[Declaração | Conversão | Python]

Introdução à conversão inteiro/flutuante e divisões

Neste texto apresentaremos a conversão entre variáveis inteiras e flutuantes, além de destacar como obter o quociente e o resto da divisão entre inteiros.

Na seção seguinte apresentamos um resumo sobre variáveis inteiras e flutuantes tanto em *C* quanto em *Python* e nas duas seções seguintes individualizamos as explicações para cada uma das duas linguagens.

1. Declaração de tipo *inteiro* e *flutuante* em *C* e em *Python*

Como o próprio nome diz, uma variável inteira pode representar apenas números inteiros, mas o significado de uma variável em *ponto flutuante* não é tão óbvio, mas simplificadamente, podemos entender que é o tipo de variável que permite trabalhar com valores "reais", por exemplo, para podermos armazenar valores como `5/2 = 2.5`. As explicações sobre a razão desse tipo de variável ser denotada por *ponto flutuante* é explicada em outro texto.

Variável = posição de memória + tamanho (em *bits*). Como explicado em outros textos, uma variável está associada à determinada posição de memória e à uma quantidade conhecida de *bits*, pois desse modo pode-se interpretar corretamente o valor ali representado. Desse modo podemos dizer que cada variável é de um **tipo** específico, como *tipo inteiro* ou *tipo real/flutuante*.

Definição de tipo. Na linguagem *Python*, o *tipo* é definido na primeira vez que a variável é interpretada, por exemplo, se ela recebe um valor inteiro, será inteira. Já na linguagem *C*, como é obrigatório declarar todas as variáveis que serão usadas, a associação é feita no momento dessa declaração.

Formatador de tipo. Para se conseguir registrar um valor digitado pelo usuário ou, inversamente, para se imprimir um valor, deve-se também saber como estão organizados os *bits* associados ao valor, além de saber quantos *bits* determinada variável usa. Ou seja, deve existir um algoritmo interno para converter os *bits* para o valor legível (usualmente em decimal). As linguagens *C* e *Python* empregam técnicas distintas para fazer isso.
Em *C*, tanto para a **leitura** (*entrada*) quanto para a **impressão** (*saída*), deve-se utilizar um formatador (e.g. `%d` para inteior e `%f` para flutuante). Para *leitura* de um inteiro e um flutuante usa-se `scanf("%d %f", &i, &f);` e para *impressão* `printf("%d %f\n", i, f);`.
Em *Python*, a formatação para os tipos também é necessária, mas dependendo do tipo de comando que usar para a **impressão** (*saída*), isso pode não ser claro, mas existe um formato de *impressão* que precisa explicitar os tipos (e.g. `%d` para inteior e `%f` para flutuante). Já para **leitura** (*entrada*), a partir da versão 3 do *Python*, deve-se utilizar uma função para converter para o tipo desejado (exceto quando desejar armazenar como sequência de caracteres). Nesse caso deve-se utilizar `int(input())` para converter para inteiro

e `float(input())` para flutuante.
Os demais tipos de variáveis tem outros formatadores.

Assim, ambas as linguagens usam como **formatador para inteiro o `%d`** e como **formatador para flutuante é o `%f`**.
A inversão desse formatadores em *C* provoca resultados mais "estranhos", quer dizer se tentar imprimir uma variável inteira para flutuante ou vice-versa, o resultado pode ser muito distindo do valor (e.g. `int a=7; printf("a=%f\n", a);` pode resultar valor *0*).
Em *Python* trocar o formatador de impressão geralmente não produz resultados tão inesperados.

Nos três exemplos a seguir ilustramos como é a leitura e impressão de variáveis inteiras e flutuantes. No exemplo 1 tratamos da linguagem *C* e nos exemplos 2 e 3, a linguagem *Python*.

Exemplo 1 [*C*]: ler 1 valor inteiro, ler 1 valor "real", imprimir na mesma linha os 2 valores lidos.

```
int v1; float v2; scanf("%d %f", &v1. &v2); printf("v1=%d, v2=%f\n", v1, v2);
```

Na linguagem *Python* não é necessário declarar as variáveis, mas ao ler um valor é preciso convertê-lo para o tipo a ser trabalhado, além disso, em impressões pode-se usar os formatadores para `%d` para inteiro e `%f` para flutuante.

