MC-102 — Aula 04
Escrita, Leitura e Operações Aritméticas

Instituto de Computação – Unicamp

Primeiro Semestre de 2012

Escrevendo na tela

- Podemos imprimir um de texto puro utilizando o comando `printf`. O texto pode ser uma constante do tipo string.

Exemplo

```
printf("Ola Pessoal!");
```

Saída: `Ola Pessoal!`

- No meio da constante string pode haver comandos especiais. O símbolo especial `\n` é responsável por pular uma linha na saída.

Exemplo

```
printf("Ola Pessoal!  \n Ola Pessoal");
```

Saída:
```
Ola Pessoal!
Ola Pessoal
```

Escrevendo o conteúdo de uma variável na tela

- Podemos imprimir, além de texto puro, o conteúdo de uma variável utilizando o comando `printf`. Para isso, utilizamos símbolos especiais no texto para representar que aquele trecho deve ser substituído por uma variável e, no final, passamos uma lista de variáveis ou constantes, separadas por vírgula.

Exemplo

```
printf("A variável %s contém o valor %d","a", a);
```

imprime `A variável a contém o valor 10`

- Nesse caso, %s deve ser substituído por uma variável ou constante do tipo `string` enquanto %d deve ser substituído por uma variável ou constante do tipo inteiro.

Formatos inteiros

%d — Escreve um inteiro na tela.

Exemplo

```
printf ("%d", 10);
```

imprime 10

Exemplo

```
int a=12;
printf ("O valor e %d", a);
```

imprime O valor e 12

Formatos inteiros

- A letra d pode ser substituida pelas letras u e ld, quando desejamos escrever variáveis do tipo unsigned ou long, respectivamente.

Exemplo

```
printf ("%d", 4000000000);
```

escreve -294967296 na tela, enquanto que

```
printf ("%ld", 4000000000);
```

escreve 4000000000.

Formatos ponto flutuante

`%f` — Escreve um ponto flutuante na tela.

Exemplo

```
printf ("%f", 10.0);
```

imprime `10.000000`

Formatos ponto flutuante

%e — Escreve um ponto flutuante na tela, em notação científica

Exemplo

```
printf ("%e", 10.02545);
```

imprime `1.002545e+01`

Formatos ponto flutuante

%.< *decimais* >f — Escreve um ponto flutuante na tela, com < *decimais* > casas decimais.

Exemplo

```
printf ("%.2f", 10.1111);
```

imprime `10.11`

Formatos ponto flutuante

- A letra f pode ser substituida pelas letras lf, para escrever um `double` ao invés de um `float`

Exemplo

```
printf ("%6.2lf", 10.0);
```

imprime < *espaco* >10.00

Formato caracter

%c — Escreve uma letra.

Exemplo

```
printf ("%c", 'A');
```

imprime a

Note que `printf ("%c", 65)` também imprime a letra A.

Formato string

%s — Escreve uma string

Exemplo

```
printf ("%s", "Meu primeiro programa");
```

imprime `Meu primeiro programa`

A função `scanf`

- Realiza a leitura de um texto a partir do teclado.
- Parâmetros:
  - Uma string, indicando os tipos das variáveis que serão lidas e o formato dessa leitura.
  - Uma lista de variáveis.
- Aguarda que o usuário digite um valor e atribui o valor digitado à variável.

A função scanf

O programa abaixo é composto de quatro passos:

1. Cria uma variável `n`;
2. Escreve na tela `Digite um número:`
3. Lê o valor do número digitado
4. Imprime o valor do número digitado

```
#include <stdio.h>
int main(){
  int n;
  printf("Digite um número: ");
  scanf("%d",&n);
  printf("O valor digitado foi %d\n",n);
}
```

A função scanf

Leitura de várias variáveis:

```
#include <stdio.h>
main(){
   int m, n, o;
   printf("Digite três números: ");
   scanf("%d %d %d",&m, &n, &o);
   printf("O valores digitados foram\
           %d %d %d\n", m, n, o);
}
```

Formatos de leitura de variável

Os formatos de leitura são muito semelhantes aos formatos de escrita utilizados pelo `printf`. A tabela a seguir mostra alguns formatos possíveis de leitura

Código	Função
%c	Lê um único caracter
%s	Lê uma série de caracteres
%d	Lê um número decimal
%u	Lê um decimal sem sinal
%ld	Lê um inteiro longo
%f	Lê um número em ponto flutuante
%lf	Lê um double

Expressões

- Já vimos que constantes e variáveis são expressões.
- Uma expressão também pode ser é um conjunto de operações aritméticas, lógicas ou relacionais utilizados para fazer "cálculos" sobre os valores das variáveis.

