### **GOVERNO DO ESTADO DO ESPÍRITO SANTO**

### **SECRETARIA ESTADUAL DE EDUCAÇÃO**

### **ENSINO MÉDIO TÉCNICO INTEGRADO INTEGRAL**

### **TURNO VESPERTINO INTERMEDIÁRIO – ANO LETIVO 2023**

### 

### **DISCIPLINA: ALGORITMO E LOGICA DE PROGRAMAÇÃO**

### **PROFESSORA: SANDRA VALIM**

[**GITHUB.COM/DRICASAVALIM**](http://github.com/DRICASAVALIM)

**NA GUIA ALGORITMO E LOGICA DE PROGRAMAÇÃO**

CÓDIGOS

2TRIAULA 01

2TRIAULA 02

2TRIAULA 03

2TRIAULA 04

### 

### **Apostila 2º Trimestre**

**MATRIZES**

Arrays são muito importantes em qualquer linguagem de programação. Eles fornecem uma maneira mais conveniente de armazenar variáveis ​​ou uma coleção de dados de um tipo de dados semelhante juntos, em vez de armazená-los separadamente. Cada valor do array será acessado separadamente.

Declarar um array em C++

A declaração de array em C++ envolve indicar o tipo e também o número de elementos a serem armazenados pelo array. Sintaxe:

type array-Name [ array-Size ];

Regras para declarar um array unidimensional em C++.

Tipo: O tipo é o tipo de elementos a serem armazenados na matriz e deve ser um tipo de dados C++ válido.

Nome da matriz: O array-Name é o nome a ser atribuído ao array.

Tamanho da matriz: O array-Size é o número de elementos a serem armazenados no array. Deve ser um número inteiro e maior que 0.

Por exemplo, você pode criar uma matriz chamada idade e armazenar as idades de 5 alunos da seguinte forma:

int age[5];

A matriz age armazenará 5 inteiros representando as idades de diferentes alunos.

Inicialização de Array

A inicialização do array é o processo de atribuir/armazenar elementos a um array. A inicialização pode ser feita em uma única instrução ou uma por uma. Observe que o primeiro elemento de um array é armazenado no índice 0, enquanto o último elemento é armazenado no índice n-1, onde n é o número total de elementos no array.

No caso do array age, o primeiro elemento será armazenado no índice 0, enquanto o último elemento será armazenado no índice 4.

Vamos usar o array age para demonstrar como a inicialização do array pode ser feita:

int age[5] = {19, 18, 21, 20, 17};

O número total de elementos em { } não pode exceder o valor indicado em [ ]. O elemento 19 está no índice 0, 18 no índice 1, 21 no índice 2, 20 no índice 3 e 17 no índice 4. Se você não indicar o número de elementos a serem armazenados na matriz dentro de [], a matriz será grande o suficiente apenas para conter os elementos adicionados em { }. Por exemplo:

int age[] = {19, 18, 21, 20, 17};

A instrução acima criará exatamente o mesmo array que o anterior. Você também pode atribuir um elemento a um array usando seu índice. Por exemplo:

age[3] = 20;

A instrução acima armazenará o valor 20 no índice 3 do array denominado idade. Isso significa que 20 serão os 4th elemento da matriz.

Tipos de matrizes

Existem dois tipos de matrizes C++:

Matriz unidimensional

Matriz multidimensional

Ponteiro para um array

Matriz Unidimensional

Esta é uma matriz na qual os itens de dados são organizados linearmente em apenas uma dimensão. É comumente chamado de array 1-D. Sintaxe:

datatype array-name[size];

O nome do array é o nome do array.

O tamanho é o número de itens a serem armazenados na matriz.

**Por exemplo:**

#include <iostream>

using namespace std;

int main()

{

int age[5] = { 19, 18, 21, 20, 17 };

for (int x = 0; x < 5; x++)

{

cout <<age[x]<<"\n";

}

}

Saída:

Matriz Unidimensional

Aqui está uma captura de tela do código:

Matriz Unidimensional

**Explicação do código:**

Incluindo o arquivo de cabeçalho iostream em nosso código. Isso nos permitirá ler e escrever no console.

Incluindo o namespace std para usar suas classes e funções sem chamá-lo.

Chamando a função main() dentro da qual deve ser adicionada a lógica do programa.

Início do corpo da função main().

Declarando um array chamado idade para armazenar 5 inteiros. Os 5 inteiros também foram inicializados.

