Estácio

ALGORITMOS E COMPLEXIDADE (ARA0174)

AULA 1

DATA: 17/02/2022

DISCIPLINA: ALGORITMOS E COMPLEXIDADE

Contextualização:

O estudo da disciplina Algoritmos e Complexidade, componente fundamental no aprendizado da Área da Computação, é a base sobre a qual muitos outros campos dessa área são construídos. O conhecimento dos Algoritmos e Complexidade é importante para estudantes que desejem trabalhar em implementações, testes e manutenção de projetos de qualquer sistema de software.

Estácio

DISCIPLINA: ALGORITMOS E COMPLEXIDADE

CONTEÚDO:

TEMA 1 – Relembrando conceitos básicos de Algoritmos e Estrutura de Dados com variados EXERCÍCIOS.

Análise de Algoritmo - Notação O.

TEMA 2 – Recursividade.

TEMA 3 - Algoritmo de Ordenação Avançados (Ordenação por intercalação – MERGESORT e Ordenação rápida - QUICKSORT).

TEMA 4 - Estruturas de dados dos tipos Árvore Binária e Árvore AVL.

TEMA 5 - Algoritmos em Grafos

(CRÉDITO DIGITAL).

Estácio

DISCIPLINA: ALGORITMOS E COMPLEXIDADE

AVALIAÇÃO

AVALIAÇÃO OBRIGATÓRIA!

AVD:
Avaliação digital dos(s) tema(s) vínculados ao crédito digital no valor total de 10 (dez) pontos.

Algoritmos

- Algoritmos são um conjunto finito de passos elementares que são aplicados sistematicamente até que a solução seja atingida. De forma simples, podemos dizer que **um algoritmo define o caminho que deve ser seguido para chegar até a solução de um determinado problema**.

Estrutura de Dados

- Trabalha com estruturas mais complexas em relação aos tipos de dados, para organizar os dados de acordo com um determinado problema.

- Benefícios?
  - Organização da informação;
  - Melhora o desempenho;
  - Proporciona o reuso de código;
  - Proporciona interoperabilidade;
  - Diminui custos.

Tipos de Dados

- Define o conjuntos de valores (domínio) que uma variável pode assumir.
- O dado pode ser um tipo padrão do próprio compilador.
- No caso da linguagem C:

Existem 5 tipos de dados primitivos (pré-definidos) em C.
São eles:

Palavra Chave	Tipo
char	caracter
int	inteiro
float	real de precisão simples
double	real de precisao dupla
void	vazio (sem valor)

- Exemplo:

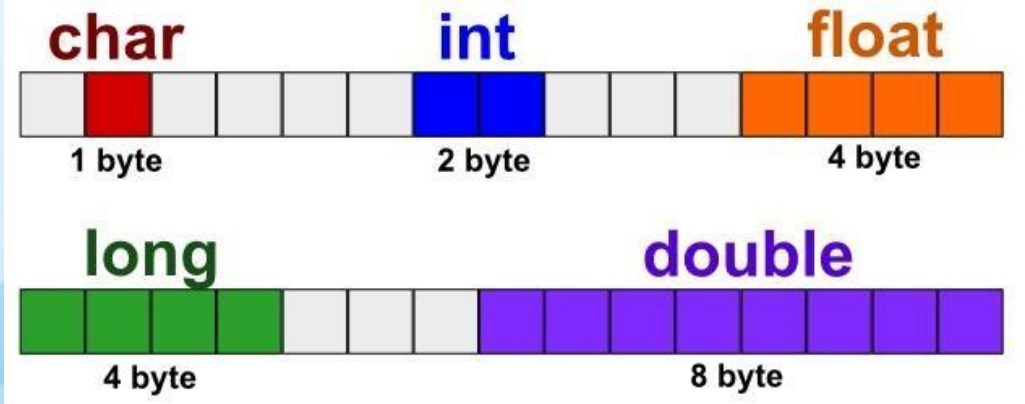
$$int \rightarrow \cdots -2, -1, 0, 1, 2, 3 \ldots$$

Tipos de Dados

- Para cada dado armazenado no computador, devemos definir o conjunto de valores que ele pode assumir. Denominamos a esse conjunto de tipo de dados.

- Por exemplo, um dado (variável) do **tipo lógico** pode assumir o valor **verdadeiro ou falso** .

Tipo Básico de Dados



O tamanho e a faixa dos dados dependem do processador e da implementação do compilador.

Estrutura de Dados

- Relacionamento lógico entre os tipos de dados.

- A implementação de um TAD (Tipo Abstrato de Dados) escolhe uma estrutura de dados (ED) para representá-lo. Cada ED pode ser construída a partir de tipos básicos (inteiro, real, caracter) ou estruturada (array, registro) de uma determinada linguagem de programação.

Estrutura de Dados

- As estruturas de dados de tipos de dados estruturadas se divide em:

  * homogêneos (vetores e matrizes);
  * heterogêneos (registros).

- As estruturas homogêneas são conjuntos de dados formados pelo **mesmo tipo** de dado primitivo.

- As estruturas heterogêneas são conjuntos de dados formados por tipos de dados primitivos **diferentes** (campos do registro) em uma mesma estrutura.

