Aula 09

MC 102 - Algoritmos e Programação de Computadores

Conversão entre tipos numéricos. Identação e outras dicas para programação.

Conversão de Tipos

C faz conversão automatica de tipos no cálculo de expressões aritméticas x <op> y

- x e y são de mesmo tipo, o resultado é deste tipo
- x e y são de tipos diferentes, um deles será convertido (durante o cálculo) para o tipo do outro, que será o tipo do resultado

Em alguns casos, há conversão mesmo quando ambos são do mesmo tipo

Note: a conversão é temporária; o tipo da variável não é alterado

Conversão de Tipos

Operadores do mesmo tipo:

Tipo da variável	Conversão para
char e short	int
unsigned char e unsigned short	short

Conversão de Tipos

Operadores do tipo diferentes:

Seguem a hieraquia de tipos, na qual o operador de menor tipo é convertido para o de maior, que passará a ser o tipo da expressão:

int < unsigned < long < unsigned long < float < double

Conversão de Tipos: Exemplo

char c; long l; double d; float f;
int i; short s; unsigned u;

Expressão	Tipo
c – s / i	int
u * 3 – i	unsigned
u * 3.0 – i	double
f * 3.0 – i	float
c + 1	int
c + 1.0	double
3 * s * l	long

int < unsigned < long < unsigned long < float < double

char e short => int

Conversão de Tipos

A conversão automática (implícita) também ocorre em atribuições

```
int a = 3, b;
double c, d = 2.4;
c = a;
b = d; -> é implícita, mas há perda de
          informação
```


Conversão de Tipos

Nas expressões isso nem sempre funciona

```
int a = 3, b= 2;
double c, d = 2.4;

c = (a + 0.0)/b; -> conversão automática
d = (a + 0.0)/b;

c =  (a / b); -> ocorre perda de informação
```


Conversão Explícita (cast)

<var tipo1> = (tipo1) <expressão>

```
c = (double) a/b;  -> sem perda de informação
c = (double) (a/b);-> com perda de informação
```

O cast pode ser usado também em expressões:

```
k = (int) ((int) x + (double) i + j)
c = (char) (3 - 3.14 * x);
f = (float) (x = 77)
```

~~f = (float) x = 77;~~

Conversão Explícita (cast)

Mesmo em conversão implícita, o cast pode ser usado

```
int a;
float f=3.1, g=1.9;

a = f/g;  /* = floor( f/g ) */

a = ((int) f)/((int) g);/* = floor(f) /
                           floor(g) */
```


Identação

Tão importante quanto escrever um programa é torná-lo inteligível e compreensível a outra pessoa.

Mas como identar corretamente um programa? Quais as principais regras a seguir?

Deve-se sempre usar o bom senso...

Identação

Geralmente, indica dependência para a execução de um comando ou de um grupo de comandos para outro.

Exemplo não identado:

```
printf("Digite 10 numeros (0 p/ finalizar).\n");
i = 1;
while (i <= 10) {
printf("%do.numero: ", i);
scanf("%d", &x);
if (x == 0)
break;
soma += x;
i++;
}
```


Identação

Exemplo identado (as cores ilustram a dependência):

```
printf("Digite 10 numeros (0 p/ finalizar).\n");
i = 1;
while (i <= 10) {
   printf("%do.numero: ", i);
   scanf("%d", &x);
   if (x == 0)
      break;
   soma += x;
   i++;
}
```

Deixa claro a dependência entre os comandos internos de um comando condicional ou de repetição

Identação

1) Procurar definir um comando por linha

Exemplo não identado:

```
a = 1; b = 2;
if (a < b) c = a + b;
else c = a - b;
c = c / 2;
```

Exemplo identado:

```
a = 1;
b = 2;
if (a < b)
  c = a + b;
else
  c = a - b;
c = c / 2;
```


Identação

2) Bloco de comandos (definidos por "{" e "}") ou comandos simples dependentes de uma condição devem ser deslocados para a direita (sempre usando a mesma quantidade de espaços ou tabulação)

Exemplo não identado:

```
if (a < b) c = a + b;
else
{ c = a - b;
while (c > 0)
b = b * a - 1; }
```

Exemplo identado:

```
if (a < b)
  c = a + b;
else {
  c = a - b;
  while (c > 0)
    b = b * a - 1;
}
```


Identação

3) Todo "{" deve estar na mesma linha do comando condicional ou de repetição, e o "}" referente deve estar na mesma coluna, definido sozinho

Exemplo não identado:

```
if (a < b) { c = a +
  b;
  b = a; }
else
{ c = a - b;
while (c > 0) {
b = b * a - 1;
a = a + 2; } }
```

Exemplo identado:

```
if (a < b) {
  c = a + b;
  b = a;
}
else {
  c = a - b;
  while (c > 0) {
    b = b * a - 1;
    a = a + 2;
  }
}
```


Cometários

É importante documentar o código

- facilita entendimento
- explica o que está sendo feito

Em C há duas formas de comentar o código

// o que está após // e na mesma linha é comentário e portanto não será processado

```
int a; // esta variavel será usada como contador
```

Nos C89 e C90 ocorre mensagem de aviso quando usa-se //

Cometários

/* */ o que estiver entre /* */ é comentário e portanto não será processado, mesmo que ocupe várias linhas

```
/*
* Este programa realiza o calculo da media de n
* numeros dados
*/
#include <stdio.>
```


Esqueleto do Programa

Cabeçalho

#include, #define

Corpo

int main ( ) {

definição das variáveis

inicialização das variáveis

lógica para resolver o problema

return 0;

}

Outras dicas úteis

Antes de pensar no programa, escreva o algoritmo desejado para resolver o problema.

Verifique se o algoritmo realmente resolve o problema (com casos reais) para então escrever o programa.

Faça um teste de mesa!