BCC 201 - Introdução à Programação I

Procedimentos e Funções

Guillermo Cámara-Chávez
UFOP

Funções e Procedimentos I

Faça um procedimento que inverta a ordem de uma sequência de números salvos em um vetor

Funções e Procedimentos II

```c
void Inverte(int*, int);
void Le(int*, int);
void Mostra(int*, int);

int main()
{
    int A[5];
    Le(A, 5);
    Mostra(A, 5);
    Inverte(A, 5);
    Mostra(A, 5);
    return 0;
}
```

Funções e Procedimentos III

```
void Le(int* V, int tam)
{
    for (int i = 0; i < tam; i++)
    {
        cout << "Digite um numero: ";
        cin >> V[i];
    }
}
```

Funções e Procedimentos IV

```
void Mostra(int* V, int tam)
{
    for (int i = 0; i < tam; i++)
    {
        cout << V[i] << " ";
    }
    cout << endl;
}
```

Funções e Procedimentos V

```
void Inverte(int* V, int tam)
{
    int tmp;
    for (int i = 0, j = tam-1; i < j; i++, j--)
    {
        tmp = V[i];
        V[i] = V[j];
        V[j] = tmp;
    }
}
```

Funções e Procedimentos VI

Escreva um programa que leia 5 pares de valores positivos (**LePositivo**). Imprima se os elementos de cada par são números amigos (ou não). Dois números A e B são amigos se a soma dos divisores de A excluindo A é igual a B e a soma dos divisores de B excluindo B é igual a A. Para a verificar se dois números são amigos utilize a função **SaoAmigos**.

```
Exemplo:
220 e 284 sao amigos, pois
220: 1+2+4+5+10+11+20+22+44+55+110=284
284: 1+2+4+71+142=220
```

Funções e Procedimentos VII

```
int SomaDivisores(int);
bool SaoAmigos(int, int);
void LePositivo(int*, int*, int);

int main()
{
    int A[5], B[5];
    LePositivo(A, B, 5);
    for (int i = 0; i < 5;  i++)
    {
        if ( SaoAmigos(A[i], B[i]) )
            cout << A[i] << "eh amigo de " << B << endl;
        else
            cout << A[i] << "nao eh amigo de " << B << endl;
    }
}
```

Funções e Procedimentos VIII

```
int SomaDivisores(int num)
{
    int soma = 0;
    for (int i = 1; i < num; i++)
        if (num % i == 0)
            soma += i;
    return soma;
}
```

Funções e Procedimentos IX

```
bool SaoAmigos(int n1, int n2)
{
    if (SomaDivisores(n1) == SomaDividores(n2))
        return true;
    else
        return false;
}
```

Funções e Procedimentos X

```
void LePositivo(int* A, int* B, int nelem)
{
    for (int i = 0; i < nelem; i++)
    {
        cout << "Digite um par de valores: ";
        cin >> A[i] >> B[i];
    }
}
```

Funções e Procedimentos XI

Um número *a* é dito permutação de um número *b* se os dígitos de *a* formam uma permutação dos dígitos de *b*.
Exemplo: 5412434 é uma permutação de 4321445, mas não é uma permutação de 4312455.
Obs.: Considere que o dígito 0 (zero) não aparece nos números.

Funções e Procedimentos XII

```
int contaFreq(int , int );
bool EhPermutacao(int , int );

int main()
{
    int A, B;
    cout << "Digite dois numeros: ";
    cin >> A >> B;
    if (EhPermutacao(A, B))
        cout << A << "eh permutacao de " << B;
    else
        cout << A << "nao eh permutacao de " << B;
    return 0;
}
```

Funções e Procedimentos XIII

```
int contFreq(int num, int dig)
{
    int cont = 0, unidade;
    while(num > 0)
    {
        unidade = num % 10;
        num = num / 10;
        if (unidade == dig)
            cont++;
    }
    return cont;
}
```

Funções e Procedimentos XIV

```
bool EhPermutacao(int A, int B)
{
    for (int i = 1; i <=9; i++)
        if ( contFreq(A,i) != contFreq(B,i))
            return false;
    return true;
}
```

Funções e Procedimentos XV

Construa uma função encaixa que dados dois inteiros positivos *a* e *b* verifica se *b* corresponde aos últimos dígitos de *a*.

Ex.:

```
    a      b
567890    890 => encaixa
  1243   1243 => encaixa
  2457    245  => não encaixa
   457   2457 => não encaixa
```

Funções e Procedimentos XVI

```
bool encaixa(int , int);

int main()
{
    int a, b;
    cout << "Digite dois numeros: ";
    cin >> a >> b;
    if ( encaixa(a,b) )
        cout << "encaixa";
    else
        cout << "nao encaixa";

    return 0;
}
```

Funções e Procedimentos XVII

```
bool encaixa(int a, int b)
{
    while (b != 0 && a % 10 == b % 10)
    {
        a = a / 10;
        b = b / 10;
    }
    if (b == 0)
        return true;
    else
        return false;
}
```

Funções e Procedimentos XVIII

Faça uma função `arctan` que recebe o número real $x \in [0, 1]$ e devolve uma aproximação do arco tangente de x (em radianos) através da série incluindo todos os termos da série

$$\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$$

incluindo todos os termos da série até $|\frac{x^k}{k}| < 0.0001$

Funções e Procedimentos XIX

```
double m_arctan(double);

int main()
{
    double num;
    do{
        cout << "Digite um numero entre 0 e 1: ";
        cin >> num;
    }while(num < 0 || num > 1);
    cout << "A tangente de " << m_arctan(num) << "=" << num;
    return 0;

}
```

Funções e Procedimentos XX

```
double m_arctan(double x)
{
    double den, num, s = 0;
    int impar = 1, sinal = 1;
    do{
        num = pow(x, impar);
        den = impar;
        s += sinal * (num / den);
        impar += 2;
        sinal = -sinal;
    } while ( fabs(num/den) > 0.0001 );
    return s;
}
```

FIM