



ALGORITMOS III

Prof. Ms. Ronan Loschi.

ronan.loschi@unifasar.edu.br

31-98759-9555

HORÁRIO PROVISÓRIO:

Turma Soft.3

	Segunda	Terça	Quarta	Quinta	Sexta
01º	Sistemas Operacionais Luiz Otávio	Sistemas Operacionais Luiz Otávio	Algoritmos III Ronan Loschi	Engenharia de Req.Softw. Ronan Loschi	Redes de Computadores José Carlos
02º	Sistemas Operacionais Luiz Otávio	Sistemas Operacionais Luiz Otávio	Algoritmos III Ronan Loschi	Engenharia de Req.Softw. Ronan Loschi	Redes de Computadores José Carlos
03º	Algoritmos III Ronan Loschi	Engenharia de Req.Softw. Ronan Loschi	Redes de Computadores José Carlos	Prog.Integ.:Apl.p/Cen.Rea Denilson Gomes	Empr.Gest.Comput.II José Carlos
04º	Algoritmos III Ronan Loschi	Engenharia de Req.Softw. Ronan Loschi	Redes de Computadores José Carlos	Prog.Integ.:Apl.p/Cen.Rea Denilson Gomes	Empr.Gest.Comput.II José Carlos

1º PERÍODO
DISCIPLINAS
ALGORITMOS E ESTRUTURAS DE DADOS I
CÁLCULO I
DESENVOLVIMENTO DE INTERFACES WEB
FUNDAMENTOS DE ENGENHARIA DE SOFTWARE
INTRODUÇÃO A COMPUTAÇÃO
PROGRAMA INTEGRADOR: APLICAÇÕES WEB
2º PERÍODO
DISCIPLINAS
ARQUITETURA DE COMPUTADORES
ALGORITMOS E ESTRUTURAS DE DADOS II
CÁLCULO II
EMPREENDEDORISMO E GESTÃO EM COMPUTAÇÃO I
LABORATÓRIO DE PROGRAMAÇÃO MODULAR
MODELAGEM DE PROCESSOS DE NEGÓCIOS
PROGRAMAÇÃO MODULAR
PROGRAMA INTEGRADOR: APLICAÇÕES PARA PROCESSOS DE NEGÓCIOS
3º PERÍODO
DISCIPLINAS
ALGORITMOS E ESTRUTURAS DE DADOS III
REDES DE COMPUTADORES
ENGENHARIA DE REQUISITOS DE SOFTWARE
EMPREENDEDORISMO E GESTÃO EM COMPUTAÇÃO II
SISTEMAS OPERACIONAIS
PROGRAMA INTEGRADOR: APLICAÇÕES PARA CENÁRIOS REAIS

ENGENHARIA DE REQUISITOS DE SOFTWARE

EMENTA:

- Protótipo de Funções
- Ponteiros, alocação dinâmica, arquivos.
- Fundamentos de análise de algoritmos.
- Ordenação e pesquisa em memória principal.
- Tipos abstratos de dados lineares e flexíveis (Pilha, Fila, Árvore binária).
- Árvores. Balanceamento de árvores.
- Tabelas e Dicionário

Objetivo Geral

Desenvolver a capacidade dos alunos de projetar, implementar e analisar **algoritmos avançados** em C++, aprimorando a lógica de programação e a eficiência computacional para a resolução de problemas complexos.

Objetivos Específicos:

- **Aprofundar o conhecimento sobre estruturas de dados dinâmicas** como listas encadeadas, pilhas, filas e árvores, compreendendo suas aplicações e complexidade computacional.
- **Explorar técnicas de ordenação e busca** eficientes, analisando sua performance e aplicabilidade em diferentes cenários computacionais.
- **Introduzir conceitos de programação modular e recursiva**, promovendo a organização e reutilização do código por meio do uso de funções, bibliotecas e classes em C++.

PROTÓTIPO DE FUNÇÕES:

O que é um Protótipo de Função?