Estruturas e Dados

Estruturas

- Vetores
- Matrizes
- Registros
- Listas
- Filas
- Pilhas
- Árvores

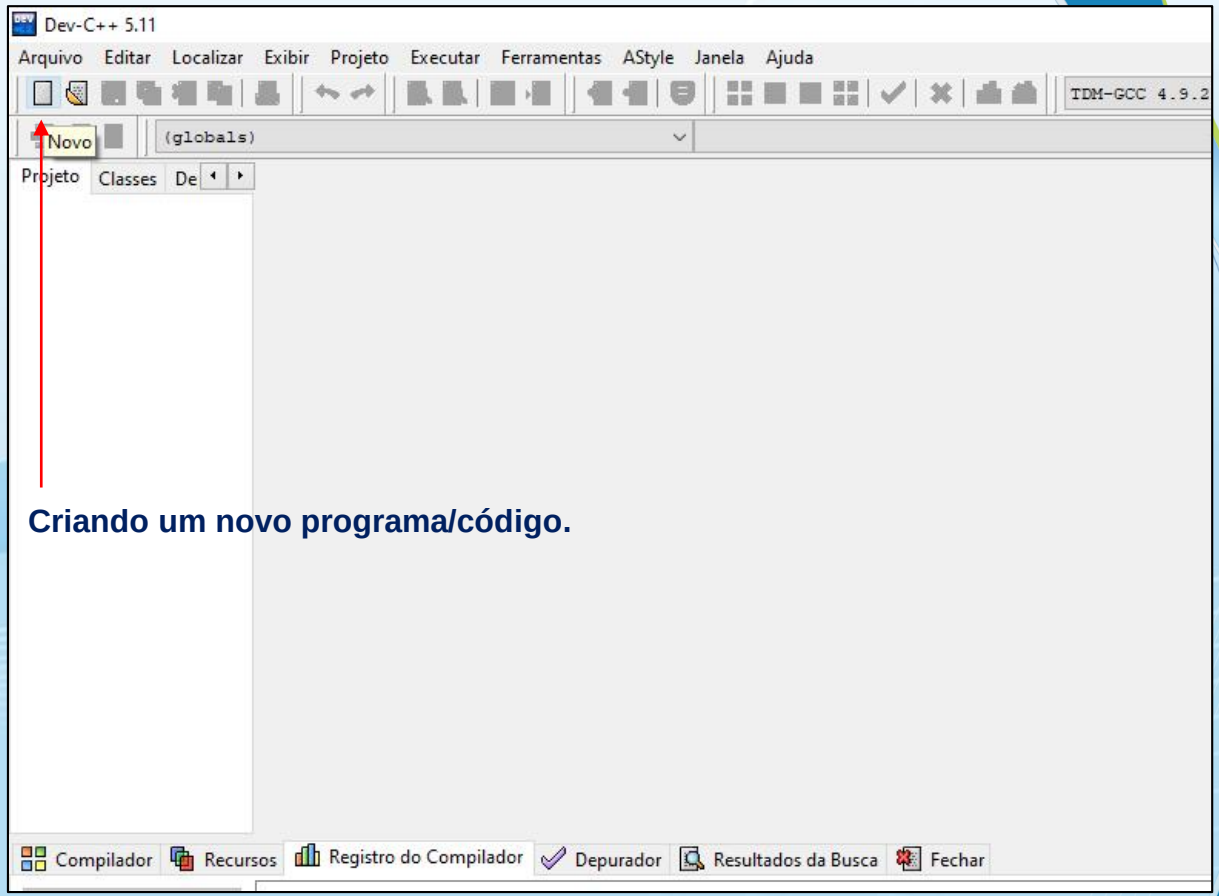
Dados

- int (Inteiro)
- float (Ponto Flutuante)
- double (Ponto Flutuante)
- char (Caracter)
- string (Texto)

Relembrando tipo de dados (EXEMPLO)

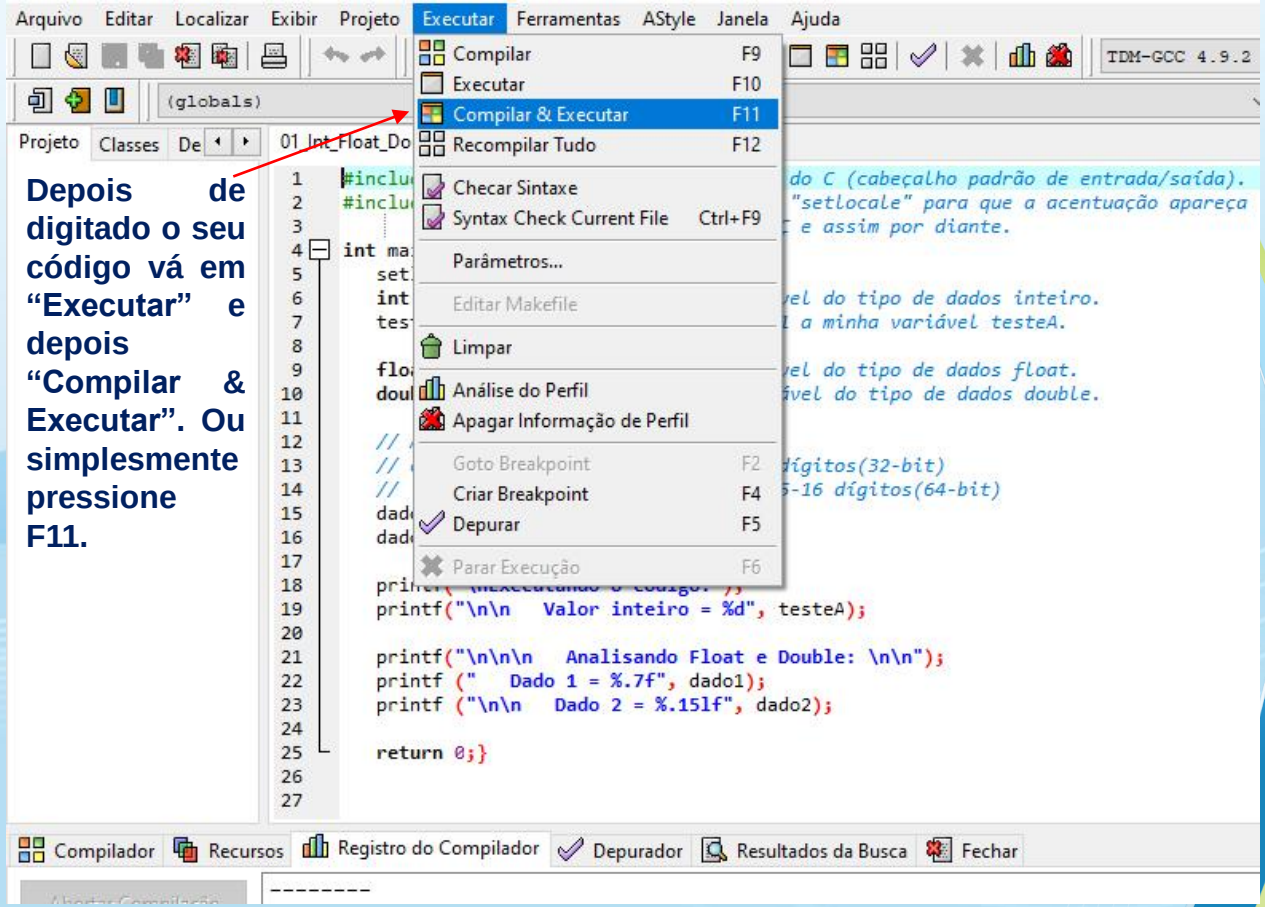
Mas antes relembrando como se utiliza o ambiente de desenvolvimento DEV-C++

