

Algoritmos e Estruturas de Dados I

# Funções

Pedro Olmo Stancioli Vaz de Melo

# Exercício

- Brutus e Olívia foram ao médico, que disse a eles que ambos estão fora do peso ideal. Ambos discordaram veementemente da afirmação do médico. Para provar que estava certo, o médico mostrou o Índice de Massa Corporal (IMC) de ambos, considerando que Brutus tem 1,84m e pesa 112kg e Olívia tem 1,76m e pesa 49kg. Implemente um programa para mostrar o IMC de Brutus e Olívia e quantos kilos Brutus e Olívia devem perder/ganhar para atingirem um peso saudável segundo a classificação do IMC.

# Exercício

IMC	Classificação
$< 16$	Magreza grave
16 a $< 17$	Magreza moderada
17 a $< 18.5$	Magreza leve
18.5 a $< 25$	Saudável
25 a $< 30$	Sobrepeso
30 a $< 35$	Obesidade grau I
35 a $< 40$	Obesidade grau II (severa)
$\geq 40$	Obesidade grau III (mórbida)

Table 3.1: Classificação do IMC

# Exercício

```
1  #include <stdio.h>
2
3  void main() {
4      float pesoOlivia = 45, alturaOlivia = 1.76;
5      float pesoBrutus = 122, alturaBrutus = 1.84;
6      float pesoIdealOlivia = 18.5 * (alturaOlivia*alturaOlivia);
7      float pesoIdealBrutus = 25 * (alturaBrutus*alturaBrutus);
8
9      printf("Olivia deve ganhar %.2f Kg.\n", pesoIdealOlivia-pesoOlivia);
10     printf("Olivia deve perder %.2f Kg.\n", pesoBrutus-pesoIdealBrutus);
11
12 }
```

# Exercício

Uma conta poupança foi aberta com um depósito de R\$500,00, com rendimentos 1% de juros ao mês. No segundo mês, R\$200,00 reais foram depositados nessa conta poupança. No terceiro mês, R\$50,00 reais foram retirados da conta. Quanto haverá nessa conta no quarto mês?

# Exercício

- Pensando no problema...
  - Preciso de armazenar o valor dessa conta
  - O valor será acrescido todo mês de 1%
  - Vou adicionar 200 reais a esse valor no segundo mês
  - Vou retirar 50 reais desse valor no terceiro

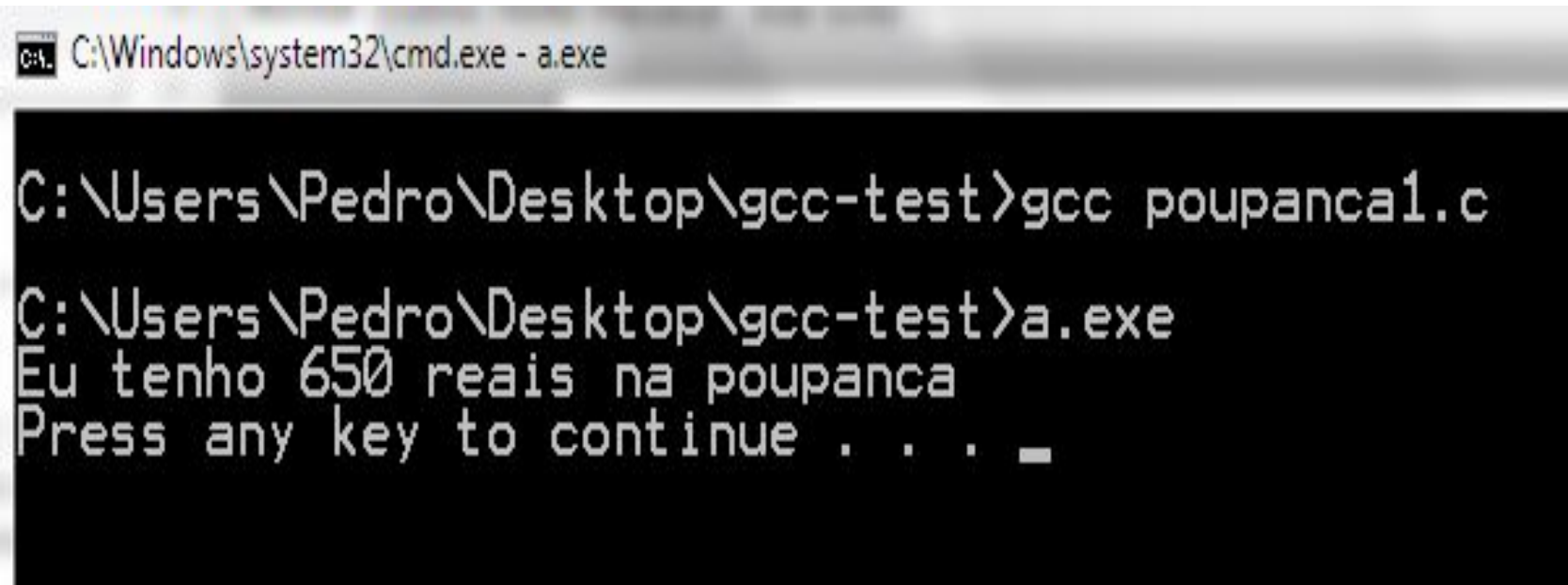
# Solução 1

```
#include <stdio.h>

void main(void) {
    int poupanca;
    //saldo inicial
    poupanca = 500;
    //rendimento apos o primeiro mes
    poupanca = poupanca + poupanca * (1/100);
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca + poupanca * (1/100);
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca + poupanca * (1/100);

    printf("Eu tenho %d reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```

# Solução 1



