) ;  
enquanto Resposta = 'S' ;
```

Estruturas de Repetição

EXERCÍCIO EXEMPLO

Sucessor! Também correto?

Escrever programa para ler um número inteiro qualquer, e exibir o sucessor deste. Em seguida, perguntar ao usuário se deseja continuar. Em caso afirmativo (S), apresentar o sucessor do número exibido. E novamente perguntar o desejo do usuário, e continuar até que ele responda não (N).

```
leia(N);  
faça  
    escreva('Sucessor: ', N+1);  
    escreva('Outra vez? S|N');  
    leia(Resposta);  
enquanto Resposta='S';
```

```

int main(){
    int Num1, Num2;
    char Operador, Resposta;
    do
    { system("cls");
      printf("Quais os numeros? ");
      scanf("%d%d", &Num1, &Num2);
      printf("Qual operacao realizar? \n");
      printf("A - Adicao \n");
      printf("S - Subtracao \n");
      printf("M - Multiplicacao \n");
      printf("D - Divisao \n");
      scanf(" %c", &Operador);
      switch (Operador){
          case 'A': printf("A soma eh: %d.",Num1+Num2); break;
          case 'S': printf("A subtracao eh: %d.", Num1-Num2); break;
          case 'M': printf("A multiplicacao eh: %d.", Num1*Num2); break;
          case 'D': if (Num2!=0)
                      printf("A divisao eh: %f.",(float)Num1/Num2);
                      else
                          printf("Tentativa de divisao por zero."); break;
          default:
              printf("\n Opcao Invalida!!!");}
      printf("\n Continuar? S/N");
      scanf(" %c",&Resposta);}
    while (Resposta == 'S' || Resposta=='s');
    return 0;}

```

Retomando...



E para funcionar
para diversas
entradas?

```

int main(){
    int Num1, Num2;
    char Operador, Resposta;
    do
    { system("cls");
      printf("Quais os numeros? ");
      scanf("%d%d", &Num1, &Num2);
      printf("Qual operacao realizar? \n");
      printf("A - Adicao \n");
      printf("S - Subtracao \n");
      printf("M - Multiplicacao \n");
      printf("D - Divisao \n");
      scanf(" %c", &Operador);
      switch (Operador){
          case 'A': printf("A soma eh: %d.",Num1+Num2); break;
          case 'S': printf("A subtracao eh: %d.", Num1-Num2); break;
          case 'M': printf("A multiplicacao eh: %d.", Num1*Num2); break;
          case 'D': if (Num2!=0)
                      printf("A divisao eh: %f.",(float)Num1/Num2);
                      else
                          printf("Tentativa de divisao por zero."); break;
          default:
              printf("\n Opcao Invalida!!!");}
      printf("\n Continuar? S/N");
      scanf(" %c",&Resposta);}
    while (Resposta == 'S' || Resposta=='s');
    return 0;}

```

Retomando...



Qual o papel do
system(cls);?

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO E `break`

```
1 //o desafio eh digitar o máximo de múltiplos de 7
2
3 #include <stdio.h>
4
5 int main(){
6     int N, Quant = 0;
7     do{
8         scanf("%d",&N);
9         if (N % 7 == 0){
10             Quant++;
11             printf("Voce digitou %d valores divisiveis por 7! \n",Quant);}
12         else
13             break;}
14     while (1);
15     printf("Voce digitou %d, nao divisivel por 7!",N);
16     return 0;}
```

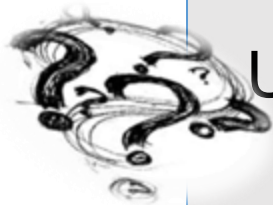


Qual o objetivo
deste?

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO E `break`

```
1 //o desafio eh digitar o máximo de múltiplos de 7
2
3 #include <stdio.h>
4
5 int main(){
6     int N, Quant = 0;
7     do{
8         scanf("%d",&N);
9         if (N % 7 == 0){
10             Quant++;
11             printf("Voce digitou %d valores divisiveis por 7! \n",Quant);}
12         else
13             break;}
14     while (1);
15     printf("Voce digitou %d, nao divisivel por 7!",N);
16     return 0;}
```



Uso do `do-while`
adequado?

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO E `break`