Ambiente de desenvolvimento DEV-C++



Criando um novo programa/código.

Ambiente de desenvolvimento DEV-C++



Depois de digitado o seu código vá em "Executar" e depois "Compilar & Executar". Ou simplesmente pressione F11.

Relembrando tipo de dados

EXEMPLO 1

```c
#include <stdio.h> // Biblioteca padrão do C (cabeçalho padrão de entrada/saída).
#include <locale.h> // Na utilização do "setlocale" para que a acentuação apareça
                    // corretamente, o Ç e assim por diante.
int main(){
   setlocale(LC_ALL, "portuguese");
   int testeA;  // Declarando uma variável do tipo de dados inteiro.
   testeA = 21; // Atribuindo o valor 21 a minha variável testeA.

   float dado1; // Declarando uma variável do tipo de dados float.
   double dado2; // Declarando uma variável do tipo de dados double.

   // A diferença entre Float e Double
   // é basicamente precisão: Float: 7 dígitos(32-bit)
   //                         Double: 15-16 dígitos(64-bit)
   dado1 = 3.14159265358979323;
   dado2 = 3.14159265358979323;

   printf("\nExecutando o código:");
   printf("\n\n   Valor inteiro = %d", testeA);

   printf("\n\n\n   Analisando Float e Double: \n\n");
   printf ("   Dado 1 = %.7f", dado1);
   printf ("\n\n   Dado 2 = %.15lf", dado2);

   return 0;}
```

Relembrando tipo de dados

COMPILANDO E EXECUTANDO O EXEMPLO 1 VAI APARECER A TELA ABAIXO:

```
Executando o código:

   Valor inteiro = 21


   Analisando Float e Double:

   Dado 1 = 3,1415927

   Dado 2 = 3,141592653589793
--------------------------------
Process exited after 0.04517 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

```c
#include <s                                          ada/saída).
#include <l                                          ção apareça

int main(){
    setlocal
    int test
    testeA =

    float da
    double d

    // A dif
    // é bas
    //
    dado1 = 3.14159265358979323;
    dado2 = 3.14159265358979323;

    printf("\nExecutando o código:");
    printf("\n\n   Valor inteiro = %d", testeA);

    printf("\n\n\n   Analisando Float e Double: \n\n");
    printf ("   Dado 1 = %.7f", dado1);
    printf ("\n\n   Dado 2 = %.15lf", dado2);

    return 0;}
```

Relembrando tipo de dados

EXEMPLO 2

```c
#include <stdio.h> // Biblioteca padrão do C (cabeçalho padrão de entrada/saída).
#include <string.h> // Fornece funções, macros e definições da biblioteca padrão
                    // da linguagem de programação C para manipulação de cadeias de
                    //  caracteres e regiões de memória. Para o "strlen" funcionar
                    // precisa dessa biblioteca.
#include <locale.h> // Na utilização do "setlocale" para que a acentuação apareça
                    // corretamente, o Ç e assim por diante.
 int main() {
    setlocale(LC_ALL, "portuguese");
    char nome[5];

    printf("Digite o seu nome: ");
    fgets(nome, 5, stdin);

    // strcspn = Retorna um caractere em um string.
    nome[strcspn(nome, "\n")] = '\0';

    printf("O nome é: %s ", nome);

    for (int i=0;i<5;i++) {

      if (nome[i] != '\0') {
         printf("\n\n Entendendo: Posição: %d tem a letra %c", i, nome[i]);
     }

      if (nome[i] == '\0') {
         printf("\n\n Entendendo: Posição: %d é igual a 0 (ZERO).", i);
         break;
     }   }
    return 0;
}
```

Relembrando tipo de dados

COMPILANDO E EXECUTANDO O EXEMPLO 2 VAI APARECER A TELA ABAIXO:

```
Digite o seu nome: leo
O nome é: leo

 Entendendo: Posição: 0 tem a letra l

 Entendendo: Posição: 1 tem a letra e

 Entendendo: Posição: 2 tem a letra o

 Entendendo: Posição: 3 é igual a 0 (ZERO).
--------------------------------
Process exited after 1.696 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

```
#include
#include
#include
int mai
    setl
    char
    prin
    fget
    // s
    nome
    printf("O nome é: %s ", nome);
    for (int i=0;i<5;i++) {
      if (nome[i] != '\0') {
         printf("\n\n Entendendo: Posição: %d tem a letra %c", i, nome[i]);
      }
      if (nome[i] == '\0') {
         printf("\n\n Entendendo: Posição: %d é igual a 0 (ZERO).", i);
         break;
      }   }
    return 0;
}
```

Tipo de Dados Abstratos(TDA)

"Um tipo de dado abstrato pode ser definido como um conjunto de valores e uma coleção de operações que atuam sobre esses valores.

