

MC102 – Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Segundo Semestre de 2013

Roteiro

- 1 Maior número
- 2 Soma de n números
- 3 Fatorial
- 4 Máximo Divisor Comum (MDC)
- 5 Números primos
- 6 Fatoração em números primos
- 7 Números de Fibonacci
- 8 Conversão de números binários para decimais
- 9 Conversão de números decimais para binários

Introdução

- Vimos quais são os comandos de repetição em C.
- Veremos agora alguns exemplos de utilização desses comandos.

Maior número

- Vamos escrever um programa que recebe n números ($n \geq 1$) e informa qual deles é o maior.
- O programa deve ter os seguintes passos:
 - 1 Lê um número n .
 - 2 Repete n vezes a leitura de um número, determinando o maior.
- Como determinar o maior?

Maior número

- A ideia é criarmos uma variável maior que sempre armazena o maior número lido até então.

```
maior(n)
  ler um número maior
  repetir n-1 vezes
    ler um numero aux
    se aux > maior então
      maior = aux
  imprimir maior
```

Maior número

```
#include <stdio.h>

int main() {
    int i, n;
    double maior, aux;

    printf("Digite a quantidade de numeros: ");
    scanf("%d", &n);

    printf("Digite um numero: ");
    scanf("%lf", &maior);
    for (i = 1; i < n; i++) {
        printf("Digite um numero: ");
        scanf("%lf", &aux);
        if (aux > maior)
            maior = aux;
    }
    printf("Maior numero: %.2f\n", maior);
    return 0;
}
```

Soma de n números

- Vamos escrever um programa que recebe n números do teclado e informa a soma destes.
- Uma variável `soma` irá armazenar a soma dos números lidos.
- Ao ler um próximo número, como atualizar a soma?
 - ▶ Basta fazer: `soma = soma + numero;`

Soma de n números

```
#include <stdio.h>

int main() {
    int i, n, aux, soma = 0;

    printf("Quantidade de numeros: ");
    scanf("%d", &n);

    for (i = 1; i <= n; i++) {
        printf("Digite um numero: ");
        scanf("%d", &aux);
        soma = soma + aux;
    }

    printf("Soma dos numeros: %d\n", soma);
    return 0;
}
```


Fatorial

- Vamos escrever um programa que lê um número inteiro positivo n do teclado e informa qual o seu fatorial.
- O programa deve ter os seguintes passos:
 - 1 Lê um número n .
 - 2 Calcula $n * (n - 1) * \dots * 2 * 1$
- Como fazer este cálculo?
- Note que n não é fixo, portanto, temos que usar comandos de repetição.

Fatorial

- A ideia é criarmos uma variável `fatorial` que na i -ésima iteração do laço vale $i!$
- Note que $(i + 1)! = (i + 1) \times i!$, portanto, na $(i + 1)$ -ésima iteração podemos fazer `fatorial = (i+1) * fatorial`

```
fatorial(n)
  fatorial = 1
  para i de 1 até n faça
    fatorial = fatorial * i
  imprimir fatorial
```

Fatorial

```
#include <stdio.h>

int main() {
    int contador, n;
    unsigned int fatorial = 1;

    printf("Digite numero: ");
    scanf("%d", &n);

    for (contador = 1; contador <= n; contador++) {
        fatorial = fatorial * contador;
    }

    printf("%d! = %d\n", n, fatorial);

    return 0;
}
```

Fatorial

- No exemplo anterior, o fatorial é calculado corretamente para $n \leq 14$, entretanto, falha para $n \geq 15$.
- Por quê?

Solução:

- Podemos trocar o tipo da variável fatorial de `unsigned int` para `unsigned long int` ou mesmo por `double`.
- Com `unsigned long int` é possível calcular fatoriais até 20, enquanto que, com `double`, até 170 (neste caso, com perda de precisão numérica).