Um **protótipo de função** é uma **declaração** da função antes de sua implementação. Ele informa ao compilador:

- Nome da função
- Tipo de retorno
- Parâmetros esperados

SINTAXE:

```
tipo_de_retorno nome_da_funcao(tipo1 param1, tipo2 param2, ...);
```

✓ **Vantagem:** Permite definir funções **depois** da **main()**, tornando o código mais organizado.

EXEMPLO:

```
#include <iostream>
using namespace std;

// Protótipo da função
int soma(int a, int b); ←

int main() {
    cout << "Soma: " << soma(3, 4) << endl;
    return 0;
}

// Definição da função ←
int soma(int a, int b) {
    return a + b;
}
```

Protótipos com Diferentes Tipos de Retorno:

Podemos usar **qualquer tipo de retorno**, como double, char, void, etc.

```
#include <iostream>
using namespace std;

double dividir(double x, double y); // Protótipo
↑
int main() {
    cout << "Resultado: " << dividir(10, 2) << endl;
    return 0;
}

double dividir(double x, double y) {
    return x / y;
}
```


Protótipos com Parâmetro Padrão:

```
#include <iostream>
using namespace std;

// Protótipo com valor padrão
void saudacao(string nome = "Visitante");

int main() {
    saudacao(); // Usa "Visitante"
    saudacao("Maria"); // Usa "Maria"
    return 0;
}

void saudacao(string nome) {
    cout << "Olá, " << nome << "!" << endl;
}
```

Protótipos com Parâmetro do Tipo VOID:

Funções void não retornam valores, mas ainda podem ter **protótipos**.

```
#include <iostream>
using namespace std;

// Protótipo
void exibirMensagem();

int main() {
    exibirMensagem();
    return 0;
}

void exibirMensagem() {
    cout << "Bem-vindo ao curso de C++!" <<
}
```

Protótipos com Ponteiros:

Também podemos usar ponteiros como parâmetros em protótipos.

```
#include <iostream>
using namespace std;

// Protótipo
void dobrarValor(int *num);

int main() {
    int x = 5;
    dobrarValor(&x);
    cout << "Valor dobrado: " << x << endl;
    return 0;
}

void dobrarValor(int *num) {
    *num *= 2;
}
```

NÚMEROS ALEATÓRIOS EM C++:

O comando `rand()`, gera um número aleatório (randômico), inteiro e positivo obtido diretamente do computador.

SINTAXE: `int nome_variavel = rand () % max;`

SINTAXE: `int nome_variavel = rand () % complemento + min;`

Para usar o comando `random`, deve-se adicionar a biblioteca **`stdlib.h`**

Em C++, **random** geralmente se refere à biblioteca `<random>`, introduzida no C++11, que fornece uma maneira mais robusta e moderna de gerar números aleatórios em comparação com **rand()** e **srand()**.

Principais diferenças entre `rand()` e `<random>`

Característica	<code>rand()</code> e <code>srand()</code>	<code><random></code> (C++11)
Método de geração	Baseado em um gerador linear congruente (LCG), menos seguro e previsível.	Usa geradores modernos como Mersenne Twister, mais aleatórios e confiáveis.
Faixa de valores	<code>0</code> até <code>RAND_MAX</code> (depende do sistema, geralmente <code>32767</code>).	Permite definir intervalos personalizados.
Controle da aleatoriedade	Menos preciso e repetitivo sem <code>srand(time(0))</code> .	Melhor controle sobre distribuição e geração.
Segurança	Não adequado para criptografia.	Possui geradores adequados para mais segurança.