As operações devem ser consistentes com os tipos de valores".

(MORAES, C.R., 2001, p.5)

METODOLOGIA DE ESTUDO

- Lembre-se de que Algoritmos e Estruturas de Dados formam uma parceria perfeita, contribuindo para seu aperfeiçoamento como desenvolvedor.

REVISÃO - MATRIZES

MATRIZES

$$A_{1x4} \begin{pmatrix} 2 & 3 & 1 & 13 \end{pmatrix}$$

$$A_{4x1} \begin{pmatrix} 2 \\ 3 \\ 1 \\ 13 \end{pmatrix}$$

$$A_{mxn} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{3n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

MATRIZES

$$A_{1x4} \begin{pmatrix} 2 & 3 & 1 & 13 \end{pmatrix}$$

unidimensional

$$A_{4x1} \begin{pmatrix} 2 \\ 3 \\ 1 \\ 13 \end{pmatrix}$$

bidimensional

$$A_{mxn} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{3n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

Operação - Multiplicar por um escalar

$$A_{4x1} \begin{pmatrix} 2 \\ 3 \\ 1 \\ 13 \end{pmatrix} \quad X\ 2 = \quad D_{4x1} \begin{pmatrix} 4 \\ 6 \\ 2 \\ 26 \end{pmatrix}$$

Operação – Somar duas matrizes

$$A \begin{pmatrix} 2 \\ 3 \\ 1 \\ 13 \end{pmatrix} + B \begin{pmatrix} 4 \\ 6 \\ 2 \\ 26 \end{pmatrix} = S \begin{pmatrix} 6 \\ 9 \\ 3 \\ 39 \end{pmatrix}$$

Operação: calcular média aritmética

$$\left(A \begin{pmatrix} 4 \\ 5 \\ 6 \\ 10 \end{pmatrix} + B \begin{pmatrix} 8 \\ 7 \\ 9 \\ 4 \end{pmatrix} \right) / 2 = M \begin{pmatrix} 6 \\ 6 \\ 7.5 \\ 7 \end{pmatrix}$$

Operação: somar os elementos da matriz

$$A \begin{pmatrix} 4 \\ 5 \\ 6 \\ 10 \end{pmatrix}$$

Soma = 25

Dimensionando Matrizes

```
tipo nomeMatriz[tamanho] ;
```

Dimensionando Matrizes

```
tipo nomeMatriz[tamanho] ;
```

```
tipo nomeMatriz[tamLinha][tamColuna] ;
```

Dimensionando Matrizes

```
tipo nomeMatriz[tamanho] ;
```

```
tipo nomeMatriz[tamLinha][tamColuna] ;
```

int
float
double
char
long long int
...

TIPOS AGREGADOS

Os tipos agregados classificam-se como **homogêneos** e **heterogêneos**.

Homogêneos

Conjunto de elementos de **mesmo tipo** que podem ser acessados por um subscrito (índice). Podem possuir uma dimensão (vetores) ou mais de uma (matrizes).

Declaração:

< tipo_variável > < nome_variável > [< no.elementos >]

Exemplo: float notas [10];
int codigo [10] [2];

TIPOS AGREGADOS

Homogêneos

< tipo_variável > < nome_variável > [< no.elementos >]

Exemplo: float notas [10]; // Vai de 0 a 9.
int codigo [10] [2]; // Linha: Vai de 0 a 9.
// Coluna: Vai de 0 a 1.

- primeiro elemento do vetor é sempre o de índice zero.

Caso o vetor possua 10 elementos, seus índices irão de 0 a 9, cabendo ao programador garantir que o número de elementos não seja ultrapassado.

TIPOS AGREGADOS

Homogêneos

< tipo_variável > < nome_variável > [< no.elementos >]

Exemplo: float notas [10]; // Vai de 0 a 9.
int codigo [10] [2]; // Linha: Vai de 0 a 9.
// Coluna: Vai de 0 a 1.

• Uma cadeia de caracteres (string) é declarada utilizando-se o mesmo mecanismo de declaração de qualquer vetor . De acordo com a definição da linguagem, as cadeias de caracteres terminam sempre por um caracter nulo, representado por **"\0"**. Por isto, o número de elementos da string deve incluir um espaço a mais para armazenar este caracter nulo.

1) Guardar 10 idades.

1) Guardar 10 idades.

```
int idades[10];
```

1) Guardar 10 idades.

```
int idades[10];
```

2) Duas notas e a média de 50 alunos.

1) Guardar 10 idades.

```
int idades[10];
```

2) Duas notas e a média de 50 alunos.

```
float nt1[50], nt2[50], m[50];
```

3) Matrícula e CR de 60 alunos.

3) Matrícula e CR de 60 alunos.

```
float CR[60]; int matric[60];
```

4) Nome de uma pessoa.

4) Nome de uma pessoa.

```
char nome[30];
```

4) Nome de uma pessoa.

```
char nome[30];
```

5) Nomes de 10 pessoas.

4) Nome de uma pessoa.

```
char nome[30];
```

5) Nomes de 10 pessoas.

```
char nomes[10][30];
```

DECLARAÇÃO / ATRIBUIÇÃO

Inicialização sem especificação de tamanho

Podemos, em alguns casos, inicializar matrizes das quais não sabemos o tamanho a priori.

