Aula 09 – Type Casting e Escopo de Variáveis

Norton T. Roman & Luciano A. Digiampietri

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de usarmos apenas 2 casas decimais?

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de usarmos apenas 2 casas decimais?
  - Sim!

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de usarmos apenas 2 casas decimais?
  - Sim!

```
printf("%.2f\n",12.566370614359172);
```

Type Casting

- Voltando ao código, reparou no tamanho de nossa resposta?
  - 12.566370614359172
- Não haveria um modo de usarmos apenas 2 casas decimais?
  - Sim!

    ```
    printf("%.2f\n",12.566370614359172);
    ```

  - 12.57

Type Casting

- E se quisermos fazer um programa que "trunque" o valor 12.566370614359172 para 12.56?

Type Casting

- E se quisermos fazer um programa que "trunque" o valor 12.566370614359172 para 12.56?

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
         trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- `(int)(valor*100)` ?

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- `(int)(valor*100)` ?
  - Multiplique `valor` (double) por 100 (o resultado será do tipo *double*)

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- `(int)(valor*100)` ?
  - Multiplique `valor` (double) por 100 (o resultado será do tipo *double*)
  - Transforme esse resultado em um inteiro (guarda 1256 em `novoValor`)

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- Ao final, transformamos `novoValor` novamente em double

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- Ao final, transformamos `novoValor` novamente em double
  - E se tivéssemos feito *return(novoValor/100)*?

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- Ao final, transformamos `novoValor` novamente em double
  - E se tivéssemos feito *return(novoValor/100)*?
  - O resultado seria 12.000000

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- Ao final, transformamos `novoValor` novamente em double
  - E se tivéssemos feito *return(novoValor/100)*?
  - O resultado seria 12.000000
- Mudanças assim são chamadas de Type casting

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- Ao final, transformamos `novoValor` novamente em double
  - E se tivéssemos feito *return(novoValor/100)*?
  - O resultado seria 12.000000
- Mudanças assim são chamadas de Type casting
  - Mudança de um tipo para outro

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- Cuidado! Mudanças de tipos menores para maiores não geram perda (ex: int $\rightarrow$ long)

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- Cuidado! Mudanças de tipos menores para maiores não geram perda (ex: int $\rightarrow$ long)
  - `(double)novoValor/100` não gerou perda

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- Cuidado! Mudanças de tipos menores para maiores não geram perda (ex: int $\rightarrow$ long)
  - `(double)novoValor/100` não gerou perda
- Já de tipos maiores para menores podem gerar perda (ex: long $\rightarrow$ int)

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Type Casting

- Cuidado! Mudanças de tipos menores para maiores não geram perda (ex: int $\rightarrow$ long)
  - `(double)novoValor/100` não gerou perda
- Já de tipos maiores para menores podem gerar perda (ex: long $\rightarrow$ int)
  - `(int)(valor*100)` gerou uma perda

```
/*
    Trunca um valor na 2a casa
*/
#include <stdio.h>
double trunca(double valor) {
  int novoValor = (int)(valor*100);
  return((double)novoValor/100);
}

int main() {
  printf("%f\n",
          trunca(12.566370614359172));
  return 0;
}
```

Variáveis

- Suponha agora que queremos também saber o valor da construção, tomando como base o valor do metro quadrado
- Como fazer?

Variáveis

- Suponha agora que queremos também saber o valor da construção, tomando como base o valor do metro quadrado
- Como fazer? Duas alternativas:

Variáveis

- Suponha agora que queremos também saber o valor da construção, tomando como base o valor do metro quadrado
- Como fazer? Duas alternativas:
- Definir o valor dentro da função:

```
double valor(double area) {
  double valorM2 = 1500;
  return(valorM2*area);
}
```

Variáveis

- Ou passar o valor como parâmetro

```
double valor(double area,
                    double valorM2){
  return(valorM2*area);
}
```

Variáveis

- Ou passar o valor como parâmetro

```
double valor(double area,
                    double valorM2){
  return(valorM2*area);
}
```

- Qual das duas alternativas seria a melhor?