```
1 //o desafio eh digitar o máximo de múltiplos de 7
2
3 #include <stdio.h>
4
5 int main(){
6     int N, Quant = 0;
7     do{
8         scanf("%d",&N);
9         if (N % 7 == 0){
10             Quant++;
11             printf("Voce digitou %d valores divisiveis por 7! \n",Quant);}
12         else
13             break;}
14     while (1);
15     printf("Voce digitou %d, nao divisivel por 7!",N);
16     return 0;}
```

Provoca a quebra da execução de repetição ou conjunto de comandos. Efetuada essa quebra, a execução continua a partir da primeira linha seguinte ao loop.

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO E `break`

```
1 //o desafio eh digitar o máximo de múltiplos de 7
2
3 #include <stdio.h>
4
5 int main(){
6     int N, Quant = 0;
7     do{
8         scanf("%d",&N);
9         if (N % 7 == 0){
10             Quant++;
11             printf("Voce digitou %d valores divisiveis por 7!", Quant);
12         }
13         else
14             break;}
15 while (1);
16 printf("Voce digitou %d, nao divisivel por 7!", N);
17 return 0;}
```

7
Voce digitou 1 valores divisiveis por 7!
70
Voce digitou 2 valores divisiveis por 7!
700
Voce digitou 3 valores divisiveis por 7!
14
Voce digitou 4 valores divisiveis por 7!
21
Voce digitou 5 valores divisiveis por 7!
28
Voce digitou 6 valores divisiveis por 7!
35
Voce digitou 7 valores divisiveis por 7!
49
Voce digitou 8 valores divisiveis por 7!
60
Voce digitou 60, nao divisivel por 7!

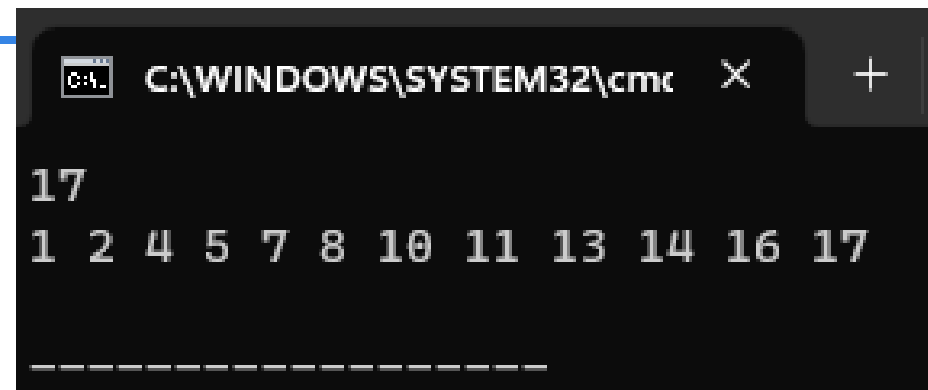
Observar o uso do loop
"aparentemente" infinito.

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO E `continue`

Termina a execução do passo da repetição corrente, mas continua na repetição, no próximo passo; ou seja, apenas os comandos restantes são ignorados.

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



```
C:\WINDOWS\SYSTEM32\cmd
17
1 2 4 5 7 8 10 11 13 14 16 17
-----
```

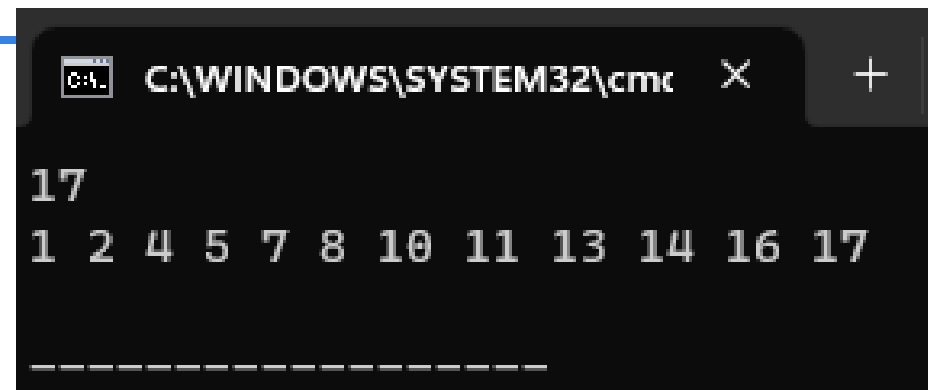
Exibe os números de 1 a N, sendo N dado pelo usuário, com exceção dos múltiplos de 3.

Estruturas de Repetição

INSTRUÇÕES DE REPETIÇÃO E `continue`

Termina a execução do passo da repetição corrente, mas continua na repetição, no próximo passo; ou seja, apenas os comandos restantes são ignorados.

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



