

Aula 07 – Variáveis

Norton T. Roman & Luciano A. Digiampietri

Variáveis

```
#include <stdio.h>

int main() {
    printf("Programa para cálculo da área da casa\n");
    printf("A área da sala é %i\n", 10*10);
    printf("A área do quarto é %i\n", 5*7);
    printf("A área do banheiro é %i\n", 5*7);
    printf("A área total é %i\n", 10*10 + 5*7 + 5*7);
    return 0;
}
```

- Algo de estranho nesse programa?

Variáveis

```
#include <stdio.h>

int main() {
    printf("Programa para cálculo da área da casa\n");
    printf("A área da sala é %i\n", 10*10);
    printf("A área do quarto é %i\n", 5*7);
    printf("A área do banheiro é %i\n", 5*7);
    printf("A área total é %i\n", 10*10 + 5*7 + 5*7);
    return 0;
}
```

- Algo de estranho nesse programa?
 - “5*7” é repetido 4 vezes no código

Variáveis

```
#include <stdio.h>

int main() {
    printf("Programa para cálculo da área da casa\n");
    printf("A área da sala é %i\n", 10*10);
    printf("A área do quarto é %i\n", 5*7);
    printf("A área do banheiro é %i\n", 5*7);
    printf("A área total é %i\n", 10*10 + 5*7 + 5*7);
    return 0;
}
```

- Algo de estranho nesse programa?
 - “5*7” é repetido 4 vezes no código
 - “10*10” repete 2 vezes

Variáveis

```
#include <stdio.h>

int main() {
    printf("Programa para cálculo da área da casa\n");
    printf("A área da sala é %i\n", 10*10);
    printf("A área do quarto é %i\n", 5*7);
    printf("A área do banheiro é %i\n", 5*7);
    printf("A área total é %i\n", 10*10 + 5*7 + 5*7);
    return 0;
}
```

- E se precisarmos trocar algum dos valores?

Variáveis

```
#include <stdio.h>

int main() {
    printf("Programa para cálculo da área da casa\n");
    printf("A área da sala é %i\n", 10*10);
    printf("A área do quarto é %i\n", 5*7);
    printf("A área do banheiro é %i\n", 5*7);
    printf("A área total é %i\n", 10*10 + 5*7 + 5*7);
    return 0;
}
```

- E se precisarmos trocar algum dos valores?
 - Teremos que trocar em vários lugares no código

- Que fazer então para evitar as repetições?

Variáveis

- Que fazer então para evitar as repetições?
- Seria interessante ter algo assim:

Variáveis

- Que fazer então para evitar as repetições?
- Seria interessante ter algo assim:
 - Calculamos a área da sala e guardamos na memória

Variáveis

- Que fazer então para evitar as repetições?
- Seria interessante ter algo assim:
 - Calculamos a área da sala e guardamos na memória
 - Calculamos a área do quarto e guardamos na memória

- Que fazer então para evitar as repetições?
- Seria interessante ter algo assim:
 - Calculamos a área da sala e guardamos na memória
 - Calculamos a área do quarto e guardamos na memória
 - Para a do banheiro, usamos a do quarto, que está na memória

- Que fazer então para evitar as repetições?
- Seria interessante ter algo assim:
 - Calculamos a área da sala e guardamos na memória
 - Calculamos a área do quarto e guardamos na memória
 - Para a do banheiro, usamos a do quarto, que está na memória
 - Para a área total, somamos a da sala com 2 vezes a do banheiro

- Que fazer então para evitar as repetições?
- Seria interessante ter algo assim:
 - Calculamos a área da sala e guardamos na memória
 - Calculamos a área do quarto e guardamos na memória
 - Para a do banheiro, usamos a do quarto, que está na memória
 - Para a área total, somamos a da sala com 2 vezes a do banheiro
- Todas em memória

- Como guardar algo na memória?

Variáveis

- Como guardar algo na memória?
- Primeiro, temos que reservar um espaço (alocação)

Variáveis

- Como guardar algo na memória?
- Primeiro, temos que reservar um espaço (alocação)
 - De que tamanho?

Variáveis

- Como guardar algo na memória?
- Primeiro, temos que reservar um espaço (alocação)
 - De que tamanho?
 - O suficiente para guardar o valor que queremos → um inteiro

Variáveis

- Como guardar algo na memória?
- Primeiro, temos que reservar um espaço (alocação)
 - De que tamanho?
 - O suficiente para guardar o valor que queremos → um inteiro
- Como?