Exemplo 2 [*Python*]: ler 1 valor inteiro, ler 1 valor "real", imprimir em linhas separadas os 2 valores lidos.

```
v1 = int(input()); v2 = float(input()); # ler 2 valores, o primeiro como int, o segundo como float
print("v1=", v1); print("v2=", v2); # note que neste caso havera uma quebra de linha entre v1 e v2
```

Exemplo 3 [*Python*]: ler 1 valor inteiro, ler 1 valor "real", imprimir na mesma linha os 2 valores lidos (usando diferente técnicas).

```
# No Python pode-se imprimir sem mudar de linha usando o parametro ' , end = ""' assim: print("Algo ", end = "");
# mas para isso precisa carregar a biblioteca 'print_function' como na linha abaixo
from __future__ import print_function # no Python 3 esta linha e' desnecessaria...

der main () : # declara uma funcao de nome 'main'
  v1 = 2; v2 = 5.0;
  # Tecnica 1 para imprimir na mesma linha: mais facil
  print("v1=", v1, "v2=", v2);

  # Tecnica 2: o %d e %f indica que ali virao valores 'int' e 'float', respectivamente,
  #            e o ultimo % indica que a seguir deve vir uma lista com os valores para %d e %f (nessa ordem)
  print("v1=%d, v2=%f\nxxx" % (v1, v2)); # dentro das aspas o \n forca uma quebra de linha => o xxx vai para a prox linha
```

```
  # Tecnica 3: permite fazer uma impressao e NAO quebrar a linha, ou seja, o
proximo 'print' vai para a mesma linha
  print("v1=", v1, ", v2=", v2, end = ""); # o ultimo parametro 'end=""' manda
nao quebrar a linha
  print(" ***");                          # portanto, esses asteriscos irao para
a mesma linha do 'v1= 5, v2= 2'

main(); # invoque a funcao 'main()'
```

2. Conversão de tipo em *C* (inteiro e "real")

Em *C* é preciso cuidado ao realizar operações com variáveis, por exemplo se realizar uma divisão entre variáveis (ou constante) inteiras, o resultado é um valor inteiro, mesmo que a variável armazenando o resultado seja "real" (flutuante).

Exemplo 4: usar variáveis `int v1` e `float v2` e fazê-las receber o resultado de `5 / 2`.

```
#include <stdio.h>
int main (void) {
  int v1;
  float v2;
  v1 = 5/2; // como 5 e 2 sao inteiro => faz-se a divisao inteira!
  v2 = 5/2; // idem! (ou seja, o fato de v2 ser float nao altera o resultado)
  printf("v1=%d, v2=%f\n", v1, v2); // o resultado sera': v1=2, v2=2.000000
  return 1; // se foi algum outro programa que invocou este, devolva o valor 1
  }
```

Assim, se o desejado é que `v2` receba o valor `2.5`, então deve-se usar um conversor de tipo (*type cast*), convertendo ou o `5` ou o `2` para *float*. Assim: `v2 = (float)5 / 2; // ou v2 = 5 / (float) 2`.

Agora se o interesse for obter o resto da divisão inteira, não é preciso de um truque "matemático", basta usar o operador especial `%`, como indicado no exemplo 5 abaixo.

Exemplo 5: um programa para imprimir o resto da divisão inteira entre dois inteiros.

```
#include <stdio.h>
int main (void) {
  int v1, v2;
  scanf("%d %d", &v1. &v2); // solicita que sejam digitados 2 valores inteiros

  printf("v1=%d, v2=%f\n", v1, v2); // imprime o que foi digitado

  // Supondo que foi digitado 5 para v1 e 2 para v2, o proximo 'printf' imprime:
resto da divisao entre 5 e 2 e': 1
  printf("resto da divisao entre %d e %d e': %d\n", v1, v2, v1 % v2); // note o
"v1 % v2"

  // pode-se fazer a impressao ou armazenar o resto da divisao usando constantes
  printf("%d\n", 5%2); // imprime: 1
  v1 = 5%2;            // v1 recebe o resto da divisao inteira de 5 por 2 (ou
seja, recebe 1)
  return 1;
  }
```