Fatorial

```
#include <stdio.h>

int main() {
    unsigned short int contador, n;
    unsigned long int fatorial = 1;

    printf("Digite numero: ");
    scanf("%hu", &n);

    for (contador = 1; contador <= n; contador++) {
        fatorial = fatorial * contador;
    }

    printf("%hu! = %lu\n", n, fatorial);

    return 0;
}
```

Fatorial

```
#include <stdio.h>

int main() {
    unsigned short int contador, n;
    double fatorial = 1;

    printf("Digite numero: ");
    scanf("%hu", &n);

    for (contador = 1; contador <= n; contador++) {
        fatorial = fatorial * contador;
    }

    printf("%hu! = %.0f\n", n, fatorial);

    return 0;
}
```

Máximo Divisor Comum (MDC)

- O algoritmo de Euclides para o cálculo do Máximo Divisor Comum (MDC) entre dois números inteiros positivos m e n , apresentado em 300 a.C., é um dos algoritmos mais antigos do mundo.
- O algoritmo pode ser resumido na seguinte fórmula:

$$\text{mdc}(m, n) = \begin{cases} m, & \text{se } n = 0 \\ \text{mdc}(n, m \% n), & \text{se } n > 0 \end{cases}$$

Máximo Divisor Comum (MDC)

```
#include <stdio.h>

int main() {
    int m, n, aux;

    printf("Entre com dois numeros inteiros positivos: ");
    scanf("%d %d", &m, &n);

    while (n > 0) {
        aux = n;
        n = m % n;
        m = aux;
    }

    printf("MDC = %d\n", m);

    return 0;
}
```


Números primos

- Um número é primo se ele tem exatamente dois divisores naturais distintos: o número um e ele mesmo.
- Dado um número $n \geq 2$, como verificar se ele é primo?
 - 1 Lê um número n .
 - 2 Testa se nenhum dos números entre 2 e $n - 1$ divide n .
- Lembre-se que o operador % retorna o resto da divisão inteira.
- Portanto $(a \% b)$ é zero se, e somente se, b divide a .
- Note que não é necessário testar os números entre $\lfloor n/2 \rfloor + 1$ e $n - 1$.
- De fato é possível testar menos números ainda...
 - 1 Se $n = a \times b$, sendo n , a e b números naturais e $a \leq b$, então $a \leq \sqrt{n} \leq b$.
 - 2 Logo, precisamos testar apenas os números inteiros entre 2 e $\lfloor \sqrt{n} \rfloor$.

Números primos

```
primo(n)
  aux = 2
  primo = verdadeiro
  enquanto primo e aux <= n/2 faça
    se aux for um divisor de n então
      primo = falso
    aux = aux + 1
  imprimir primo
```

Números primos

```
#include <stdio.h>

int main() {
    unsigned int i, n, primo = 1;

    printf("Digite um numero inteiro: ");
    scanf("%u", &n);

    for (i = 2; primo && (i <= n/2); i++)
        if ((n % i) == 0)
            primo = 0;

    if (primo)
        printf("Numero primo\n");
    else
        printf("Numero composto\n");

    return 0;
}
```

Números primos

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

int main() {
    unsigned int i, n, primo = 1;

    printf("Digite um numero inteiro: ");
    scanf("%u", &n);

    for (i = 2; primo && (i <= sqrt(n)); i++)
        if ((n % i) == 0)
            primo = 0;

    if (primo)
        printf("Numero primo\n");
    else
        printf("Numero composto\n");

    return 0;
}
```

Fatoração em números primos

- Dado um número inteiro positivo n , como fatorá-lo em números primos?
- Basta verificar todos os candidatos a fatores de n , ou seja, números menores ou iguais a n .
- Seja *fator* o número que está sendo testado numa certa iteração. Se *fator* for um divisor de n , então imprima *fator* e divida n por *fator*. Caso contrário, incremente *fator*.

Fatoração em números primos