```
CA. C:\Windows\system32\cmd.exe - a.exe

C:\Users\Pedro\Desktop\gcc-test>gcc poupanca1.c

C:\Users\Pedro\Desktop\gcc-test>a.exe
Eu tenho 650 reais na poupanca
Press any key to continue . . . _
```



# Solução 1

```
#include <stdio.h>

void main(void) {
    int poupanca;
    //saldo inicial
    poupanca = 500;
    //rendimento apos o primeiro mes
    poupanca = poupanca + poupanca * (1/100);
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca + poupanca * (1/100);
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca + poupanca * (1/100);

    printf("Eu tenho %d reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```

# Solução 2

```
#include <stdio.h>

void main(void) {
    float poupanca;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca + poupanca * (1.0/100);
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca + poupanca * (1.0/100);
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca + poupanca * (1.0/100);

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```

# Solução 2

```
#include <stdio.h>

void main(void) {
    float poupanca;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca + poupanca * (1.0/100);
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca + poupanca * (1.0/100);
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca + poupanca * (1.0/100);

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```



# Solução 2

C:\Windows\system32\cmd.exe - a.exe

```
C:\Users\Pedro\Desktop\gcc-test>gcc poupanca2.c
```

```
C:\Users\Pedro\Desktop\gcc-test>a.exe  
Eu tenho 668.670471 reais na poupanca  
Press any key to continue . . .
```

# Tem como melhorar?

```
#include <stdio.h>

void main(void) {
    float poupanca;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca + poupanca * (1.0/100);
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca + poupanca * (1.0/100);
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca + poupanca * (1.0/100);

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```

# Tem como melhorar?

Se o rendimento da poupança mudar de 1% para 2%?

```
#include <stdio.h>

void main(void) {
    float poupanca;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca + poupanca * (1.0/100);
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca + poupanca * (1.0/100);
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca + poupanca * (1.0/100);

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```

# Solução 3

```
#include <stdio.h>

void main(void) {
    float poupanca;
    float juros = 1.0/100;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca + poupanca * juros;
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca + poupanca * juros;
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca + poupanca * juros;

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```



# Solução 3

```
#include <stdio.h>

void main(void) {
    float poupanca;
    float juros = 1.0/100;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca + poupanca * juros;
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca + poupanca * juros;
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca + poupanca * juros;

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```



# Tem como melhorar?

```
#include <stdio.h>

void main(void) {
    float poupanca;
    float juros = 1.0/100;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca + poupanca * juros;
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca + poupanca * juros;
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca + poupanca * juros;

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```

# Tem como melhorar?

- Um pouco de matemática
  - $\text{poupanca} = \text{poupanca} + \text{poupanca} * \text{juros};$
  - $\text{poupanca} = \text{poupanca} * (1 + \text{juros});$
  - //como  $\text{juros} == 0.01$ , então:
  - $\text{poupanca} = \text{poupanca} * 1.01$

# Solução 4

```
#include <stdio.h>

void main(void) {
    float poupanca;
    float juros = 1.01;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca * juros;
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca * juros;
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca * juros;

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```

# Solução 4

```
#include <stdio.h>

void main(void) {
    float poupanca;
    float juros = 1.01;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca * juros;
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca * juros;
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca * juros;

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```



# Tem como melhorar?

```
#include <stdio.h>

void main(void) {
    float poupanca;
    float juros = 1.01;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = poupanca * juros;
    //deposito de 200 reais no segundo mes
    poupanca = poupanca + 200;
    //rendimento apos o segundo mes
    poupanca = poupanca * juros;
    //retirada de 50 reais no terceiro mes
    poupanca = poupanca - 50;
    //rendimento apos o terceiro mes
    poupanca = poupanca * juros;

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```

# Tem como melhorar?

- E se o banco definir que junto com o rendimento mensal, R\$3,14 reais devem ser retirados da conta como taxa de manutenção?

# Solução 5

- E se o banco definir que junto com o rendimento mensal, R\$3,14 reais devem ser retirados da conta como taxa de manutenção?
- Podemos criar uma **função** que centraliza as operações mensais padrões do banco

# Solução 5

```
void main(void) {  
    float poupanca;  
    //saldo inicial  
    poupanca = 500.0;  
    //rendimento apos o primeiro mes  
    poupanca = rendePoupanca(poupanca);  
    //deposito de 200 reais no segundo mes  
    poupanca = poupanca + 200;  
    //rendimento apos o segundo mes  
    poupanca = rendePoupanca(poupanca);  
    //retirada de 50 reais no terceiro mes  
    poupanca = poupanca - 50;  
    //rendimento apos o terceiro mes  
    poupanca = rendePoupanca(poupanca);  
  
    printf("Eu tenho %f reais na poupanca", poupanca);  
    printf("\n");  
    system("PAUSE");  
}
```



# Solução 5

```
void main(void) {  
    float poupanca;  
    //saldo inicial  
    poupanca = 500.0;  
    //rendimento apos o primeiro mes  
    poupanca = rendePoupanca(poupanca);  
    //deposito de 200 reais no segundo mes  
    poupanca = poupanca + 200;  
    //rendimento apos o segundo mes  
    poupanca = rendePoupanca(poupanca);  
    //retirada de 50 reais no terceiro mes  
    poupanca = poupanca - 50;  
    //rendimento apos o terceiro mes  
    poupanca = rendePoupanca(poupanca);  
  
    printf("Eu tenho %f reais na poupanca", poupanca);  
    printf("\n");  
    system("PAUSE");  
}
```

