

IPC2

1999/2000

F. Nunes Ferreira

Acetatos baseados no livro

C: How to Program (second edition)

H. M. Deitel

P. J. Deitel

Prentice Hall, 1994

Introdução

IPC2

1999/2000

F. Nunes Ferreira

Acetatos baseados no livro

C: How to Program (second edition)

H. M. Deitel

P. J. Deitel

Prentice Hall, 1994

Introdução

Um primeiro programa em C

```
/* Um programa muito simples */

#include <stdio.h>

int main(void)
{
    printf("Bem-vindo ao C.\n");

    return 0;
}
```

Bem-vindo ao C.

Introdução

Um primeiro programa em C

- ◆ PELO MENOS, um comentário impõe-se antes de uma

função: `/* Um comentário */`

- ◆ `#include <stdio.h>`

`printf()` é da biblioteca *standard* de *io*

`printf("Uma cadeia de caracteres...");`

- ◆ A execução do programa começa na função ***main***
- ◆ **`return 0`** : indicação ao sist. operativo que o programa terminou bem

Introdução

Algumas sequências de *escape*... e o *inteiro* associado

10 \n Nova linha. Posiciona o cursor no início da próxima linha

9 \t Tab horizontal. Posiciona cursor no próximo tab

8 \b Uma posição para trás

13 \r Retorno ao início de linha. Mantém-se na mesma linha

7 \a Alerta sonoro.

```
Printf("cadeia\n");
```

92 \\ Imprime um \

34 \" Imprime um "

0 \0 character nulo

Introdução

Um programa em C com entrada de dados

```
/* Soma de inteiros */  
#include <stdio.h>  
int main(void)  
{  
    int num1, num2, soma;  
  
    printf("Primeiro inteiro\n");  
    scanf("%d", &num1);  
    printf("Segundo inteiro\n");  
    scanf("%d", &num2);  
  
    soma = num1 + num2;  
    printf("A soma e' %d\n", soma);  
    return 0;  
}
```

```
Primeiro inteiro  
32  
Segundo inteiro  
17  
A soma e' 49
```

Introdução

Um programa em C com entrada de dados

- ◆ Nomes de variáveis

primeiroInteiro ou *primeiro_inteiro*

- ◆ especificador de conversão... na cadeia de controlo

```
scanf("%d", &num1);
```

```
printf("A soma e' %d\n", soma);
```

- ◆ Atribuição

```
soma = num1 + num2;
```

Introdução

Aritmética em C

◆ Operadores aritméticos


Adição	+	Subtracção	-
Multiplicação	*	Divisão	/
Módulo	%		

◆ Prioridades

()	Alta
* / %	
+ -	Baixa

◆ $7 / 4 = 1$

$17 / 5 = 3$

 **$17 / 5.0 = 3.4$**

◆ $7 \% 4 = 3$

$17 \% 5 = 2$

Introdução

Decisões

◆ Operadores de relação

igual	==	diferente	!=
maior que	>	menor que	<
maior ou igual	>=	menor ou igual	<=

◆ if (num1 == num2)

```
printf("%d e' igual a %d\n", num1, num2);
```

◆ if (num1 != num2)

```
printf("%d nao e' igual a %d\n", num1, num2);
```

Introdução

Prioridades e associatividade dos operadores

◆ Prioridade decrescente

()

* / %

+ -

< <= > >=

== !=

=

Associatividade

Esquerda para a direita

Esquerda para a direita

Esquerda para a direita

Esquerda para a direita

Esquerda para a direita

Direita para a esquerda

◆ Utilizar () em caso de dúvidas

Exercício

- ◆ Escrever um programa em C que lê 3 inteiros através do teclado e imprime a soma, a média, o produto, o menor e o maior dos inteiros lidos.

```
Indicar 3 inteiros: 13 27 14
```

```
A soma e': 54
```

```
A media e': 18
```

```
O produto e': 4914
```

```
O menor e': 13
```

```
O maior e': 27
```