Variáveis

```
#include <stdio.h>

int main() {
    // área do quarto
    int areaq;
    // área da sala
    int areas;
    // área total
    int areat;
    return 0;
}
```

Variáveis

```
#include <stdio.h>

int main() {
    // área do quarto
    int areaq;
    // área da sala
    int areas;
    // área total
    int areat;
    return 0;
}
```

Isso diz ao compilador para reservar (alocar) espaço na memória suficiente para 3 inteiros, dando a eles o nome de “areaq”, “areas” e “areat”

Variáveis

```
#include <stdio.h>

int main() {
    // área do quarto
    int areaq;
    // área da sala
    int areas;
    // área total
    int areat;
    return 0;
}
```

Isso diz ao compilador para reservar (alocar) espaço na memória suficiente para 3 inteiros, dando a eles o nome de “areaq”, “areas” e “areat”

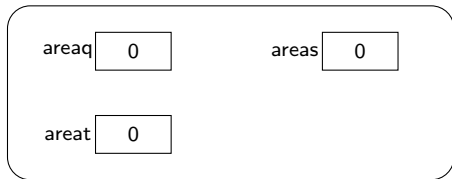
- *areaq*, *areas* e *areat* são **variáveis**
 - *int* é seu **tipo**

Variáveis

```
#include <stdio.h>

int main() {
    // área do quarto
    int areaq;
    // área da sala
    int areas;
    // área total
    int areat;
    return 0;
}
```

Esquema da memória:



Variáveis

Alocado o espaço, podemos por algo lá → **atribuição**

Variáveis

Alocado o espaço, podemos por algo lá → **atribuição**

```
#include <stdio.h>
int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    printf("Programa para cálculo da área da casa\n");
    areas = 10*10;
    printf("A área da sala é %i\n", areas);
    areaq = 7*5;
    printf("A área do quarto é %i\n", areaq);
    printf("A área do banheiro é %i\n", areaq);
    areat = areas + 2*areaq;
    printf("A área total é %i\n", areat);
    return 0;
}
```


Variáveis

```
#include <stdio.h>

int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    areas = 10*10;
    ...
    areaq = 7*5;
    ...
    areat = areas + 2*areaq;
    ...
}
```

Ao fazermos
`nome_var = valor;`
estamos armazenando
valor na região da memória
correspondente a `nome_var`



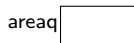
Variáveis

```
#include <stdio.h>

int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    areas = 10*10;
    ...
    areaq = 7*5;
    ...
    areat = areas + 2*areaq;
    ...
}
```

Ao fazermos
`nome_var = valor;`
estamos armazenando
valor na região da memória
correspondente a `nome_var`



areaq

Variáveis

```
#include <stdio.h>

int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    areas = 10*10;
    ...
    areaq = 7*5;
    ...
    areat = areas + 2*areaq;
    ...
}
```

Ao fazermos
`nome_var = valor;`
estamos armazenando
valor na região da memória
correspondente a `nome_var`

areaq



areas



Variáveis

```
#include <stdio.h>

int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    areas = 10*10;
    ...
    areaq = 7*5;
    ...
    areat = areas + 2*areaq;
    ...
}
```

Ao fazermos
`nome_var = valor;`
estamos armazenando
valor na região da memória
correspondente a `nome_var`

areaq

areas

areat

Variáveis

```
#include <stdio.h>

int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    areas = 10*10;
    ...
    areaq = 7*5;
    ...
    areat = areas + 2*areaq;
    ...
}
```

Ao fazermos
`nome_var = valor;`
estamos armazenando
valor na região da memória
correspondente a `nome_var`

areaq

areas

areat

Variáveis

```
#include <stdio.h>

int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    areas = 10*10;
    ...
    areaq = 7*5;
    ...
    areat = areas + 2*areaq;
    ...
}
```

Ao fazermos
`nome_var = valor;`
estamos armazenando
valor na região da memória
correspondente a `nome_var`

areaq

areas

areat

Variáveis

```
#include <stdio.h>

int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    areas = 10*10;
    ...
    areaq = 7*5;
    ...
    areat = areas + 2*areaq;
    ...
}
```

Ao fazermos
`nome_var = valor;`
estamos armazenando
valor na região da memória
correspondente a `nome_var`

areaq 35

areas 100

areat 170

Variáveis

```
#include <stdio.h>

int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    areas = 10*10;
    ...
    areaq = 7*5;
    ...
    areat = areas + 2*areaq;
    ...
}
```

Ao fazermos
`nome_var = valor;`
estamos armazenando
valor na região da memória
correspondente a `nome_var`

areaq 35

areas 100

areat 170

Não é um igual

Variáveis

```
#include <stdio.h>

int main() {
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    areas = 10*10;
    ...
    areaq = 7*5;
    ...
    areat = areas + 2*areaq;
    ...
}
```