# Solução 5

- A função **rendePoupanca** é encarregada de realizar todas as operações de rendimento mensais da conta poupança

```
void main(void) {  
    float poupanca;  
    //saldo inicial  
    poupanca = 500.0;  
    //rendimento apos o primeiro mes  
    poupanca = rendePoupanca(poupanca);  
    //deposito de 200 reais no segundo mes  
    poupanca = poupanca + 200;  
    //rendimento apos o segundo mes  
    poupanca = rendePoupanca(poupanca);  
    //retirada de 50 reais no terceiro mes  
    poupanca = poupanca - 50;  
    //rendimento apos o terceiro mes  
    poupanca = rendePoupanca(poupanca);  
  
    printf("Eu tenho %f reais na poupanca", poupanca);  
    printf("\n");  
    system("PAUSE");  
}
```

# Solução 5

```
float rendePoupanca(float poupanca) {  
    float juros = 1.01;  
    float taxa = 3.14;  
    return poupanca*juros-taxa;  
}
```

# Solução 5

tipo do retorno da função

parâmetro da função

```
float rendePoupanca(float poupanca) {  
    float juros = 1.01;  
    float taxa = 3.14;  
    return poupanca*juros-taxa;  
}
```

retorno da função

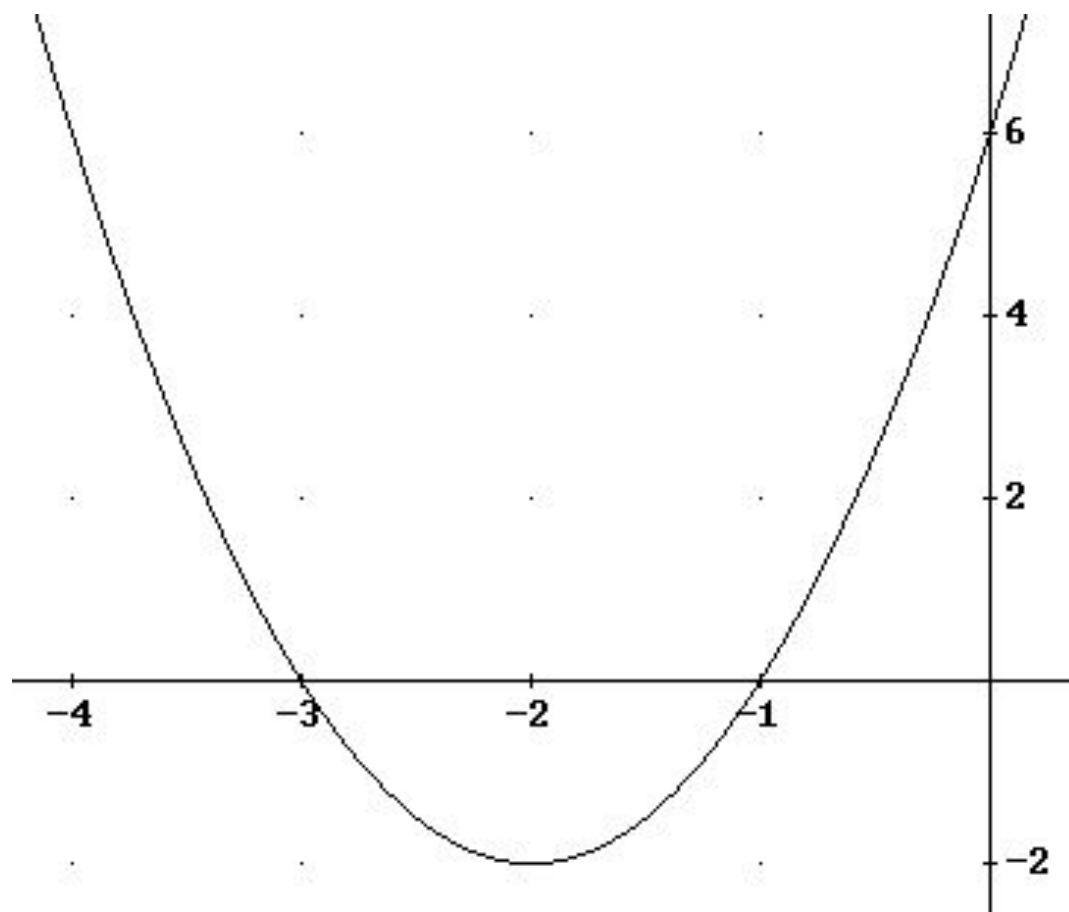
# Funções - revisão

---

- Funções definem operações que são usadas frequentemente
- Funções, na matemática, requerem parâmetros de entrada, e definem um valor de saída

# Funções - Exemplos

- Função quadrática  $y = ax^2 + bx + c$ 
  - Entrada:  $x$
  - Saída:  $y$



# Funções na programação

---

- Em linguagens imperativas, TODOS os programas usam funções
- No C, o programa SEMPRE começa executando a função **main**.

```
#include <stdio.h>

int main(void)
{
    printf("Olá, Mundo!");
    return 0;
}
```

# Funções

---

- Usamos funções para evitar de escrever várias vezes o mesmo código
  - Código que será executado várias vezes em um programa, mas com valores diferentes
- Operações comuns a um ou mais programas



# Funções - C

---

- Em C, definimos a função por:
  - Zero ou mais parâmetros de entrada, com os seus tipos
  - Um parâmetro de saída com tipo, sendo que o tipo pode ser “sem saída” (**void**)
  - Código da função
- Ao chamarmos a função, devemos passar valores para TODOS os parâmetros, sem exceção (não é o caso no C++...)