```
C:\WINDOWS\SYSTEM32\cmd
17
1 2 4 5 7 8 10 11 13 14 16 17
-----
```



Uso do `while`
adequado?

Estruturas de Repetição

EXERCÍCIO 03

Ler notas de alunos e identificar se cada foi aprovado. Considerar como sistema de avaliação: em princípio, 3 notas; são aprovados os alunos com média 7; para os alunos com média acima de 3, é possibilitado fazer 4ª avaliação, então a média passa a ser 5 para obtenção da aprovação. Após a identificação de que um aluno foi aprovado, ou não; deve ser questionado se o usuário (do programa) deseja verificar a situação de outro aluno.

Estruturas de Repetição

EXERCÍCIO 04

Escrever programa , sem aplicar resto de divisão (%) e valor do quociente (/), para calcular a quantidade de meses contidos numa dada quantidade de dias. Considerar que todos os meses são compostos por 30 dias e desprezar o montante de dias que não formam pelo menos um mês completo (resto).

Carla é uma profissional muito dedicada! Ela é responsável por analisar o pH de várias substâncias e determinar se elas são ácidas, básicas ou neutras. Ela não para enquanto não tiver terminado de analisar todas as soluções pendentes.

Escreva um programa para ajudar a nossa querida Carla no seu trabalho. O programa vai receber como entrada uma sequência de números, cada um em uma linha, representando o pH de cada solução. A última entrada vai ser o número -1, indicando que não há mais soluções para serem analisadas e o programa pode encerrar sua execução.

Para cada solução, o programa vai determinar a sua acidez: ACIDA (pH menor que 7), BASICA (pH maior que 7), ou NEUTRA (pH igual a 7).

E aí, você vai ajudar a Carla? Bom trabalho!

Formato de entrada


A entrada é composta por diferentes números, cada um em uma linha. O último número sempre será -1.0..

Formato de saída

A saída terá as palavras ACIDA, BASICA e/ou NEUTRA, escritas em maiúsculas e sem acentos.

Note que quando a entrada for o número -1, nada a mais deve ser impresso na tela

Exemplos de:

 Entrada	 Saída
1	ACIDA
1	ACIDA
1	ACIDA
2	ACIDA
1	ACIDA
3	ACIDA
-1	

Fonte: The Huxley, <<https://www.thehuxley.com/>>

O IBGE realizou um concurso para contratar pessoas para trabalhar no censo. Cada candidato fez uma prova de português com 50 questões, outra de matemática com 35 questões, e uma prova de redação.

Para ser aprovado, era necessário acertar pelo menos 80% da prova de português, 60% da prova de matemática, e ter nota igual ou superior a 7 na redação.

Escreva um programa que receba como entrada, para cada candidato, a quantidade de questões certas em português e em matemática, e também a nota na redação, e depois exiba quantos candidatos foram aprovados.

Formato de entrada

Dois números inteiros seguidos por um número real para cada candidato

A entrada deve encerrar quando a quantidade de questões de português informada for inferior a zero

Formato de saída

Um número inteiro

Exemplos de:

 Entrada 	 Saída 
42 27 8.3 44 20 7.5 -3	1

Fonte: The Huxley, <<https://www.thehuxley.com/>>

O IBGE realizou um concurso para contratar pessoas para trabalhar no censo. Cada candidato fez uma prova de português com 50 questões, outra de matemática com 35 questões, e uma prova de redação.

Para ser aprovado, era necessário acertar pelo menos 80% da prova de português, 60% da prova de matemática, e ter nota igual ou superior a 7 na redação.

Escreva um programa que receba como entrada, para cada candidato, a quantidade de questões certas em português e em matemática, e também a nota na redação, e depois exiba quantos candidatos foram aprovados.

Formato de entrada



Dois números inteiros seguidos por um número real para cada candidato

A entrada deve encerrar quando a quantidade de questões de português informada for inferior a zero

Formato de saída

Um número inteiro

Exemplos de:

 Entrada	
42 27 8.3 44 20 7.5 -3	1

Escrever programa com saída mais elegante. Por exemplo, perguntar ao usuário se desejar sair, oferecendo as opções: S – Sim ou N - Não.

Fonte: Adaptado do The Huxley, <<https://www.thehuxley.com/>>

Estruturas de Repetição

EXERCÍCIO 08

Zelda e seus amigos tiveram uma brilhante ideia durante as aulas da monitoria da cadeira de introdução a programação: que tal fazer um programa que, dado um número n ($1 \leq n \leq 40$) imprima na tela os números de 1 até o número da iteração atual, sendo que serão feitas n iterações, como demonstrado no exemplo a seguir:

Supondo para $n = 5$:

(primeira iteração): 1
(segunda iteração): 1 2
(terceira iteração): 1 2 3
(quarta iteração): 1 2 3 4
(quinta iteração): 1 2 3 4 5

Formato de entrada

Um inteiro n ($1 \leq n \leq 40$)

Formato de saída

A sequência (1 ... M), onde M é o número da iteração atual do laço, que será executada n vezes.

Exemplos de:

 Entrada 	 Saída 
1	1
 Entrada 	 Saída 
10	1 1 2 1 2 3 1 2 3 4 1 2 3 4 5 1 2 3 4 5 6 1 2 3 4 5 6 7 1 2 3 4 5 6 7 8 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 10

Bernardo é um garoto que adora números e curiosidades matemáticas. Certo dia, enquanto tomava banho (ocasião essa que costuma ser propícia para ter boas ideias), lhe ocorreu uma epifania matemática: todo número ímpar é a diferença de dois quadrados consecutivos.

Terminada sua sessão de higiene, Bernardo começou a rabiscar suas ideias, observando as propriedades interessantes que os números carregavam:

$$1 - 0 = 1$$

$$4 - 1 = 3$$

$$9 - 4 = 5$$

$$16 - 9 = 7$$

$$25 - 16 = 9$$

Tendo se divertido o suficiente com sua descoberta, desafiou seu amigo estudante de Engenharia da Computação a fazer um programa que escrevesse um número ímpar como a diferença de dois quadrados, para que ele nunca mais se esquecesse da ideia que teve.





Formato de entrada

A entrada consiste de vários casos de teste. Em cada linha teremos um inteiro ímpar x ($1 \leq x \leq 10000$), e na última linha o número 0.

Formato de saída

Para cada inteiro dado como entrada, mostrar este escrito como diferença de dois quadrados. no formato dado na saída

Exemplos de:

 Entrada 	 Saída 
3 5 571 0	4 - 1 9 - 4 81796 - 81225

Fonte: The Huxley, <<https://www.thehuxley.com/>>