Variáveis

- Ou passar o valor como parâmetro

```
double valor(double area,
                    double valorM2){
  return(valorM2*area);
}
```

- Qual das duas alternativas seria a melhor?
- Ambas apresentam problemas semelhantes

Variáveis

- Ou passar o valor como parâmetro

```
double valor(double area,
                    double valorM2){
  return(valorM2*area);
}
```

- Qual das duas alternativas seria a melhor?
- Ambas apresentam problemas semelhantes
  - Mudanças no valor do metro quadrado são difíceis de serem feitas
    - Ou devem ser buscadas dentro da função (onde quer que ele esteja no código)
    - Ou devem ser buscadas em cada chamada à função

Variáveis

- O preço do metro quadrado parece mais uma variável do problema como um todo
- É único para o programa como um todo
  - Algo que, em softwares gerais, estaria em algum menu "Opções", "Setup" etc.

Variáveis

- E como declarar variáveis assim?

Variáveis

- E como declarar variáveis assim?
- Fora de qualquer função no programa
- Deixamos variável para permitir mudanças

```
/* valor do metro
                    quadrado */
double valorM2 = 1500;
...
```

Variáveis

- E como podemos acessar o valor?

Variáveis

- E como podemos acessar o valor?

  - De dentro de qualquer função (ou corpo) do programa
  - Como faríamos com uma constante

```
double valorM2 = 1500;
   ...

double valor(double area){
    return(valorM2*area);
}
```

Escopo

- Consideremos agora outra função do programa

```
double valorM2 = 1500;
...
void areaCasa(float lateral,
                  float cquarto){
    float areaq;
    float areas;
    float areat;
    printf("Programa ...\n");
    areas = lateral*lateral;
    printf("A área ... %f\n", areas);
    areaq = cquarto*(lateral/2);
    printf("A área ... %f\n", areaq);
    printf("A área ... %f\n", areaq);
    areat = areas + 2*areaq;
    printf("A área ... %f\n", areat);
}
...
double valor(double area) {
    areat = 3;
    valorM2 = 5;
    return(valorM2*area);
}
...
```

Escopo

- Consideremos agora outra função do programa
- Conseguiremos fazer essas atribuições?

```
double valorM2 = 1500;
...
void areaCasa(float lateral,
                  float cquarto){
    float areaq;
    float areas;
    float areat;
    printf("Programa ...\n");
    areas = lateral*lateral;
    printf("A área ... %f\n", areas);
    areaq = cquarto*(lateral/2);
    printf("A área ... %f\n", areaq);
    printf("A área ... %f\n", areaq);
    areat = areas + 2*areaq;
    printf("A área ... %f\n", areat);
}
...
double valor(double area) {
    areat = 3;
    valorM2 = 5;
    return(valorM2*area);
}
...
```

Escopo

- Compilando...

```
main.c:29:2: error: use of
undeclared identifier 'areat';
did you mean 'area'?
        areat = 3;
        ^~~~~
        area
1 error
```

- areat não foi encontrada, mas valorM2 foi

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

int valorM2 = 1500;

void areaCasa(float lateral,
                  float cquarto){
    float areaq;
    float areas;
    float areat;
    ...
}

...
double valor(double area) {
  areat = 3;
  valorM2 = 5;
  return(valorM2*area);
}
...
```

Escopo

- Variáveis declaradas dentro de uma função:
  - Visibilidade: dentro da própria função
  - Diz-se que seu escopo é a função
- Variáveis declaradas fora de qualquer função (inclusive a *main*):
  - Visibilidade: dentro de todo o programa
  - Diz-se que seu escopo é o programa

Escopo

Cuidado!

- Da mesma forma que qualquer função pode acessar uma variável, se ela não for *const* (constante) qualquer função poderá também modificá-la

```
double valor(double area) {
  valorM2 = 5;
  return(valorM2*area);
}
```

Aula 09 – Type Casting e Escopo de Variáveis

Norton T. Roman & Luciano A. Digiampietri