# C - Exemplo

---

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x));  
3. }
```

# C - Exemplo

---

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x));  
3. }
```

**Nome da  
função**



# C - Exemplo

---

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x));  
3. }
```

**Parâmetro de entrada da  
função**

# C - Exemplo

---

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x));  
3. }
```

**Tipo de saída da  
função**

# C - Exemplo

---

```
1. double logistica(double x) {  
2.     return 1.0 / (1.0 + exp(-1.0 * x)) ;  
3. }
```

**código da  
função**



## C – Exemplo: usando funções

---

```
1. double logistica(double x) {  
2.     return 1.0/(1.0+exp(-1.0*x));  
3. }  
4.  
5. int main() {  
6.     double entrada = 10.0;  
7.     double saida = logistica(entrada);  
8.     printf("%f", saida);  
9.     return 0;  
10. }
```



**Variável saida recebe o valor da função**

## Exemplo 2

```
int sep (int v[], int p, int r) {
    int w[1000], i = p, j = r, c = v[p], k;
    for (k = p+1; k <= r; ++k)
        if (v[k] <= c) w[i++] = v[k];
        else w[j--] = v[k];
    // agora i == j
    w[i] = c;
    for (k = p; k <= r; ++k) v[k] = w[k];
    return i;
}
```

**Função sep: três parâmetros, retorna um inteiro**



## *Exemplo 3 – usando o main*

---

- A função main é especial:

## *Exemplo 3 – usando o main*

---

- A função main é especial:
  - É a primeira a ser chamada no programa
    - Todo programa tem uma!

## Exemplo 3 – usando o main

---

- A função main é especial:
  - É a primeira a ser chamada no programa
    - Todo programa tem uma!
  - Seu retorno pode indicar se o programa executou corretamente (retorno **0**) ou não (retorno **!= 0**)

## Exemplo 3 – usando o main

---

- A função main é especial:
  - É a primeira a ser chamada no programa
    - Todo programa tem uma!
  - Seu retorno pode indicar se o programa executou corretamente (retorno **0**) ou não (retorno **!= 0**)
  - Seus parâmetros, quando existem, são os parâmetros passados para o programa quando foi executado (**int argc, char \*argv[]**)

# Funções na programação

---

- Em linguagens imperativas, TODOS os programas usam funções
- No C, o programa SEMPRE começa executando a função **main**.

```
#include <stdio.h>

int main(void)
{
    printf("Olá, Mundo!");
    return 0;
}
```

# Funções sem retorno – C

- Funções sem retorno (ou **procedimentos**) devem ter o tipo de retorno **void**
- Exemplo: função para imprimir mensagem de boas-vindas do programa

```
1. void saudacao() {  
2.     printf("Ola usuario! Digite o comando que quer  
   executar, ou ? para ajuda.");  
3. }  
4. int main() {  
5.     saudacao();  
6.     ...  
7.     return 0;  
8. }
```

# Porque não retornar valor?

---

- Porque o importante pode ser a ação **colateral** da função, e não o seu valor de saída:
  - Impressão de uma mensagem
  - Ligar/desligar um componente do hardware
  - ...
- Porque a função **sempre executa sem erro**:
  - `void exit();`

# *Funções: escopo de variáveis*

---

- Variáveis podem ser acessadas somente dentro do seu escopo



# *Funções: escopo de variáveis*

---

- Variáveis podem ser acessadas somente dentro do seu escopo
- No C, o escopo é definido do momento da declaração até o fim do bloco

# *Funções: escopo de variáveis*

---

- Variáveis podem ser acessadas somente dentro do seu escopo
- No C, o escopo é definido do momento da declaração até o fim do bloco
- No C, uma variável declarada dentro de um bloco de laço vive somente uma iteração do laço

# Funções: escopo de variáveis

---

```
int teste(int x) {  
    ...  
}  
  
int main() {  
    int y;  
    for(int i=0;i<10;i++) {  
        if(i < 5) {  
            int a;  
        } else {  
            int b;  
        }  
    }  
    return 0;  
}
```

# Funções: escopo de variáveis

---

```
int teste(int x) {  
    ... //escopo de x  
}
```

```
int main() {  
    int y;  
    for(int i=0;i<10;i++) {  
        if(i < 5) {  
            int a;  
        } else {  
            int b;  
        }  
    }  
    return 0;  
}
```

# Funções: escopo de variáveis

```
int teste(int x) {  
...  
}
```

```
int main() {  
    int y;  
    for(int i=0;i<10;i++) {  
        if(i < 5) {  
            int a;  
        } else {  
            int b;  
        }  
    }  
    return 0;  
}
```

} //escopo de y

# Funções: escopo de variáveis

```
int teste(int x) {  
...  
}
```

```
int main() {  
    int y;  
    for(int i=0;i<10;i++) {  
        if(i < 5) {  
            int a;  
        } else {  
            int b;  
        }  
    }  
    return 0;  
}
```

} //escopo de i

# Funções: escopo de variáveis

---

```
int teste(int x) {  
...  
}  
  
int main() {  
    int y;  
    for(int i=0;i<10;i++) {  
        if(i < 5) {  
            int a; //escopo de a  
        } else {  
            int b;  
        }  
    }  
    return 0;  
}
```

# Funções: escopo de variáveis

---

```
int teste(int x) {  
...  
}  
  
int main() {  
    int y;  
    for(int i=0;i<10;i++) {  
        if(i < 5) {  
            int a;  
        } else {  
            int b; //escopo de b  
        }  
    }  
    return 0;  
}
```



# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741		
3742		
3743		
3744		
3745		

# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741	poupanca	
3742	juros	1.01
3743		
3744		
3745		

# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741	poupanca	500.0
3742	juros	1.01
3743		
3744		
3745		

# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741	poupanca	500.0
3742	juros	1.01
3743		
3744		
3745		



# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741	poupanca	500.0
3742	juros	1.01
3743	x	500.0
3744		
3745		

# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741	poupanca	500.0
3742	juros	1.01
3743	x	500.0
3744	novoValor	505.0
3745		



# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741	poupanca	500.0
3742	juros	1.01
3743	x	500.0
3744	novoValor	505.0
3745		

# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741	poupanca	500.0
3742	juros	1.01
3743	x	500.0
3744	novoValor	505.0
3745		

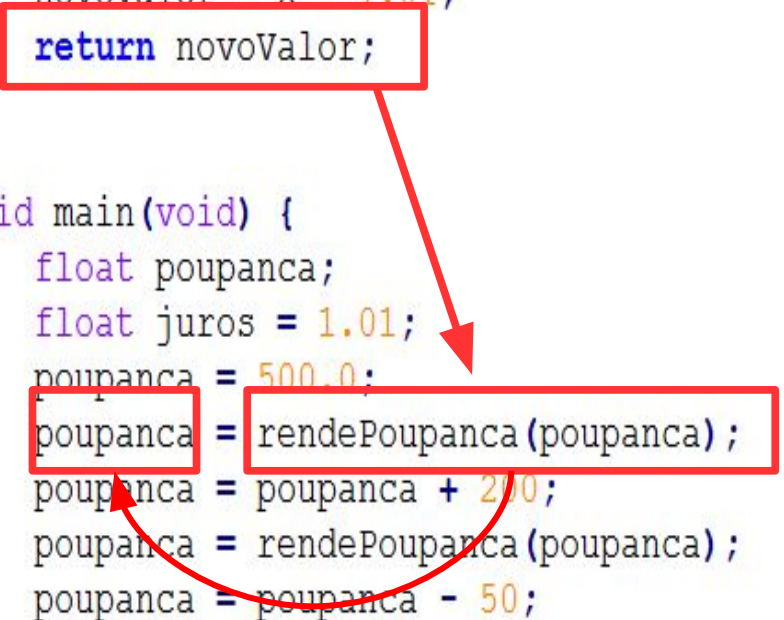
# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741	poupanca	505.0
3742	juros	1.01
3743	x	500.0
3744	novoValor	505.0
3745		

# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```



endereço	variável	conteúdo
3741	poupanca	505.0
3742	juros	1.01
3743		
3744		
3745		



# Funções: escopo de variáveis

```
1  #include <stdio.h>
2
3  float rendePoupanca(float x) {
4      float novoValor;
5      novoValor = x * 1.01;
6      return novoValor;
7  }
8
9  void main(void) {
10     float poupanca;
11     float juros = 1.01;
12     poupanca = 500.0;
13     poupanca = rendePoupanca(poupanca);
14     poupanca = poupanca + 200;
15     poupanca = rendePoupanca(poupanca);
16     poupanca = poupanca - 50;
17     poupanca = rendePoupanca(poupanca);
18     printf("Eu tenho %f reais na poupanca", poupanca);
19     printf("\n");
20     system("PAUSE");
21 }
```

endereço	variável	conteúdo
3741	poupanca	705.0
3742	juros	1.01
3743		
3744		
3745		

*Módulos*



- Um módulo é uma forma de organizar um programa grande
- Dividimos o programa em módulos, onde cada um deles possui um conjunto de tarefas bem específico:
  - Módulo de entrada/saída
  - Módulo de gerenciamento de memória
  - Módulo de cálculo
  - ....

Módulos terão uma ou mais funções, que desta forma realizam operações similares

- Módulo de operações matemáticas
- Módulo de operações sobre o tempo
- Módulo para entrada e saída

# Exemplos de módulos em C

---

- `math.h`  
operações **matemáticas** (`log`, `pow`, `sqrt`, ...)
- `time.h`  
operações sobre o **tempo** (`time`, `difftime`, ...)
- `stdio.h`  
operações de **entrada e saída** (`printf`, `scanf`, ...)

# Módulos e bibliotecas

---

- Módulos muito úteis podem ser empacotados em **bibliotecas**, para que possam ser utilizados em outros programas

# Módulos e bibliotecas - C

- Em C, carregamos módulos e bibliotecas com o comando `#include`

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

```
#include <math.h>
```

```
#include "meumodulo.h"
```

O uso de `<>` ou `" "` depende da localização do módulo/biblioteca

- No diretório de bibliotecas do sistema: `<>`
- Em outro lugar (ex, no diretório onde está o programa): `" "`

# C - Definindo um módulo

---

- O módulo consiste em:
  - Arquivo de cabeçalhos de funções e declaração de tipos de dados (extensão **.h**)
  - Arquivo com o código das funções (extensão **.c**)

# C - Definindo um módulo

## ▪ Arquivo simples.h

```
1. double media (double  
   a, double b);  
2.  
3. double dif(double a,  
   double b);  
4.
```

## ▪ Arquivo simples.c

```
1. #include "simples.h"  
2.  
3. double media(double a,  
   double b) {  
4.     return (a+b)/2;  
5. }  
6.  
7. double dif(double a,  
   double b) {  
8.     return a - b;  
9. }
```

# *Bibliotecas padrão do C*

---

- Muitas funções comuns:
  - `stdio.h` – Entrada e saída
  - `math.h` – Funções matemáticas mais complexas
  - `stdlib.h` – gerenciamento do programa: alocar memória, sair, ...
  - `time.h` – Gerenciar o tempo: imprimir datas, ver a hora/data atual...