Ao fazermos
`nome_var = valor;`
estamos armazenando
valor na região da memória
correspondente a `nome_var`

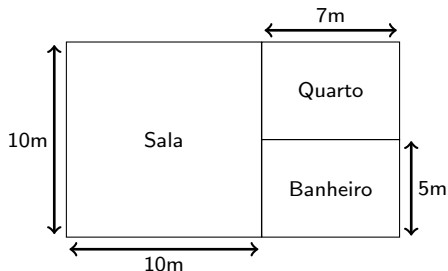
areaq 35

areas 100

areat 170

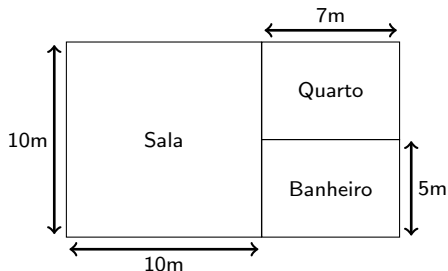
Não é um igual
A variável que recebe
sempre está à esquerda

- Que outros detalhes podemos notar da cabana?



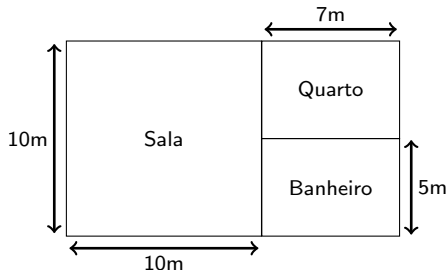
Variáveis

- Que outros detalhes podemos notar da cabana?
- A sala é quadrada \rightarrow basta sabermos o lado



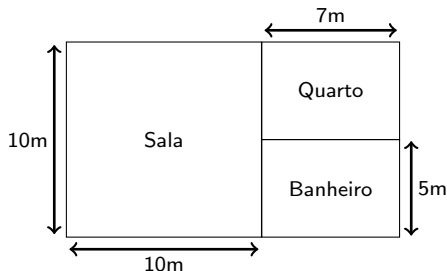
Variáveis

- Que outros detalhes podemos notar da cabana?
- A sala é quadrada \rightarrow basta sabermos o lado
- Tanto o quarto quanto o banheiro possuem metade do lado da sala



Variáveis

- Que outros detalhes podemos notar da cabana?
- A sala é quadrada \rightarrow basta sabermos o lado
- Tanto o quarto quanto o banheiro possuem metade do lado da sala
- Vamos reescrever o programa...



Variáveis

```
#include <stdio.h>
int main() {
    int lateral = 10; // comprimento da lateral da cabana
    int cquarto = 7; // comprimento da lateral maior do quarto
    int areaq; // área do quarto
    int areas; // área da sala
    int areat; // área total

    printf("Programa para cálculo da área da casa\n");
    areas = lateral*lateral;
    printf("A área da sala é %i\n", areas);
    areaq = cquarto*(lateral/2);
    printf("A área do quarto é %i\n", areaq);
    printf("A área do banheiro é %i\n", areaq);
    areat = areas + 2*areaq;
    printf("A área total é %i\n", areat);
    return 0;
}
```

- E qual a vantagem disso?

- E qual a vantagem disso?
 - Reduz nossa dependência a valores externos: antes eram 3 (10, 7 e 5), agora são 2 (10 e 7)
 - Reduz a chance de erros na substituição

- E qual a vantagem disso?
 - Reduz nossa dependência a valores externos: antes eram 3 (10, 7 e 5), agora são 2 (10 e 7)
 - Reduz a chance de erros na substituição
- E a desvantagem?

- E qual a vantagem disso?
 - Reduz nossa dependência a valores externos: antes eram 3 (10, 7 e 5), agora são 2 (10 e 7)
 - Reduz a chance de erros na substituição
- E a desvantagem?
 - Gasta mais memória, com as variáveis *lateral* e *cquarto*

Divisão Inteira

```
#include <stdio.h>
int main() {
    int lateral = 11;
    int cquarto = 7;
    int areaq;
    int areas;
    int areat;
    printf("Programa...\n");
    areas = lateral*lateral;
    printf("A área... %i\n", areas);
    areaq = cquarto*(lateral/2);
    printf("A área... %i\n", areaq);
    printf("A área... %i\n", areaq);
    areat = areas + 2*areaq;
    printf("A área... %i\n", areat);
    return 0;
}
```

- E se a lateral for 11?
Qual a saída?