O compilador C vai, neste caso verificar o tamanho do que você declarou e considerar como sendo o tamanho da matriz.

Isto ocorre na hora da compilação e não poderá mais ser mudado durante o programa, sendo muito útil, por exemplo, quando vamos inicializar uma string e não queremos contar quantos caracteres serão necessários. Alguns exemplos:

```
char x [ ] = "Linguagem C: flexibilidade e poder.";

int y [ ][2] = { 1, 2 , 2 , 4 , 3 , 6, 4 , 8 , 5 , 10 };
```

DECLARAÇÃO / ATRIBUIÇÃO

Inicialização sem especificação de tamanho

```
char x [ ] = "Linguagem C: flexibilidade e poder.";
```

TOTAL DE 36 CARACTERES !.

```
int y [ ][2] = { 1, 2 , 2 , 4 , 3 , 6, 4 , 8 , 5 , 10 };
```

TOTAL DE 5 LINHAS E 2 COLUNAS !.

Armazenamento na Memória Principal

Vetor de tamanho 5 e do tipo inteiro de quatro bytes

Armazenamento na Memória Principal

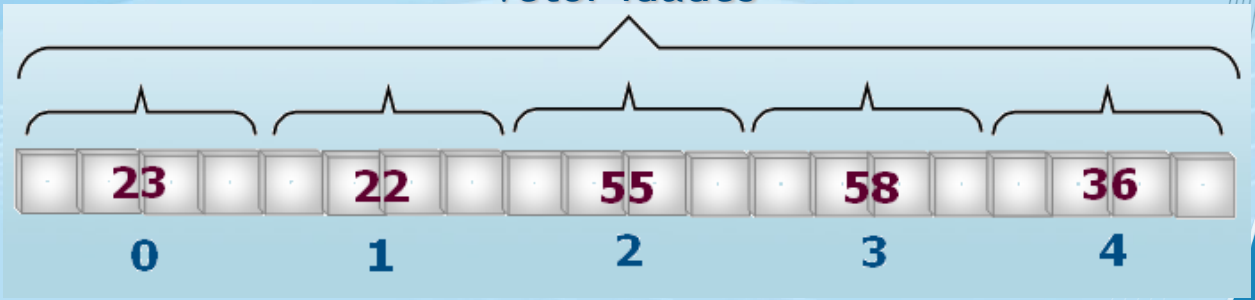
Vetor de tamanho 5 e do tipo inteiro de quatro bytes

Vetor idades

Armazenamento na Memória Principal

Vetor de tamanho 5 e do tipo inteiro de quatro bytes

Vetor idades

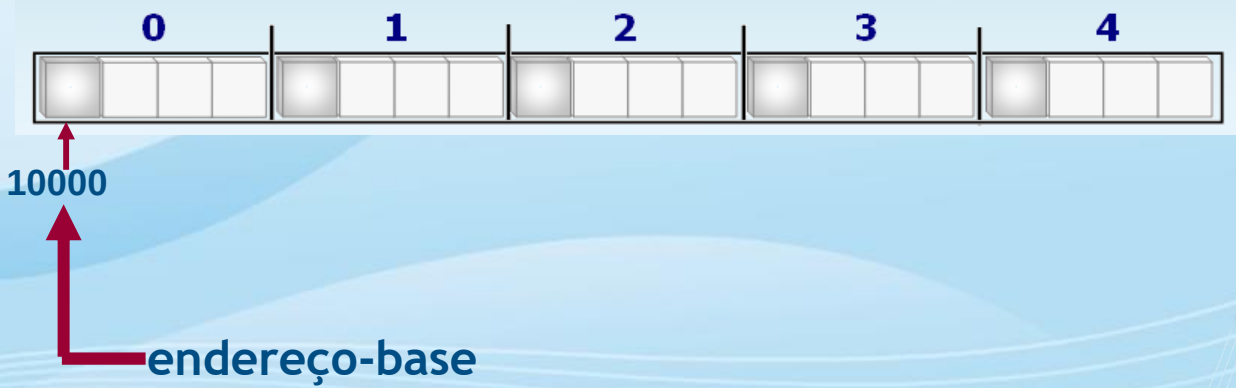


Armazenamento na Memória Principal

O nome de uma matriz corresponde ao primeiro endereço do conjunto de endereços da Memória Principal. Para localizarmos um elemento da matriz, usamos a fórmula abaixo.

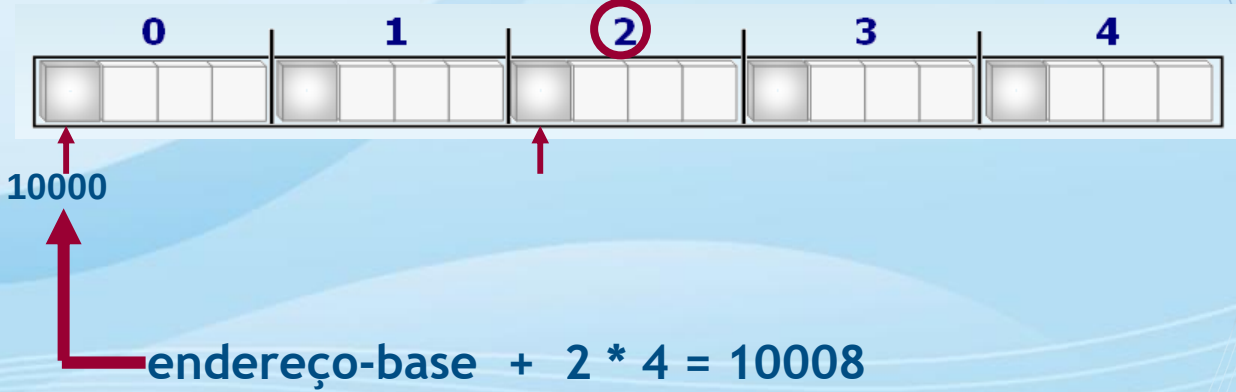
endereço-base + deslocamento * tamanho do tipo