# *Bibliotecas padrão do C*

---

Podemos encontrar a lista de funções em manuais, livros e em sites Web, i.e.:

[https://en.wikipedia.org/wiki/C\\_standard\\_library](https://en.wikipedia.org/wiki/C_standard_library)

<https://www.programiz.com/c-programming/library-function>

# Solução 5

- A função **rendePoupanca** é encarregada de realizar todas as operações de rendimento mensais da conta poupança

```
void main(void) {  
    float poupanca;  
    //saldo inicial  
    poupanca = 500.0;  
    //rendimento apos o primeiro mes  
    poupanca = rendePoupanca(poupanca);  
    //deposito de 200 reais no segundo mes  
    poupanca = poupanca + 200;  
    //rendimento apos o segundo mes  
    poupanca = rendePoupanca(poupanca);  
    //retirada de 50 reais no terceiro mes  
    poupanca = poupanca - 50;  
    //rendimento apos o terceiro mes  
    poupanca = rendePoupanca(poupanca);  
  
    printf("Eu tenho %f reais na poupanca", poupanca);  
    printf("\n");  
    system("PAUSE");  
}
```

# Solução 5

```
float rendePoupanca(float poupanca) {  
    float novoValor;  
    float juros = 1.01;  
    novoValor = poupanca * juros;  
    return novoValor;  
}
```

# Solução 6

```
void main(void) {  
    float poupanca;  
    //saldo inicial  
    poupanca = 500.0;  
    //rendimento apos o primeiro mes  
    poupanca = rendePoupanca(poupanca);  
    //deposito de 200 reais no segundo mes  
    poupanca = depositoPoupanca(poupanca, 200);  
    //rendimento apos o segundo mes  
    poupanca = rendePoupanca(poupanca);  
    //retirada de 50 reais no terceiro mes  
    poupanca = retiradaPoupanca(poupanca, 50);  
    //rendimento apos o terceiro mes  
    poupanca = rendePoupanca(poupanca);  
  
    printf("Eu tenho %f reais na poupanca", poupanca);  
    printf("\n");  
    system("PAUSE");  
}
```



# Solução 6

```
void main(void) {  
    float poupanca;  
    //saldo inicial  
    poupanca = 500.0;  
    //rendimento apos o primeiro mes  
    poupanca = rendePoupanca(poupanca);  
    //deposito de 200 reais no segundo mes  
    poupanca = depositoPoupanca(poupanca, 200);  
    //rendimento apos o segundo mes  
    poupanca = rendePoupanca(poupanca);  
    //retirada de 50 reais no terceiro mes  
    poupanca = retiradaPoupanca(poupanca, 50);  
    //rendimento apos o terceiro mes  
    poupanca = rendePoupanca(poupanca);  
  
    printf("Eu tenho %f reais na poupanca", poupanca);  
    printf("\n");  
    system("PAUSE");  
}
```

# Solução 6

```
float rendePoupanca(float poupanca) {  
    float novoValor;  
    float juros = 1.01;  
    novoValor = poupanca * juros;  
    return novoValor;  
}  
  
float depositoPoupanca(float poupanca, float valor) {  
    float novoValor;  
    novoValor = poupanca + valor;  
    return novoValor;  
}  
  
float retiradaPoupanca(float poupanca, float valor) {  
    float novoValor;  
    novoValor = poupanca - valor;  
    return novoValor;  
}
```

# Solução 6

- Tanto o “void main” quanto as funções podem estar no mesmo arquivo .c (exemplo: solucao6.c)
- No entanto, pode-se criar um módulo (exemplo: poupanca.h) para lidar com as operações padrões do banco

# Solução 7

- arquivo poupanca.h

```
float rendePoupanca(float poupanca);  
  
float depositoPoupanca(float poupanca, float valor);  
  
float retiradaPoupanca(float poupanca, float valor);  
|
```



# Solução 7

- arquivo poupanca.c

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

float rendePoupanca(float poupanca) {
    float novoValor;
    float juros = 1.01;
    novoValor = poupanca * juros;
    return novoValor;
}