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

int main(void)
{
    int prim, seg, terc;
    int menor, maior;

    printf("\n\n\nIndicar 3 inteiros: ");
    scanf("%d%d%d", &prim, &seg, &terc);
    printf("\nA soma e': %d", prim + seg + terc);
    printf("\nA media e': %d", (prim + seg + terc) / 3);
    printf("\nO produto e': %d", prim * seg * terc);

    menor = maior = prim;

    if (seg < menor)
        menor = seg;
    if (terc < menor)
        menor = terc;

    if (seg > maior)
        maior = seg;
    if (terc > maior)
        maior = terc;

    printf("\nO menor e': %d", menor);
    printf("\nO maior e': %d", maior);

    return 0;
}
```

Introdução

printf() e scanf()

```
#include <stdio.h>
int main(void)
{
    char    c1, c2, c3;
    int     i;
    float   x;
    double  y;
    printf("\n%s\n%s",
           "Escrever 3 caracteres,",
           "um int, um float, e um double: ");
    scanf("%c%c%c%d%f%lf", &c1, &c2, &c3, &i, &x, &y);
    printf("\nImprimir dados fornecidos:\n");
    printf("%3c%3c%3c%5d%17.5e%17.5f", c1, c2, c3, i, x, y);

    return 0;
}
```

Introdução

`printf()` e `scanf()`

```
Escrever 3 caracteres,  
um int, um float, e um double: ABC 3 55 77.7
```

```
Imprimir dados fornecidos:
```

```
A B C      3      5.50000e+01      77.70000
```

```
Escrever 3 caracteres,  
um int, um float, e um double: A d 2 55e2 6.6e-3
```

```
Imprimir dados fornecidos:
```

```
A      C      2      5.50000e+03      0.00660
```

Introdução

`printf()` e `scanf()`

`printf()`

<code>%c</code>	character
<code>%d</code>	inteiro decimal
<code>%s</code>	cadeia de caracteres
<code>%f</code>	vírgula flutuante
	vírgula flutuante
	vírgula flutuante
<code>%e</code>	vírg. flut. formato-e
<code>%g</code>	formato-e ou formato-f (+curto)

`scanf()`

<code>%c</code>
<code>%d</code>
<code>%s</code>
<code>%f</code> (<i>float</i>)
<code>%lf</code> (<i>double</i>)
<code>%LF</code> (<i>long double</i>)

Introdução

Erros mais comuns

- ◆ Esquecer de terminar um comentário com `*/`
- ◆ Começar um comentário com `*/` ou terminá-lo com `/*`
- ◆ Utilizar letras maiúsculas impropriamente
(Ex. *Main* em vez de *main*)
- ◆ Colocar declarações de variáveis no meio de instruções executáveis
- ◆ Esquecer `"` na cadeia de controlo de formato de uma instrução *printf* ou *scanf*...
- ◆ ...

Introdução

Erros mais comuns

- ◆ ...
- ◆ Esquecer % na especificação de conversão da cadeia de controlo de formato, de uma instrução *printf* ou *scanf*...
- ◆ Colocar \n fora da cadeia de controlo de formato de uma instrução *printf* ou *scanf*...
- ◆ Esquecer as expressões cujos valores deverão ser para imprimir e cujos especificadores de conversão estão num *printf*
- ◆ ...

Introdução

Erros mais comuns

- ◆ ...
- ◆ Esquecer `&` antes da variável que recebe um valor num *scanf*
- ◆ Separar por espaço algum dos operadores `==`, `!=`, `>=` ou `<=`
- ◆ Confundir o operador de igualdade `==` com o operador de atribuição `=`

Introdução

Regras de boa programação

- ◆ Iniciar uma função com um bom comentário
- ◆ Usar correcta e sistematicamente a indentação
(*cada nível 3 espaços*)
- ◆ Utilizar `<stdio.h>` sempre que o programa inclua funções da biblioteca standard de *io*
- ◆ Nomes adequados para as variáveis diminuem a necessidade de comentários
- ◆ ...

Introdução

Regras de boa programação

- ◆ ...
- ◆ Utilizar uma das seguintes convenções para nomes multi-palavra: *valorTotal* ou *valor_total*
- ◆ Colocar um espaço de cada lado de um operador de dois operandos
- ◆ Evitar mais do que uma instrução por linha
- ◆ Atenção à prioridade dos operadores. Em caso de dúvida, utilizar parêntesis

Introdução

Estruturas de controlo

- ◆ Estrutura de
selecção simples