Exemplo

```
a + b
```

Calcula a soma de a e b

Expressões Aritméticas

- Os operadores aritméticos são: +,-,*,/
- $<$ *expressao* $>$ + $<$ *expressao* $>$: Calcula a soma de duas expressões.
  Ex: `a + b;`
- $<$ *expressao* $>$ - $<$ *expressao* $>$: Calcula a subtração de duas expressões.
  Ex: `a - b;`
- $<$ *expressao* $>$ * $<$ *expressao* $>$: Calcula o produto de duas expressões.
  Ex: `a * b;`

Expressões

- $<$ *expressao* $>$ / $<$ *expressao* $>$: Calcula a divisão de duas expressões.
  Ex: `a / b;`
- $<$ *expressao* $>$ % $<$ *expressao* $>$: Calcula o resto da divisão (inteira) de duas expressões.
  Ex: `a % b;`
- - $<$ *expressao* $>$: Inverte o sinal da expressão.
  Ex: `-b;`

Expressões

- As expressões aritméticas (e todas as expressões) operam sobre outras expressões.
- É possível compor expressões complexas como por exemplo:
  `a = b + 2 + c+(9+d*8)`

Qual o valor da expressão **5 + 10 % 3**?
E da expressão **5 * 10 % 3**?

Precedência

- Precedência é a ordem na qual os operadores serão calculados quando o programa for executado. Em C, os operadores são calculados na seguinte ordem:
  - * e /, na ordem em que aparecerem na expressão.
  - %
  - + e -, na ordem em que aparecerem na expressão.
- Exemplo: 8+10*6 é igual a 68.

Alterando a precedência

- ($<$ *expressao* $>$) também é uma expressão, que calcula o resultado da expressão dentro dela para só então permitir que as outras expressões executem.
  Ex: `5 + 10 % 3` retorna 6, enquanto `(5 + 10) % 3` retorna 0
- Você pode usar quantos parênteses desejar dentro de uma expressão, contanto que utilize o mesmo número de parênteses para abrir e fechar expressões.
- **OBS:** Use sempre parênteses em expressões para deixar claro em qual ordem a expressão é avaliada.

Incremento(++) e Decremento(−−)

- Operadores de incremento e decremento tem duas funções: servem como uma expressão **e** incrementam ou decrementam o valor da variável ao qual estão associados em uma unidade. Ex: c++ — incrementa o valor da variável c em uma unidade
- Dependendo da posição do operador de incremento e decremento, uma função é executada antes da outra.

Incremento(++) e Decremento(−−)

- Operador a esquerda da variável: Primeiro a variável é incrementada, depois a expressão retorna o valor da expressão. Ex:

```
#include <stdio.h>
main () {
  int a = 10;
  printf ("%d", ++a);
}
```

Imprime 11

Incremento(++) e Decremento(−−)

- Operador a direita da variável: Primeiro a expressão retorna o valor da variável, e depois a variável é incrementada. Ex:

```
#include <stdio.h>
int main (void) {
  int a = 10;
  printf ("%d", a++);
}
```

Imprime 10

Incremento(++) e Decremento(−−)

- Em uma expressão, os operadores de incremento e decremento são sempre calculados primeiro (tem maior precedência)

```
#include <stdio.h>
int main (void) {
  int a = 10;
  printf ("%d", a * ++a);
}
```

Imprime 121

Atribuições simplificadas

Uma expressão da forma

```
a = a + b
```

onde ocorre uma atribuição a uma das variáveis da expressão pode ser simplificada como

```
a += b
```

Atribuições simplificadas

Comando	Exemplo	Corresponde a:
+=	a += b	a = a + b;
-=	a -= b	a = a - b;
*=	a *= b;	a = a * b;
/=	a /= b;	a = a / b;
%=	a %= b;	a = a % b;

Conversão de tipos

- É possível converter alguns tipos entre si.
- Existem duas formas de fazê-lo: implícita e explícita:
- Implícita
  - Capacidade (tamanho) do destino deve ser maior que a origem senão há perda de informação.
    Ex.: `int a; short b; a = b;`
    Ex:`float a; int b=10; a = b;`
- Explícita:
  - Explicitamente informa o tipo que o valor da variável ou expressão é convertida.
    Ex. `a = (int)( (float)b / (float)c );`
  - Não modifica o tipo "real" da variável, só o valor de uma expressão.
    Ex. `int a; (float)a=1.0;` ← Errado

Um uso da conversão de tipos

A operação de divisão (/) possui dois modos de operação de acordo com os seus argumentos: inteira ou de ponto flutuante.

- Se os dois argumentos forem inteiros, acontece a divisão inteira. A expressão `10 / 3` tem como valor 3.
- Se um dos dois argumentos forem de ponto flutuante, acontece a divisão de ponto flutuante. A expressão `1.5 / 3` tem como valor `0.5`.

Quando se deseja obter o valor de ponto flutuante de uma divisão (não-exata) de dois inteiros, basta converter um deles para ponto flutuante:

Exemplo

A expressão `10 / (float) 3` tem como valor `3.33333333`

Um parêntese: comentários

- O código fonte pode conter comentários direcionados unicamente ao programador. Estes comentários devem estar delimitados pelos símbolos /* e */, e são ignorados pelo compilador.

Exemplo

```
#include <stdio.h>

/* Este é o meu primeiro programa. */
//Isto tambem é um comentário
main() {
  printf("Hello, world!\n");
}
```

Um parêntese: comentários

- Comentários são úteis para descrever o algoritmo usado e para explicitar suposições não óbvias sobre a implementação.