Armazenamento na Memória Principal



endereço-base + deslocamento * tamanho do tipo

Armazenamento na Memória Principal



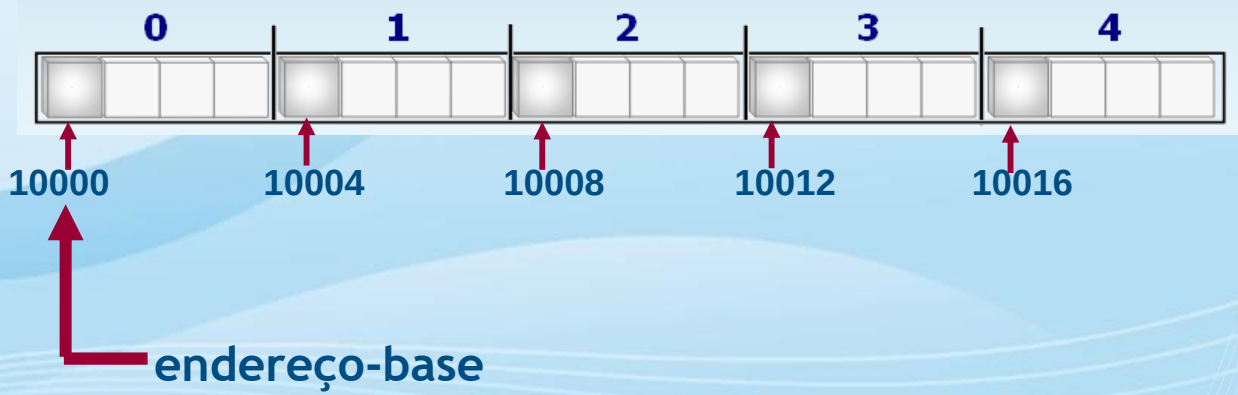
endereço-base + deslocamento * tamanho do tipo

Armazenamento na Memória Principal

endereço-base + 2 * 4 = 10008

endereço-base + deslocamento * tamanho do tipo

Armazenamento na Memória Principal



endereço-base + deslocamento * tamanho do tipo

Armazenamento na Memória Principal

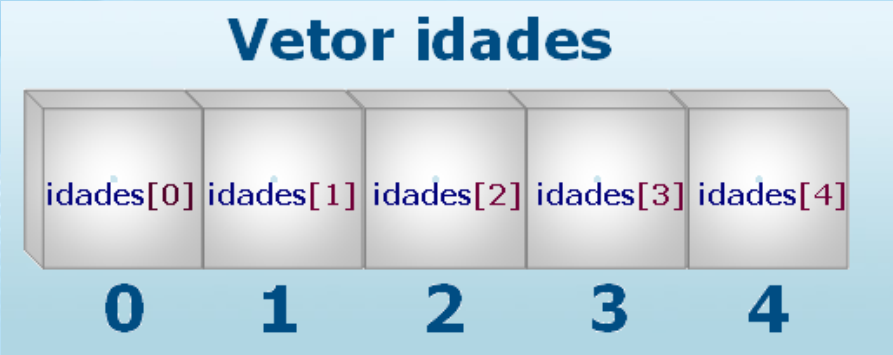
Conclui-se que **0** significa que não existe deslocamento em relação ao endereço-base.

Armazenamento na Memória Principal

E onde indicamos isso?

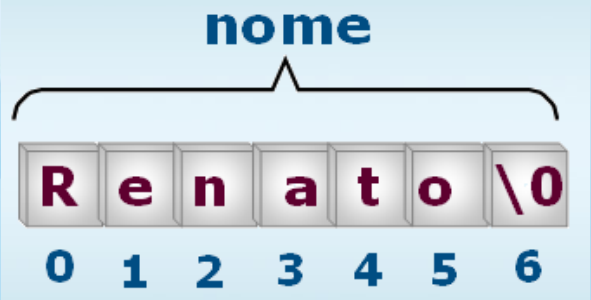
Armazenamento na Memória Principal

Observe o vetor idades[5]. Ele é formado por cinco variáveis todas com nome, idades. Dentro de um par de colchetes, fica o deslocamento.



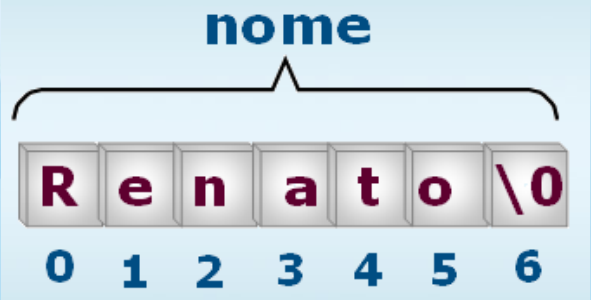
Armazenamento na Memória Principal

No Vetor de char...



Armazenamento na Memória Principal

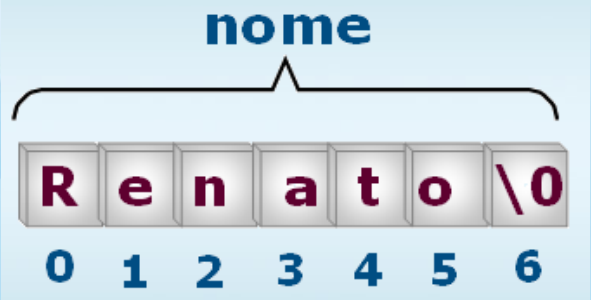
No Vetor de char...



nome[1] = toupper(nome[1]); ➡ REnato

Armazenamento na Memória Principal

No Vetor de char...