Divisão Inteira

```
#include <stdio.h>
int main() {
    int lateral = 11;
    int cquarto = 7;
    int areaq;
    int areas;
    int areat;
    printf("Programa...\n");
    areas = lateral*lateral;
    printf("A área... %i\n", areas);
    areaq = cquarto*(lateral/2);
    printf("A área... %i\n", areaq);
    printf("A área... %i\n", areaq);
    areat = areas + 2*areaq;
    printf("A área... %i\n", areat);
    return 0;
}
```

- E se a lateral for 11?
Qual a saída?

Programa para cálculo da
área da casa

A área da sala é 121

A área do banheiro é 35

A área do quarto é 35

A área total é 191

Divisão Inteira

```
#include <stdio.h>
int main() {
    int lateral = 11;
    int cquarto = 7;
    int areaq;
    int areas;
    int areat;
    printf("Programa...\n");
    areas = lateral*lateral;
    printf("A área... %i\n", areas);
    areaq = cquarto*(lateral/2);
    printf("A área... %i\n", areaq);
    printf("A área... %i\n", areaq);
    areat = areas + 2*areaq;
    printf("A área... %i\n", areat);
    return 0;
}
```

- E se a lateral for 11?
Qual a saída?

Programa para cálculo da
área da casa
A área da sala é 121
A área do banheiro é 35
A área do quarto é 35
A área total é 191

- Fez $11 \div 2 = 5 \dots$ Por
quê?

Divisão Inteira

- O problema está na linha

```
areaq = cquarto*(lateral/2);
```

- Como tanto `lateral` quanto `2` são inteiros, o compilador acha que essa é uma divisão inteira, dando somente o quociente da divisão:

$$\begin{array}{r} 11 \quad | \quad 2 \\ 1 \quad | \quad 5 \end{array}$$

Divisão Inteira

- E como obtemos o **resto da divisão**?

Divisão Inteira

- E como obtemos o **resto da divisão**?
- Usando % em vez de /. Ex:

```
#include <stdio.h>
int main() {
    printf("Parte inteira: %i\n", 11/2);
    printf("Resto: %i\n", 11%2);
}
```


Divisão Inteira

- E como obtemos o **resto da divisão**?
- Usando % em vez de /. Ex:

```
#include <stdio.h>
int main() {
    printf("Parte inteira: %i\n", 11/2);
    printf("Resto: %i\n", 11%2);
}
```

- E a saída será:

```
Parte inteira: 5
Resto: 1
```

Divisão Inteira

- Mas isso não resolve nosso problema

Divisão Inteira

- Mas isso não resolve nosso problema
 - Queremos ver 38.5 na tela

Divisão Inteira

- Mas isso não resolve nosso problema
 - Queremos ver 38.5 na tela
- Problema:
 - 38.5 é um número real, e nossas variáveis são inteiras

Divisão Inteira

- Mas isso não resolve nosso problema
 - Queremos ver 38.5 na tela
- Problema:
 - 38.5 é um número real, e nossas variáveis são inteiras
- Solução:
 - Troque o tipo das variáveis

Divisão Inteira

```
#include <stdio.h>
int main() {
    float lateral = 11;
    float cquarto = 7;
    float areaq;
    float areas;
    float areat;
    printf("Programa para cálculo da área da casa\n");
    areas = lateral*lateral;
    printf("A área da sala é %f\n", areas);
    areaq = cquarto*(lateral/2);
    printf("A área do quarto é %f\n", areaq);
    printf("A área do banheiro é %f\n", areaq);
    areat = areas + 2*areaq;
    printf("A área total é %f\n", areat);
    return 0;
}
```

Tipos Numéricos

Tipo	Conjunto	Valor mínimo	Valor máximo	Bits
short	inteiro	-32.768	32.767	16
int	inteiro	-2.147.483.648	2.147.483.647	32
long	inteiro	-9.223.372.036.854.775.808	9.223.372.036.854.775.807	64
float	real	-	-	32
double	real	-	-	64

- *float* e *double* obedecem ao IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985
- Tem representações para infinito (positivo e negativo) e para valores não numéricos (NaN), usado, por exemplo, em casos de divisão por zero, raiz de número negativo etc

Tipos Numéricos

- Curiosidade: gerando o NaN...

```
#include <stdio.h>
int main() {
    double x = 0;
    double y = 0;
    printf("%f\n",x/y);
}
```


Tipos Numéricos

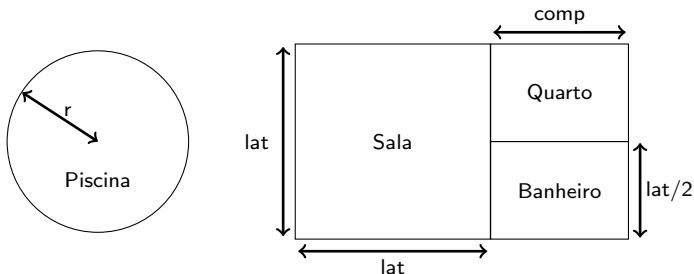
- Curiosidade: gerando o NaN...

```
#include <stdio.h>
int main() {
    double x = 0;
    double y = 0;
    printf("%f\n",x/y);
}
```

- -nan

Constantes

- Suponha que queremos incrementar nossa cabana com uma piscina:



- Queremos então fazer um programa que calcule a área da cabana e da piscina

Constantes

- Como?

