Lab. 6

Funções

ECT2303 - T02 - 19.1

6.1 Protótipo e definição de funções

Um protótipo de função diz ao compilador o **nome da função**, o **tipo dos dados retornados** pela função, o **número de parâmetros** que a função espera receber, os **tipos de parâmetros** e a **ordem** em que esses parâmetros são esperados. Por exemplo,

```
int func( int, int, int )
```

Esse protótipo de função declara que **func** utiliza três parâmetros do tipo **int** e retorna um resultado do tipo **int**.

O formato de uma **definição da função** é:

```
tipo_do_valor_de_retorno nome_da_funcão (lista de parâmetros)
{
  declaracões e comandos
}
```

Exemplo 6.1.1. Considere um programa que use a função **square** para calcular os quadrados dos números de 1 a 5.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int square( int ); // prótotipo da função
6
7  int main()
8  {
9    for (int i = 0; i <= 5; ++i)
10     cout << square( i ) << " ";
11
12   cout << endl;
13   return 0;
```

```
14 }
15 // Definição da função
16 int square( int x )
17 {
18   return x*x;
19 }
```

Um prótotipo de função não é obrigatóro se a **definição da função** aparece antes do primeiro uso da função no programa.

6.2 Exercícios de Fixação

1. Crie uma função que recebe os valores de a e b e calcula a seguinte expressão:

$$\sum_{i=a}^{b} \frac{2}{i}$$

2. Escreva uma função **`pot(x,y)`** para calcular **`x`** elevado à potência de **`y`**. Suponha que **`y`** seja um inteiro positivo, diferente de zero, e **`x`** seja um inteiro. A função **`pot`** deve usar **`for`** ou **`while`** para controlar o cálculo. Não use nenhuma das funções da biblioteca matemática.

3. Escreva uma função **`multiple`** que determina, para um par de números inteiros, se o segundo número é multiplo do primeiro. A função deve ter dois argumentos inteiros e retornar **`true`**, se o segundo número for multiplo do primeiro, ou **`false`**, caso contrário. Use essa função em um programa que leia uma série de números inteiros.

4. Escreva um programa que receba uma série de inteiros e passe cada um deles por vez para a função **`par`**, para determinar se um inteiro é par. A função deve utilizar um argumento inteiro e retornar **`true`** se o inteiro for par e **`false`** caso contrário.

5. Escreva uma função que mostre um quadrado de asteriscos cujo lado é especificado por um parâmetro **`lado`**. Por exemplo, se **`lado`** for igual a **`4`**, a função exibe:

```
****
****
****
****
```

6. Escreva uma função que retorna o menor entre três números do tipo ponto flutuante com precisão dupla.

7. O *máximo divisor comum* (MDC) de dois inteiros é o maior inteiro que divide exatamente cada um dos dois números. Escreva uma função **`mdc`** que retorna o máximo divisor comum dos dois inteiros.

8. Crie uma função que receba como parâmetro um valor inteiro e positivo N e retorne o valor de S obtido pela seguinte expressão:

$$S = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{N!}$$

9. Crie uma função que recebe os valores de x e do número de termos n e calcula o valor de e^x de acordo com a seguinte expressão:

$$e^x = \sum_{i=0}^{n-1} \frac{x^i}{i!}$$

   Implemente também a função `main`, de modo que o usuário possa digitar os valores de `x` e `n` e imprimir o valor da expressão.

10. Crie uma função que recebe os valores do ângulo x em radianos, e do número de termos n e calcula o valor de $sen(x)$ de acordo com a seguinte expressão:

$$sen(x) = \sum_{i=0}^{n-1} \frac{(-1)^i x^{2i+1}}{(2i+1)!}$$

    Implemente também a função `main`, de modo que o usuário possa digitar os valores de `x` e `n` e imprimir o valor da expressão.

11. A sequência de Fibonacci é dada por

$$1, 1, 2, 3, 5, 8, 13, 21, ...,$$

    isto é, ela é uma sequência infinita de números inteiros e positivos onde cada termo é formado pela soma dos dois anteriores, sendo os dois primeiros termos iguais a 1. Implemente uma função que recebe um número positivo n e calcula o n-ésimo número de Fibonacci. Também, crie um programa `main` que recebe o valor de n e realiza a impressão os n primeiros números da sequência de Fibonacci, como mostrado a seguir:

```
-- Exemplo 1:
Informe um numero
2
1 1
-- Exemplo 2:
Informe um numero
7
1 1 2 3 5 8 13
```

12. Escreva uma função que receba uma valor inteiro e retorne o número com seus dígitos invertidos. Por exemplo, dado o número 7631, a função deve retornar 1367.

13. Diz-se que um número inteiro é um *número perfeito* se a soma de seus fatores, incluindo 1 (mas não o própio número), é igual ao próprio número. Por exemplo, 6 é um número perfeito porque $6 = 1 + 2 + 3$. Escreva uma função **`perfect`** que determine se o parâmetro **`number`** é um número perfeito. Use essa função em um programa que determina e imprima todos os números perfeitos entre 1 e 1000.

14. Diz-se que um número inteiro é um *número primo* se for divisível apenas por 1 e por si mesmo. Por exemplo, 2, 3, 5 e 7 são primos, mas 4, 6, 8 e 9 não o são.

    (a) Escreva uma função que determina se um número é primo.

    (b) Use essa função em um programa que determine e imprima todos os números primos entre 1 e 10000.

    (c) Inicialmente você poderia pensar que $n/2$ é o limite superior que deve ser testado para verificar se um número é primo, mas você só precisa ir até a raiz quadrada de n. Por quê? Reescreva o programa e execute-o das duas maneiras. Faça uma estimativa da melhora de desempenho.

6.3 Referências Bibliográficas

1. MANZANO, J.A.; OLIVEIRA, J.F.; **Algoritmos - Lógica para Desenvolvimento de Programação**. Editora Érica.

2. ASCENCIO, A.F.G.; CAMPOS, E.A.V. **Fundamentos da Programação de Computadores - Algoritmos, Pascal e C/C++**. 3ed. Editora Pearson.

3. DEITEL, H. M.; DEITEL, P. J. **C++ Como Programar**. 3ed. Editora Bookman.