EXEMPLO usando rand () e srand (antigo menos recomendado):

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese");
    srand(time(0)); // Inicializa com o tempo atual
    cout << "Número aleatório com rand(): " << rand() % 300 << endl; // Número entre 0 e 99
    return 0;
}
```



EXEMPLO usando rand () e srand (antigo menos recomendado):

```
#include <iostream>
#include <cstdlib> // Para srand() e rand()
#include <ctime> // Para time()

using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese");

    srand(time(0)); // Define a semente baseada no tempo atual

    int numero1 = rand() % 100; // Número aleatório entre 0 e 99
    int numero2 = rand() % 50 + 1; // Número entre 1 e 50
    int numero3 = rand(); // Número aleatório

    cout << "Número aleatório entre 0 e 99: " << numero1 << endl;
    cout << "Número aleatório entre 1 e 50: " << numero2 << endl;
    cout << "Número aleatório: " << numero3 << endl;
    cout << "Intervalo: 0 - " << RAND_MAX << endl;

    return 0;
}
```


EXEMPLO usando random () (moderno, recomendado):

```
#include <iostream>
#include <random> // Necessário para usar geradores modernos

using namespace std;

int main() {
    random_device rd; // Dispositivo de número aleatório
    mt19937 gen(rd()); // Mersenne Twister PRNG
    uniform_int_distribution<int> dist(0, 100); // Define intervalo de 0 a 100

    int a = dist(gen); // Gera um número no intervalo definido
    cout << "Número aleatório: " << a << endl;
    return 0;
}
```



EXEMPLO usando random () (moderno, recomendado):

```
#include <iostream>
#include <random>
#include <ctime> // Necessário para time()


using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese");

    mt19937 gen(time(0)); // Semente baseada no tempo
    uniform_int_distribution<int> dist(0, 500); // Intervalo de 0 a 500

    cout << "Número aleatório com <random>: " << dist(gen) << endl;

    return 0;
}
```



EXEMPLO:

✓ Passo a passo para ativar o C++11 no Dev-C++ 5.11

1. Abra o Dev-C++
2. Vá em **Tools** (Ferramentas) → **Compiler Options** (Opções do Compilador)
3. Na aba **Settings**, selecione **C++ Compiler**
4. Encontre a opção "**Add the following commands when calling the compiler**"
5. **Adicione o seguinte código** no campo de texto:

```
ini
```

```
-std=c++11
```

Se quiser ativar o C++ mais recente disponível (se seu MinGW suportar):

```
ini
```

```
-std=c++14
```

OU

```
ini
```

```
-std=c++17
```


EXERCÍCIOS:

- 1) Explique a importância dos protótipos de funções.
- 2) O que acontece se chamarmos uma função sem um protótipo declarado antes da main()?
- 3) Escreva um protótipo para uma função que multiplica a por b.
- 4) Implemente um programa que use um protótipo para calcular a área de um círculo (float areaCirculo(float raio)).
- 5) Modifique o programa anterior para usar um parâmetro com valor padrão.
- 6) Construa um programa em C++ que leia a idade, a altura e o nome de 10 pessoas (Utilizar Vetores). Utilizando protótipos e funções, calcular e imprimir o nome e a idade da pessoa mais velha; bem como o nome, idade e a altura da pessoa mais nova.
- 7) Construa um programa em C++ que gere números aleatórios entre 0 e 200; entre 250 e 500 e até o RAND_MAX.

GABARITO:

- 1) Os protótipos de funções permitem que o compilador conheça a assinatura da função antes de sua implementação, evitando erros e possibilitando a organização do código.
- 2) O compilador apresentará um erro, pois não saberá que a função existe antes de sua chamada na `main()`.

GABARITO:

3)

```
int multiplica(int a, int b);
```

4)

```
#include <iostream>
using namespace std;

float areaCirculo(float raio); // Protótipo

int main() {
    float r;
    cout << "Digite o raio: ";
    cin >> r;
    cout << "Área: " << areaCirculo(r) << endl;
    return 0;
}

float areaCirculo(float raio) {
    return 3.14159 * raio * raio;
}
```

5)

```
#include <iostream>
using namespace std;

float areaCirculo(float raio = 1.0); // Protótipo com valor padrão

int main() {
    cout << "Área com raio padrão: " << areaCirculo() << endl;
    cout << "Área com raio 5: " << areaCirculo(5) << endl;
    return 0;
}

float areaCirculo(float raio) {
    return 3.14159 * raio * raio;
}
```

6)