Constantes

- Como?
 - Temos o raio da piscina
 - Basta vermos como adicionar o π

Constantes

- Como?
 - Temos o raio da piscina
 - Basta vermos como adicionar o π
- Podemos fazer:

```
#include <stdio.h>

int main() {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;

    areap = pi * raio * raio;
    printf("Área: %f\n", areap);

    return 0;
}
```

Constantes

- Como?
 - Temos o raio da piscina
 - Basta vermos como adicionar o π
- Podemos fazer:
 - E a saída será "Área: 12.566360"

```
#include <stdio.h>

int main() {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;

    areap = pi * raio * raio;
    printf("Área: %f\n", areap);

    return 0;
}
```

Constantes

- E se fizermos:

```
#include <stdio.h>

int main() {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;
    pi = 12;
    areap = pi * raio * raio;
    printf("Área: %f\n", areap);

    return 0;
}
```

Constantes

- E se fizermos:
 - Teremos “Área: 48.000000”
 - Inadvertidamente mudamos algo que deveria ser constante

```
#include <stdio.h>

int main() {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;
    pi = 12;
    areap = pi * raio * raio;
    printf("Área: %f\n", areap);

    return 0;
}
```


Constantes

- Devemos tornar π constante, fazendo:

```
#include <stdio.h>

int main() {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;

    areap = pi * raio * raio;
    printf("Área: %f\n", areap);

    return 0;
}
```

Constantes

- Devemos tornar π constante, fazendo:

```
#include <stdio.h>
#define pi 3.14159
int main() {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;

    areap = pi * raio * raio;
    printf("Área: %f\n", areap);

    return 0;
}
```

Constantes

- Devemos tornar π constante, fazendo:
- E, se tentarmos mudar o valor, teremos

```
#include <stdio.h>
#define pi 3.14159
int main() {
    // raio da piscina
    double raio = 2;
    // área da piscina
    double areap;
    // valor do pi
    double pi = 3.14159;
    pi = 12;
    areap = pi * raio * raio;
    printf("Área: %f\n", areap);

    return 0;
}
```

Constantes

- Devemos tornar π constante, fazendo:
- E, se tentarmos mudar o valor, teremos
- ```
$ clang-7 AreaPiscina.c
main.c:9:8: error: expression
 is not assignable
 pi = 12;
 ~ ~ ^
1 error generated.
```

```
#include <stdio.h>
#define pi 3.14159
int main() {
 // raio da piscina
 double raio = 2;
 // área da piscina
 double areap;
 // valor do pi
 double pi = 3.14159;
 pi = 12;
 areap = pi * raio * raio;
 printf("Área: %f\n", areap);

 return 0;
}
```

# Constantes

- “#define <nome>  
<valor>” define um valor associado a um nome
- Define uma “**constante**” (não é criada uma variável)
- Recomenda-se que essas constantes estejam em letras **maiúsculas**

```
#include <stdio.h>
#define PI 3.14159
int main() {
 // raio da piscina
 double raio = 2;
 // área da piscina
 double areap;

 areap = PI * raio * raio;
 printf("Área: %f\n", areap);

 return 0;
}
```

# Constantes

- Alternativamente, podemos usar uma constante já definida em C, em math.h:
- M\_PI, valendo  
3.14159265358979323846

```
#include <stdio.h>
#include <math.h>
int main() {
 // raio da piscina
 double raio = 2;
 // área da piscina
 double areap;
 areap = M_PI * raio * raio;
 printf("Área: %f\n", areap);

 return 0;
}
```

# Constantes

- Existe outra forma de definir constantes em C
- `const`: o modificador `const` indica que uma “variável” é uma constante

```
#include <stdio.h>
int main() {
 const double PI = 3.14159;
 // raio da piscina
 double raio = 2;
 // área da piscina
 double areap;
 areap = PI * raio * raio;
 printf("Área: %f\n", areap);

 return 0;
}
```

# Aula 07 – Variáveis

Norton T. Roman & Luciano A. Digiampietri