3. Conversão de tipo em *Python* (inteiro e "real")

Em *Python* existem vários opções para impressão, mas deve-se tomar cuidado com as variáveis/valores, pois dependendo do "tipo", o valor pode ser truncado. Outro cuidado que deve-se tomar é que, ao solicitar que o usuário digite um valor numérico, deve-se convertë-lo para o tipo desejado.

Exemplo 6: usar variáveis `v1` e `v2`, a primeira para inteiro (*int*) e a segunda para "real" (ou ponto flutuante, (*float*) e fazê-las receber o resultado de `5 / 2`.

```
der main () :
  v1 = 5/2;         # o resultado e' 2
  v2 = float(5)/2); # obrigatorio converter um dos operandos para conseguir 2.5 -
veja que "float(5/2)" resulta "2.0"

  # imprimir de modo mais simples
  print("v1=", v1, "v2=", v2);       # resultado: v1=2, v2=2.5

  # agora imprimindo usando formatadores %d e %f e uma lista de valores (v1, v2)
  print("v1=%d, v2=%d" % (v1, v2));  # resultado: v1=2, v2=2
  print("v1=%f, v2=%f" % (v1, v2));  # resultado: v1=2.000000, v2=2.500000
  print("v1=%d, v2=%f" % (v1, v2));  # resultado: v1=2, v2=2.500000
main();
```

Note que no exemplo acima, aparece `(v1, v2)` que é uma **lista** do *Python*, no caso com dois valores numéricos, o primeiro deles será lançado sobre o formatador (no caso o `%d`) e o segundo deles irá para o segundo formatador (no caso o `%f`).

Outra nota deve ser feita sobre o significado dos *formatadores*, o `%d` prepara uma saída do tipo *int*, enquanto o `%f` é para saída de valores "reais" (na verdade *float*). Assim, mesmo que determinada variável (`v2` no caso) armazene *float* se usar o formatador `%d` será impressa a parte inteira do número (ou seja, ele é **truncado**).

No exemplo abaixo, fazemos a leitura de um valor inteiro e de um valor "real". Precisamos usar as funções conversoras de tipo `int(...)` e `float(...)`, respectivamente para converter os valores lidos para `int` e para `float`, respectivamente.

Exemplo 7: usar variáveis `int v1` e `float v2`.

```
der main () :
  v1 = int(input());    # ler um valor do tipo inteiro
  v2 = float(intput()); # ler um valor do tipo "float"
  print("v1=%d, v2=%d" % (v1, v2));  # imprimir assim, o v2 sera' truncado para
inteiro
  print("v1=%d, v2=%f" % (v1, v2));  # aqui imprime corretamente v1 como "int" e
v2 como "float"
  print("v1=" + str(v1) + ", v2=" + str(v2)); # pode-se imprimir concatenando
"strings" - a funcao "str(...)" converte valor para "string"
main();
```

Do mesmo modo que *C*, a linguagem *Python* tem um operador que permite obter o resto da divisão inteira, que é o operador especial `%`, como indicado no exemplo 8 abaixo.

Exemplo 8: um programa para imprimir o resto da divisão inteira entre dois inteiros.

```
der main () :
  v1 = int(input());    # ler um valor do tipo inteiro
  v2 = float(intput()); # ler um valor do tipo "float"
  print("v1=%d, v2=%d" % (v1, v2));  # o que foi digitado

  print("resto da divisao entre %d e %d e': %d" % (v1, v2, v1 % v2)); # apos o
'%' deve vir uma lista (v1,v2,v1%v2)

main();
```

Leônidas de Oliveira Brandão
http://line.ime.usp.br

Alterações:
2021/06/14: numeração de seções e pequenas correções;
2019/07/31: redesenhada a estrutura da página e vários pequenas correções;
2018/04/02: versão inicial;