Programação Estruturada

Aula 9 - Tipos

Yuri Malheiros (yuri@ci.ufpb.br)

Caracteres

- Em C existe um tipo específico para representar caracteres
    - `char`

```
char ch

ch = 'a';
ch = 'A';
ch = '0';
```

- Caracteres são escritos entre aspas simples

Caracteres

- Caracteres são tratados como números
  - Eles são codificados em binário
- Em ASCII os caracteres são codificados usando 7 bits
  - `'a'` - 110 0001 - 97

```
char ch;
ch = 97;

printf("%c", ch);
```

Caracteres

- Podemos comparar caracteres como números
  - `'a' < 'b'` retorna `1`

Caracteres - exercício

- Faça um programa que imprime todas as letras minúsculas do alfabeto

Lendo e escrevendo caracteres

- O especificador de conversão para caracteres é o `%c`

```
char ch;

scanf("%c", &ch);
printf("%c", ch);
```

Lendo e escrevendo caracteres

- C possui funções específicas para ler e escrever caracteres
  - `putchar(ch)` - escreve um caractere
  - `getchar(ch)` - ler e retorna um caractere

```
char ch, ch2;

ch = getchar();
ch2 = getchar();

putchar(ch);
putchar(ch2);
```

Lendo e escrevendo caracteres

- `putchar` e `getchar` são mais rápidas que `printf` e `scanf`
- `getchar` retorna um valor, assim podemos usar ela em expressões

Lendo e escrevendo caracteres - exercício

- Faça um programa que recebe uma frase e exibe o seu tamanho (quantidade de caracteres)
    - A frase deve terminar com a quebra de linha `\n`

Conversão de tipos

- É possível misturar tipos diferentes numa expressão

```
printf("%f", 2.5 + 3);
```

- O compilador automaticamente realiza conversões implícitas

Conversão de tipos

- Quando os operandos não têm o mesmo tipo em operações aritméticas, o compilador realiza conversões implícitas
- Ao somar um inteiro e um float é mais seguro converter o inteiro para float que o contrário

Conversão de tipos

- Normalmente o compilador converte o tipo do operando mais restrito para o tipo do outro operando
    - Esta operação é chamada de promoção

Conversão de tipos

- Se os dois operandos forem de ponto flutuante, temos o seguinte diagrama de promoção:
    - float → double → long double

Conversão de tipos

- Se os dois operandos forem inteiros, temos o seguinte diagrama de promoção:
    - int → unsigned int → long int → unsigned long int

Conversão de tipos

- Tipos `char` e `short` são convertidos para `int` em operações aritméticas
- Tipos inteiros são mais restritos que os tipos de ponto flutuante

Conversão de tipos

- Cuidado ao misturar `int` e `unsigned int` em expressões
- Como o `int` é convertido para `unsigned int`, se o `int` for negativo, os resultados podem ser inesperados:

```c
int x = -3;
unsigned int y = 9;

printf("%d\n", x < y);
```

Conversão de tipos

- C também faz conversões implícitas durante a atribuição
- A valor à direita é convertido para o tipo da variável à esquerda
  - `int x = 'a';` - o caractere `'a'` é convertido para `int`

Conversão de tipos

- Atribuir um valor para um tipo mais restrito pode causar problemas
- `int x = 10.25;` - a parte decimal é descartada
- `char c = 1000;` - errado
- `int x = 1.0e20` - errado

Casting

- Em alguns casos, precisamos controlar a conversão de tipos
- Para isso, a linguagem C fornece operações de cast
    - `(nome do tipo) expressão`

Casting

```c
int main(void) {
    float f, f_part;
    f = 2.53;

    f_part = f - (int) f ;

    printf("%f\n", f_part);
}
```

- `(int) f` converte `f` para `int`

Casting

- Utilizando o cast, forçamos uma conversão

```
float q;
int n, d;

n = 3;
d = 2;

q = n/d;

printf("%f\n", q);
```

Casting

- Utilizando o cast, forçamos uma conversão

```
float q;
int n, d;

n = 3;
d = 2;

q = (float) n/d;

printf("%f\n", q);
```

Casting

- Cast pode ser usado também para evitar overflow

```
long i;
int j = 1000000;

i = (long) j*j;

printf("%ld\n", i);
```

Definição de tipos

- Utilizando a palavra reservada `typedef` podemos criar nossos próprios tipos

```
typedef int Bool;
Bool flag;
```

- O compilador trata `Bool` como sinônimo de `int`
- `flag` não é nada mais que uma variável do tipo `int`

Definição de tipos

- Definições de tipos podem tornar o programa mais fácil de entender
- Suponha um programa que tem as variáveis `pagamento` e `troco`
- Poderíamos ter um tipo `Dinheiro` para elas

```
typedef float Dinheiro;

Dinheiro pagamento, troco;
```

Definição de tipos

- O programa também pode ficar mais fácil de manter
- Imagine que depois de um tempo seja necessário trocar a representação de `Dinheiro` para `double` ao invés de `float`
- Para isso bastaria modificar o `typedef`
    - `typedef double Dinheiro`

Operador sizeof

- O operador `sizeof` retorna o número de bytes que um tipo usa para guardar os seus valores
  - `sizeof(nome do tipo)`
- Também podemos usar o `sizeof` com constantes ou expressões
  - `sizeof(4)`
  - `sizeof(2.5)`
- O valor retornado é um `unsigned int`, então para imprimí-lo use a conversão `%lu`