```
if (classificacao >= 10)
    printf("Passou\n");
```

- ◆ Estrutura de
selecção dupla

```
if (classificacao >= 10)
    printf("Passou\n");
else
    printf("Falhou\n");
```

- ◆ Operador condicional

```
printf("%s\n", classifica >= 10 ? "Passou" : "Falhou")
```

condição ? consequência : alternativa

Introdução

Estruturas de controlo (cont.)

◆ Estrutura de *selecção múltipla*

```
if (classificacao >= 18)
    printf("Excelente\n");
else
    if (classificacao >= 16)
        printf("Muito Bom\n");
    else
        if (classificacao >= 14)
            printf("Bom\n");
        else
            if (classificacao >= 12)
                printf("Suficiente\n");
            else
                if (classificacao >= 10)
                    printf("Sofrivel\n");
                else
                    printf("Fraco\n");
```

Outra forma equivalente, mas muito mais legível



Introdução

Estruturas de controlo (cont.)

◆ Estrutura de *selecção múltipla*

```
if (classificacao >= 18)
    printf("Excelente\n");
else if (classificacao >= 16)
    printf("Muito Bom\n");
else if (classificacao >= 14)
    printf("Bom\n");
else if (classificacao >= 12)
    printf("Suficiente\n");
else if (classificacao >= 10)
    printf("Sofrivel\n");
else
    printf("Fraco\n");
```

Introdução

Estruturas de controlo (cont.)

- ◆ Estrutura de *repetição*

```
saco = 13;  
while (saco > 0) {  
    printf("\nSaco = %d", saco);  
    saco = saco - 2;  
}
```

- ◆ Outras estruturas a ver mais tarde:

- ◆ Estrutura de repetição *for*

- ◆ Estrutura de selecção múltipla *switch*

Exercício

- ◆ Escrever um programa em C que lê um determinado número de valores a partir do teclado e, no final, indica qual foi o maior e o menor.

```
Quantos valores serao analisados? 4
```

```
Valor: 4
```

```
Valor : 34
```

```
Valor : -45
```

```
Valor : 15
```

```
34 foi o maior e -45 o menor.
```

```
#include <stdio.h>
#include <limits.h>
int main(void)
{
    int menor, maior, nValores, valorLido;

    printf("\nQuantos valores serao analisados? ");
    scanf("%d", &nValores);

    menor = INT_MAX;
    maior = INT_MIN;

    while ( nValores > 0 ) {
        printf("\nValor: ");
        scanf("%d", &valorLido);
        if (menor > valorLido)
            menor = valorLido;
        if (maior < valorLido)
            maior = valorLido;

        nValores--;
    }

    printf("\n\n%d foi o maior e %d o menor", maior, menor);

    return 0;
}
```

Introdução

```
#include <limits.h>
```

◆ Os *limites* mais usuais dos inteiros

INT_MIN <= -32767

INT_MAX >= +32767

UINT_MAX >= 65535

LONG_MIN <= -2147483647

LONG_MAX >= +2147483647

ULONG_MAX >= 4294967295

->(expt 2 16)
65536

->(expt 2 32)
4294967296

Exercício

- ◆ Escrever um programa em C que lê vários valores a partir do teclado e, no final, indica qual foi o maior e o menor.

```
Valores analisar os valores...
```

```
Valor (-1000 e' para terminar): 4
```

```
Valor (-1000 e' para terminar): 34
```

```
Valor (-1000 e' para terminar): -45
```

```
Valor (-1000 e' para terminar): 15
```

```
Valor (-1000 e' para terminar): -1000
```

```
34 foi o maior e -45 o menor.
```