O \0 serve para finalizar o vetor de char. Dessa forma, não se esqueça de *acrescentar* mais uma posição quando dimensionar um vetor de char.

PROCEDIMENTOS E FUNÇÕES

```
#include<iostream>
#include<locale.h>
#include<iomanip>
#include<conio.h>
#include<windows.h>
using namespace std;

float fmedia(float n1, float n2) { //função
    float m = 0;
    m = (n1 + n2)/2;
    return m;
}

float pmedia(){ //procedimento
    float n1 = 0, n2 = 0, m = 0;
    cout << "\n Digite a nota 1 do aluno: ";
    cin >> n1;
    cout << "\n Digite a nota 2 do aluno: ";
    cin >> n2;
    m = (n1 + n2)/2;
    cout << "\n\n A média do aluno foi: " << fixed << setprecision(1) << m;
}

int main() {
    INICIO: system("CLS");
    setlocale(LC_ALL, "portuguese");
    float num1 = 0, num2 = 0;
    int op = 0;
    char esc;
    cout << "\n Escolha uma das opções abaixo:";
    cout << "\n ==================================";
    cout << "\n Exibir média por Function? Digite 1";
    cout << "\n Exibir média por Procedure? Digite 2";
    cout << "\n ==================================";
    NOV: cout << "\n Opção: ";
    cin.clear();
    fflush(stdin);
    cin >> op;
```

```
switch(op) {
    case 1:
        cout << "\n *** Pela FUNÇÃO ***\n";
        cout << "\n Digite a nota 1 do aluno: ";
        cin >> num1;
        cout << "\n Digite a nota 2 do aluno: ";
        cin >> num2;
        cout << "\n\n A média do aluno foi: " <<
            fixed << setprecision(1) << fmedia(num1, num2);
        break;

    case 2:
        cout << "\n *** Por PROCEDIMENTO ***\n";
        pmedia();
        break;

    default: cout << "\n (Opção INVÁLIDA !!!) \n";
             Sleep(1000);
             goto NOV;
}

PERG:;
cout << "\n\n Quer continuar? (Para SIM digite S, para NÃO digite N): ";
cin >> esc;
if (esc == 'S' || esc == 's') {
   goto INICIO;
}
if (esc=='N' || esc == 'n') {
   goto FIM;
}
if (esc!='S' || esc != 's' || esc != 'N' || esc != 'n') {
   goto PERG;
}
```

```
    FIM:
        cout << "\n\n";
        cout << "\e[?25l"; //esconde o cursor
        //cout << "\e[?25h"; // se quiser ativar o cursor
        for (int i=0; i < 100; i++) {
            cout << "\r                                                  ";
            Sleep(500);
            cout << "\r                          *** FIM DO PROGRAMA ***";
            Sleep(500);
        }

    getch();
}
```

Ao compilar e executar o código anterior temos:

Opção 1 escolhida

```
Escolha uma das opções abaixo:
==================================
Exibir média por Function? Digite 1
Exibir média por Procedure? Digite 2
==================================
Opção: 1

*** Pela FUNÇÃO ***

Digite a nota 1 do aluno: 7.3

Digite a nota 2 do aluno: 8.9


A média do aluno foi: 8.1

Quer continuar? (Para SIM digite S, para NÃO digite N): n


*** FIM DO PROGRAMA ***
```

Ao compilar e executar o código anterior temos:

Opção 2 escolhida

```
Escolha uma das opções abaixo:
==================================
Exibir média por Function? Digite 1
Exibir média por Procedure? Digite 2
==================================
Opção: 2

*** Por PROCEDIMENTO ***

Digite a nota 1 do aluno: 4.5

Digite a nota 2 do aluno: 8


A média do aluno foi: 6.2

Quer continuar? (Para SIM digite S, para NÃO digite N): n


*** FIM DO PROGRAMA ***
```

Particularidades na utilização de Funções em C/C++ (Programa com ERRO)

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

float soma(float n)
{   return dobra(n)+5;  }

float dobra(float n)
{  return n*2; }

int main()
{
   float num=20;
   cout << "\nApos chamar a funcao soma : “ << soma(num);
   cout << "\nApos chamar a funcao dobra: “ << dobra(num);
   cout << "\n\n";

  system("pause");
}
```

O ERRO

```
float soma(float n)
{   return dobra(n)+5;  }

float dobra (float n)
{  return n*2; }
```

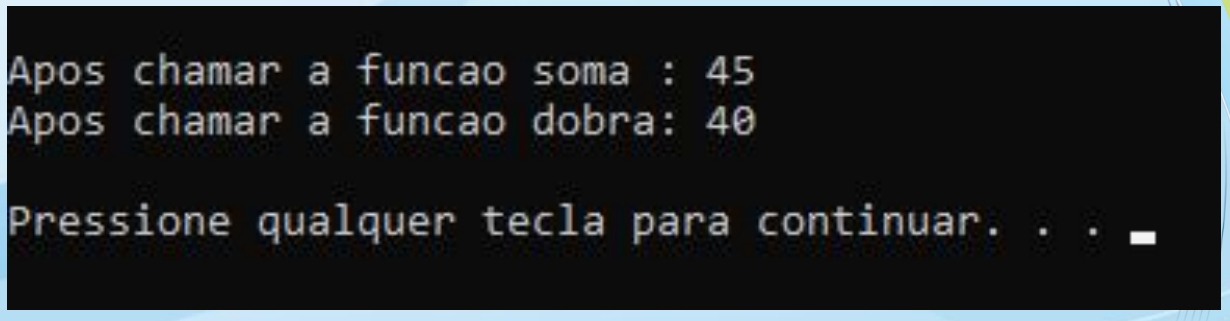
Line	Col	File	Message
		C:\Users\	In function 'float soma(float)':
8	19	C:\Users\	[Error] 'dobra' was not declared in this scope

Eu pensei que, se colocássemos antes da *main()*, nunca teriamos problemas. E agora?

A primeira solução, para esse problema, seria colocar a função *dobra(...)* antes da função *soma(...)*, mas, em outras situações, poderá ficar complicado.

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

float dobra(float n)
{  return n*2; }

float soma(float n)
{   return dobra(n)+5;  }

int main()
{
   float num=20;
   cout << "\nApos chamar a funcao soma : " << soma(num);
   cout << "\nApos chamar a funcao dobra: “ << dobra(num);
   cout << "\n\n";
   system("pause");
}
```