```
#include <iostream>  
#include <string>  
#include <locale.h>
```

```
using namespace std;
```

```
// Protótipos das funções
```

```
void encontrarMaisVelho (string nomes1[ ], int idades1[ ], int tamanho1, string  
&nomeMaisVelho1, int &idadeMaisVelho1);
```

```
void encontrarMaisNovo (string nomes2[ ], int idades2[ ], float alturas2[ ], int  
tamanho2, string &nomeMaisNovo2, int &idadeMaisNovo2, float &alturaMaisNovo2);
```

```
6) int main( ) {  
    setlocale(LC_ALL, "Portuguese");  
  
    const int NUM_PESSOAS = 10;  
    string    nomes[NUM_PESSOAS];  
    int       idades[NUM_PESSOAS];  
    float     alturas[NUM_PESSOAS];  
  
    // Variáveis para armazenar os resultados  
    string nomeMaisVelho, nomeMaisNovo;  
  
    int idadeMaisVelho, idadeMaisNovo;  
  
    float alturaMaisNovo;
```

6)

// Coleta de dados

```
for (int i = 0; i < NUM_PESSOAS; i++) {  
    cout << "Digite o nome da pessoa " << i + 1 << ": ";  
    cin >> nomes[ i ];  
  
    cout << "Digite a idade de " << nomes[i] << ": ";  
    cin >> idades[ i ];  
  
    cout << "Digite a altura de " << nomes[i] << " (em metros): ";  
    cin >> alturas[ i ];  
}
```


6)

// Chamando as funções para encontrar os resultados

encontrarMaisVelho(nomes, idades, NUM_PESSOAS, nomeMaisVelho, idadeMaisVelho);

encontrarMaisNovo (nomes, idades, alturas, NUM_PESSOAS, nomeMaisNovo, idadeMaisNovo, alturaMaisNovo);

// Exibindo os resultados

cout << "\n A pessoa mais velha é " << nomeMaisVelho << " com " << idadeMaisVelho << " anos.\n";

cout << "A pessoa mais nova é " << nomeMaisNovo << " com " << idadeMaisNovo << " anos e altura de " << alturaMaisNovo << " metros.\n";

return 0;

}

6) **// Função para encontrar a pessoa mais velha**
void encontrarMaisVelho (string nomes1[], int idades1[], int tamanho1, string &nomeMaisVelho1, int &idadeMaisVelho1) {

 idadeMaisVelho1 = **-999999**;

for (int i = 1; i < tamanho1; i++) {
 if (idades1[i] > idadeMaisVelho1) {
 idadeMaisVelho1 = idades1[i];
 nomeMaisVelho1 = nomes1[i];
 }
 }

}

6)

// Função para encontrar a pessoa mais nova e sua altura

```
void encontrarMaisNovo(string nomes2[], int idades2[], float alturas2[],  
int tamanho2, string &nomeMaisNovo2, int &idadeMaisNovo2, float  
&alturaMaisNovo2) {
```

```
    idadeMaisNovo2 = +99999;
```

```
    for (int i = 0; i < tamanho2; i++) {  
        if (idades2[i] < idadeMaisNovo2) {  
            idadeMaisNovo2 = idades2[i];  
            nomeMaisNovo2 = nomes2[i];  
            alturaMaisNovo2 = alturas2[i];  
        }  
    }
```

```
}
```

7)

```
#include <iostream>
#include <cstdlib> // Para srand() e rand()
#include <ctime>    // Para time()

using namespace std;

int main() {
    setlocale(LC_ALL, "Portuguese");

    srand(time(0)); // Define a semente baseada no tempo atual

    int numero1 = rand() % 201; // Número aleatório entre 0 e 200
    int numero2 = rand() % 251 + 250; // Número entre 250 e 500
    int numero3 = rand(); // Número aleatório

    cout << "Número aleatório entre 0 e 200: " << numero1 << endl;
    cout << "Número aleatório entre 250 e 500: " << numero2 << endl;
    cout << "Número aleatório: " << numero3 << endl;
    cout << "Intervalo: 0 - " << RAND_MAX << endl;

    return 0;
}
```

OBRIGADO!