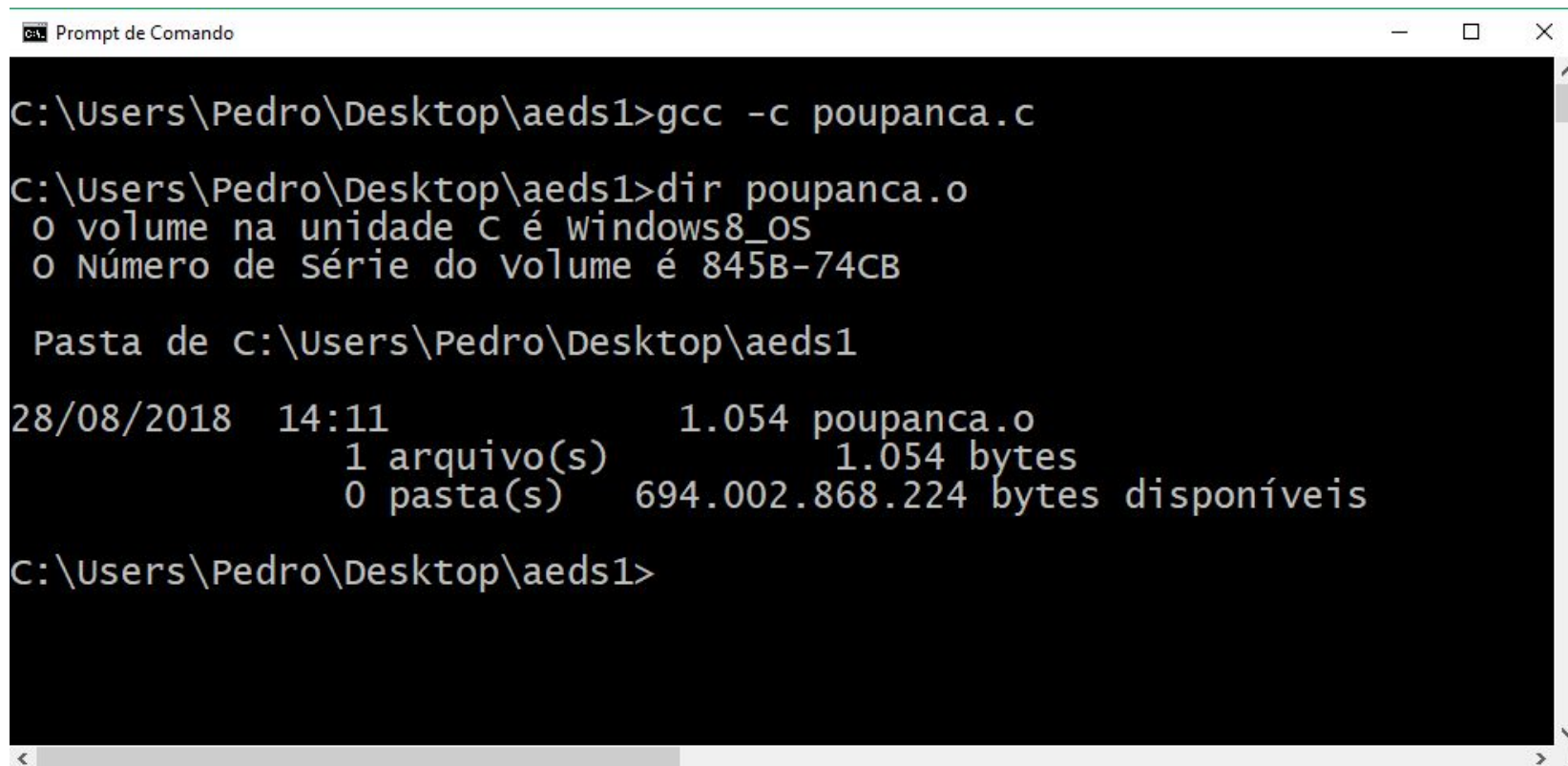
float depositoPoupanca(float poupanca, float valor) {
    float novoValor;
    printf("Eba! Depositaram %f reais!\n", valor);
    novoValor = poupanca + valor;

    return novoValor;
}

float retiradaPoupanca(float poupanca, float valor) {
    float novoValor;
    printf("Que poxa! Retiraram %f reais!\n", valor);
    novoValor = poupanca - valor;
    return novoValor;
}
```

# Solução 7

- Compilando a biblioteca:
  - gcc -c arquivo.c



```
C:\Users\Pedro\Desktop\aed1>gcc -c poupanca.c

C:\Users\Pedro\Desktop\aed1>dir poupanca.o
O volume na unidade C é windows8_OS
O Número de Série do Volume é 845B-74CB

Pasta de C:\Users\Pedro\Desktop\aed1

28/08/2018  14:11                1.054 poupanca.o
               1 arquivo(s)                1.054 bytes
               0 pasta(s) 694.002.868.224 bytes disponíveis

C:\Users\Pedro\Desktop\aed1>
```

# Solução 7

- Compilando a biblioteca:
  - gcc -c arquivo.c

Gera arquivo poupanca.o

```
C:\Users\Pedro\Desktop\aed1>gcc -c poupanca.c

C:\Users\Pedro\Desktop\aed1>dir poupanca.o
O volume na unidade C é windows8_OS
O Número de Série do Volume é 845B-74CB

Pasta de C:\Users\Pedro\Desktop\aed1
28/08/2018  14:11          1.054 poupanca.o
               1 arquivo(s)          1.054 bytes
               0 pasta(s) 694.002.868.224 bytes disponíveis

C:\Users\Pedro\Desktop\aed1>
```

# Solução 7

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

void main(void) {
    float poupanca;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = rendePoupanca(poupanca);
    //deposito de 200 reais no segundo mes
    poupanca = depositoPoupanca(poupanca, 200);
    //rendimento apos o segundo mes
    poupanca = rendePoupanca(poupanca);
    //retirada de 50 reais no terceiro mes
    poupanca = retiradaPoupanca(poupanca, 50);
    //rendimento apos o terceiro mes
    poupanca = rendePoupanca(poupanca);

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```



# Solução 7

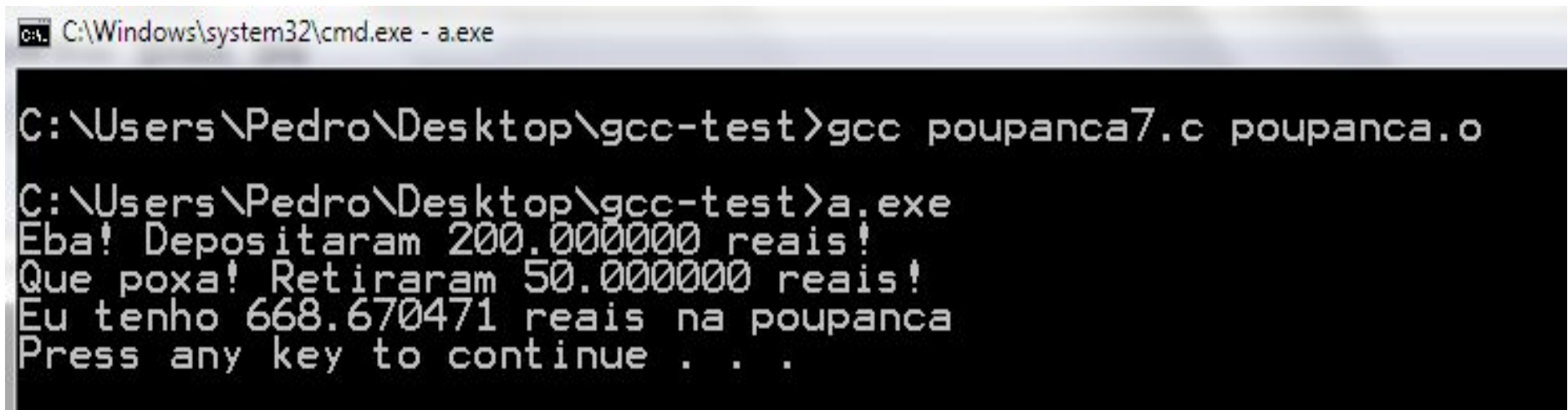
```
#include <stdio.h>
#include "poupanca.h"
```