```
#include<stdio.h>
int main()
{
    int n;
    do
    {
        printf("Digite um numero ou zero para sair: ");
        scanf("%d", &n);

        if( n%2 == 1 )
            printf("%d é ímpar\n", n);
        else
            printf("%d é par\n", n);

    }while( n != 0 );
    return 0;
}
```

Qual o objetivo alcançado a partir da execução do código dado ao lado?

Programação Imperativa

COMPLEMENTAR AULA...

Fundamentos da Programação de Computadores

Ana Fernanda Gomes Ascencio
Edilene Aparecida Veneruchi de Campos

Capítulos

Comandos de Repetição



Programação Imperativa

COMPLEMENTAR AULA...

Curso de Linguagem C
UFMG

*linux.ime.usp.br/~lucas
mmg/livecd/documenta
cao/documentos/curso_
de_c/www.ppgia.pucpr.
br/_maziero/ensino/so/
projetos/curso-c/c.html*

Aula 4

Estruturas de
Controle de Fluxo

[Aula 1: Introdução e Sumário](#)

[Aula 2 - Primeiros Passos](#)

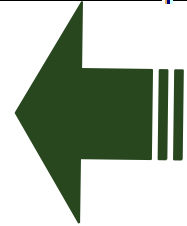
[Aula 3 - Variáveis, Constantes, Operadores e Expressões](#)

[Aula 4 - Estruturas de Controle de Fluxo](#)

[Aula 5 - Matrizes e Strings](#)

[Aula 6 - Ponteiros](#)

[Aula 7 - Funções](#)



Programação Imperativa

PRÓXIMO PASSO



Estruturas de Repetição FOR