```
#include <stdio.h>

int main() {
    unsigned int n, fator = 2;

    printf("Digite um numero inteiro positivo: ");
    scanf("%u", &n);
    printf("%d = 1", n);

    while (n > 1) {
        if ((n % fator) == 0) {
            printf(" x %d", fator);
            n = n / fator;
        } else
            fator++;
    }

    printf("\n");
    return 0;
}
```

Números de Fibonacci

- A série de Fibonacci é: 1, 1, 2, 3, 5, 8, 13, ...
- Ou seja, o n -ésimo termo é a soma dos dois anteriores

$$F(n) = F(n - 1) + F(n - 2)$$

tal que $F(1) = 1$ e $F(2) = 1$.

- Como escrever um programa que imprime os primeiros n números da série?

Números de Fibonacci

```
fibonacci(n)
  atual = 1
  proximo = 1
  para i de 1 até n faça
    imprimir atual
    temp = proximo
    proximo = atual + proximo
    atual = temp
```


Números de Fibonacci

```
#include <stdio.h>

int main() {
    unsigned long int n, atual, proximo, temp, i;

    printf("Digite um numero: ");
    scanf("%lu", &n);

    atual = 1;
    proximo = 1;

    for (i = 1; i <= n; i++) {
        printf("%lu\n", atual);
        temp = proximo;
        proximo = atual + proximo;
        atual = temp;
    }

    return 0;
}
```

Conversão de números binários para decimais

- Sabemos que um computador armazena todas as informações na forma binária, portanto, é útil saber como converter números decimais em binários (e vice-versa).
- Dado um número binário $b_n b_{n-1} \dots b_2 b_1 b_0$, este corresponde na forma decimal a:

$$\sum_{i=0}^n b_i \cdot 2^i$$

- Exemplos:

$$101 = 2^2 + 2^0 = 5$$

$$1001110100 = 2^9 + 2^6 + 2^5 + 2^4 + 2^2 = 512 + 64 + 32 + 16 + 4 = 628$$

Conversão de números binários para decimais

- Vamos supor que lemos um inteiro binário.
- Ou seja, ao lermos $n = 111$ assumimos que este é um número binário (e não “cento e onze”).
- Como transformar este número no correspondente valor decimal (7, neste caso)?
- Basta usar a expressão:

$$\sum_{i=0}^n b_i \cdot 2^i$$

Para isso, entretanto, devemos conseguir recuperar os dígitos binários (bits) individualmente.

- Note que:
 - ▶ $n \% 10$ recupera o último dígito de n .
 - ▶ $n = n / 10$ remove o último dígito de n .

Conversão de números binários para decimais

```
#include <stdio.h>

int main() {
    unsigned long int n, dec, pot, dig;

    printf("Digite um numero binario: ");
    scanf("%lu", &n);

    pot = 1;
    dec = 0;
    while (n != 0) {
        dig = n % 10;
        n = n / 10;
        dec = dec + (dig * pot);
        pot = pot * 2;
    }

    printf("%lu\n", dec);
    return 0;
}
```

Conversão de números decimais para binários

- Agora, dado um número em decimal, como obter o correspondente em binário?
- Qualquer número pode ser escrito como uma soma de potências de 2:

$$6 = 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

$$13 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

- O que acontece se dividirmos, sucessivamente, um número decimal por 2?

$$13/2 = 6, \text{ com resto } 1$$

$$6/2 = 3, \text{ com resto } 0$$

$$3/2 = 1, \text{ com resto } 1$$

$$1/2 = 0, \text{ com resto } 1$$

Conversão de números decimais para binários

```
#include <stdio.h>

int main() {
    unsigned long int n, bin, bit, pot;

    printf("Digite um numero decimal: ");
    scanf("%lu", &n);

    bin = 0;
    pot = 1;
    while (n > 0) {
        bit = n % 2;
        n = n / 2;
        bin = bin + (bit * pot);
        pot = pot * 10;
    }

    printf("%lu\n", bin);
    return 0;
}
```

Conversão de números para bases diferentes

- Dado um número inteiro positivo N escrito na base X , converta-o para a base Y , sendo que X e Y também são fornecidos como entrada do seu programa ($2 \leq X, Y \leq 10$).