Ao compilar e executar o código anterior temos:

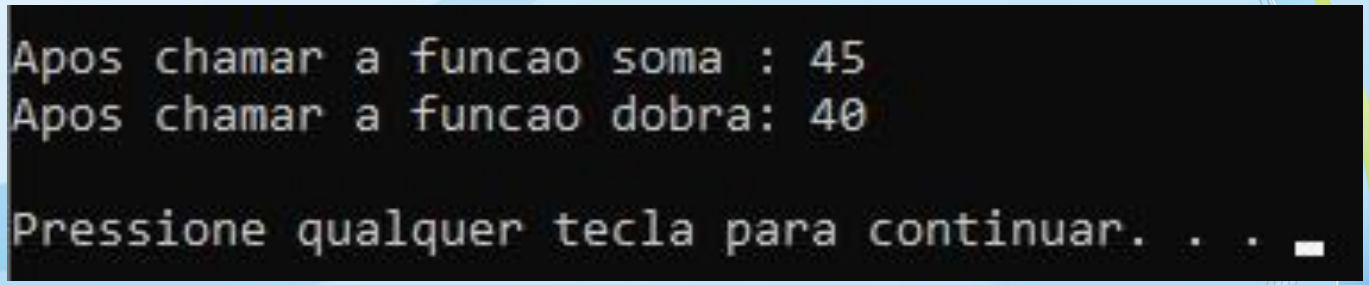


A segunda solução, para esse problema, seria **declarar (protótipos)** das funções antes de definí-las e, preferencialmente, posicioná-las depois da *main()*.

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

float soma(float n);
float dobra(float n);

int main()
{
   float num=20;
   cout<<"\nApos chamar a funcao soma : "<<soma(num);
   cout<<"\nApos chamar a funcao dobra: "<<dobra(num);
   cout<<"\n\n";
   system("pause");
}

float soma(float n)
{   return dobra(n)+5;  }

float dobra(float n)
{  return n*2; }
```

```
#include <iostream>
#include <cstdlib>
using namespace std;
float soma(float n);  // float soma(float); - Pode ser escrito assim.
float dobra(float n); // float dobra(float); - Pode ser escrito
assim.
int main()
{
   float num=20;
   cout<<"\nApos chamar a funcao soma : "<<soma(num);
   cout<<"\nApos chamar a funcao dobra: "<<dobra(num);
   cout<<"\n\n";
   system("pause");
}
float soma(float n)
{   return dobra(n)+5;  }
float dobra(float n)
{  return n*2; }
```

Ao compilar e executar o código anterior temos:



Vamos a algumas perguntas sobre o conteúdo dado até o momento.

1) Cite duas vantagens ao se definir funções?

2) O que contem o Cabeçalho de uma função?

3) O que é o protótipo de uma função em C++?

4) Onde o protótipo da função é colocado em C++?

5) Qual a vantagem do uso de protótipos de funções em nossos programas em C++?

6) O que significa um protótipo simplificado em C++?

1) Cite duas vantagens ao se definir funções?

1) Cite duas vantagens ao se definir funções?

R: Dividir o programa em partes menores e (re)usá-las em vários programas.

1) Cite duas vantagens ao se definir funções?

R: Dividir o programa em partes menores e (re)usá-las em vários programas.

2) O que contem o Cabeçalho de uma função?

1) Cite duas vantagens ao se definir funções?

R: Dividir o programa em partes menores e (re)usá-las em vários programas.

2) O que contem o Cabeçalho de uma função?

R: O tipo de função, o nome da função e a declaração dos parâmetros, quando houver.

3) O que é o protótipo de uma função em C++?

3) O que é o protótipo de uma função em C++?

R: É o cabeçalho da função com ; ao final.

3) O que é o protótipo de uma função em C++?

R: É o cabeçalho da função com ; ao final.

4) Onde o protótipo da função é colocado em C++?

3) O que é o protótipo de uma função em C++?

R: É o cabeçalho da função com ; ao final.

4) Onde o protótipo da função é colocado em C++?

R: Antes de todas as funções.

3) O que é o protótipo de uma função em C++?

R: É o cabeçalho da função com ; ao final.

4) Onde o protótipo da função é colocado em C++?

R: Antes de todas as funções.

5) Qual a vantagem do uso de protótipos de funções em nossos programas de C++?

3) O que é o protótipo de uma função em C++?

R: É o cabeçalho da função com ; ao final.

4) Onde o protótipo da função é colocado em C++?

R: Antes de todas as funções.

5) Qual a vantagem do uso de protótipos de funções em nossos programas de C++?

R: Não nos preocuparmos com as posições das funções.

6) O que significa um protótipo simplificado em C++?

R:Significa que os nomes dos parâmetros não estão presentes.

Exemplo:

```
float media(float, float);
```