```
void main(void) {
    float poupanca;
    //saldo inicial
    poupanca = 500.0;
    //rendimento apos o primeiro mes
    poupanca = rendePoupanca(poupanca);
    //deposito de 200 reais no segundo mes
    poupanca = depositoPoupanca(poupanca, 200);
    //rendimento apos o segundo mes
    poupanca = rendePoupanca(poupanca);
    //retirada de 50 reais no terceiro mes
    poupanca = retiradaPoupanca(poupanca, 50);
    //rendimento apos o terceiro mes
    poupanca = rendePoupanca(poupanca);

    printf("Eu tenho %f reais na poupanca", poupanca);
    printf("\n");
    system("PAUSE");
}
```

# Solução 7

- Compilando o programa usando o módulo criado
  - gcc arquivo.c modulo.o



```
C:\Windows\system32\cmd.exe - a.exe

C:\Users\Pedro\Desktop\gcc-test>gcc poupanca7.c poupanca.o

C:\Users\Pedro\Desktop\gcc-test>a.exe
Eba! Depositaram 200.000000 reais!
Que poxa! Retiraram 50.000000 reais!
Eu tenho 668.670471 reais na poupanca
Press any key to continue . . .
```

# Resumo do processo

- Passo #1
  - Você precisa criar dois arquivos (com o mesmo nome) para o módulo
    - 1 arquivo .h contendo só o cabeçalho das funções do módulo
      - ex: poupanca.h, slide 84
    - 1 arquivo .c contendo a implementação das funções contidas no arquivo .h
      - ex: poupanca.c, slide 85
  - não esqueça de incluir o arquivo .h (“ex: #include “poupanca.h”)
- (continua)



# Resumo do processo

- Passo #2
  - Compile o módulo criado a partir do arquivo .c criado
    - ex: `gcc -c poupanca.c`
  - Se compilar corretamente, você gerou um arquivo .o com o nome do módulo
    - ex: `poupanca.o`
- (continua)

# Resumo do processo

- Passo #3
  - crie um arquivo .c para testar o módulo criado
    - ex: `testepoupanca7.c` (slide 88)
  - este arquivo deve incluir o .h do módulo criado
    - ex: `"#include poupanca.h"`
- Passo #4
  - compile o arquivo de teste criado fazendo a ligação explícita para o arquivo compilado .o do seu módulo
    - ex: `gcc testepoupanca7.c poupanca.o`
- Passo #5
  - execute o programa a .exe

# *Bibliotecas padrão do C – Math.h*

---

- A math.h é especial: precisa de um parâmetro na compilação (somente em sistemas UNIX)
- gcc codigo.c **-lm**
- **-l"nome"** indica que queremos que o programa "incorpore" código de um módulo externo.
- TODA biblioteca precisa do **-l"nome"**, EXCETO as funções padrão do C

# Exercício

- Brutus e Olívia foram ao médico, que disse a eles que ambos estão fora do peso ideal. Ambos discordaram veementemente da afirmação do médico. Para provar que estava certo, o médico mostrou o Índice de Massa Corporal (IMC) de ambos, considerando que Brutus tem 1,84m e pesa 112kg e Olívia tem 1,76m e pesa 49kg. Implemente um programa para mostrar o IMC de Brutus e Olívia e quantos kilos Brutus e Olívia devem perder/ganhar para atingirem um peso saudável segundo a classificação do IMC.

# Exercício



IMC	Classificação
$< 16$	Magreza grave
16 a $< 17$	Magreza moderada
17 a $< 18.5$	Magreza leve
18.5 a $< 25$	Saudável
25 a $< 30$	Sobrepeso
30 a $< 35$	Obesidade grau I
35 a $< 40$	Obesidade grau II (severa)
$\geq 40$	Obesidade grau III (mórbida)

Table 3.1: Classificação do IMC

# Exercício

```
1  #include <stdio.h>
2  #include <math.h>
3
4  float calculaDifPeso(float altura, float peso) {
5      float ic = peso/(altura * altura);
6      float peso_ideal = (altura * altura) * 21;
7      float dif = peso - peso_ideal;
8      printf("\nIMC = %f, dif = %f", ic, dif);
9      return dif;
10 }
11
12 void main() {
13     float pesoOlivia = 49, pesoBrutus = 112;
14     float alturaOlivia = 1.76, alturaBrutus = 1.84;
15     float difOlivia = calculaDifPeso(alturaOlivia, pesoOlivia);
16     float difBrutus = calculaDifPeso(alturaBrutus, pesoBrutus);
17     float total = fabs(difOlivia) + fabs(difBrutus);
18     printf("\nPeso total que eles devem ganhar ou perder: %f", total);
19 }
```