Crie uma variável inteira x usando um loop for.

O início do corpo do loop for.

Usando a variável de loop x para iterar sobre os valores do array age e imprimi-los no console. O “\n” é um caractere de nova linha e é impresso em uma nova linha após cada iteração.

Fim do corpo do para laço.

Fim do corpo da função main().

O que é um Array (ou vetor) ?

O array, também chamado de vetor, é tipo de estrutura de dados (coleção de dados), do mesmo tipo. Ou seja, é um modo existente de trabalharmos com várias quantidades de variáveis.

Para que serve um vetor (ou array) ?

Imagine que você quer calcular a média de dois alunos da sua turma.

Basta declarar duas variáveis: a e b, e fazer (a+b)/2

E caso queira calcular a média de três alunos?

(a+b+c)/3

Para calcular a média de uma turma de 30 alunos?

Vai declarar 30 variáveis e fazer: (a+b+c+d+e+f+g....)/30 ?

Nem tem 30 letras no alfabeto.

É aí que entra o conceito de array.

Vamos simplesmente declarar um vetor de floats, por exemplo, de tamanho 30.

Ou seja, 30 variáveis do tipo float, vizinhas de memória, serão alocadas em sua máquina, de uma vez só, ao criarmos esse array.

Basicamente é para isso que serve um array: vamos aprender a declarar, inicializar, usar e manusear grandes blocos de informações de uma só vez, mexendo com 10, 10, mil ou 1 milhão de variáveis de uma vez só, de maneira bem automatizada e simples, através de arrays.

#include <iostream>

#include <vector>

using namespace std;

int main() {

// initializer list

vector<int> vector1 = {1, 2, 3, 4, 5};

// uniform initialization

vector<int> vector2{6, 7, 8, 9, 10};

// method 3

vector<int> vector3(5, 12);

cout << "vector1 = ";

// ranged loop

for (const int& i : vector1) {

cout << i << " ";

}

cout << "\nvector2 = ";

// ranged loop

for (const int& i : vector2) {

cout << i << " ";

}

cout << "\nvector3 = ";

// ranged loop

for (int i : vector3) {

cout << i << " ";

}

return 0;

}

**Adicione elementos a um vetor**

Para adicionar um único elemento a um vetor, usamos a push\_back()função. Ele insere um elemento no final do vetor. Por exemplo,

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> num {1, 2, 3, 4, 5};

cout << "Initial Vector: ";

for (const int& i : num) {

cout << i << " ";

}

// add the integers 6 and 7 to the vector

num.push\_back(6);

num.push\_back(7);

cout << "\nUpdated Vector: ";

for (const int& i : num) {

cout << i << " ";

}

return 0;

}

**Acesse os elementos de um vetor**

Em C++, usamos o número do índice para acessar os elementos do vetor. Aqui, usamos a at()função para acessar o elemento do índice especificado. Por exemplo,

#include <iostream>

#include <vector>

using namespace std;

int main() {

vector<int> num {1, 2, 3, 4, 5};

cout << "Element at Index 0: " << num.at(0) << endl;

cout << "Element at Index 2: " << num.at(2) << endl;

cout << "Element at Index 4: " << num.at(4);

return 0;

}

### **AGORA PRATIQUE DIGITANDO OS 5 CÓDIGOS RESOLVIDOS**

### **Atividade 1: Verificação de Números Pares e Ímpares em um Vetor**

**Descrição:** Crie um programa que leia 10 números inteiros e os armazene em um vetor. Em seguida, use uma estrutura if-else para verificar e exibir quais números são pares e quais são ímpares.

**Instruções:**

1. Leia 10 números inteiros e armazene-os em um vetor.
2. Percorra o vetor e utilize if-else para determinar se cada número é par ou ímpar.
3. Exiba os números pares e os números ímpares.

#include <iostream>

using namespace std;

int main() {

int numeros[10];

// Leitura dos números

for (int i = 0; i < 10; i++) {

cout << "Digite o número " << i+1 << ": ";

cin >> numeros[i];

}

// Verificação de pares e ímpares

cout << "Números pares: ";

for (int i = 0; i < 10; i++) {

if (numeros[i] % 2 == 0) {

cout << numeros[i] << " ";

}

}

cout << "\nNúmeros ímpares: ";

for (int i = 0; i < 10; i++) {

if (numeros[i] % 2 != 0) {

cout << numeros[i] << " ";

}

}

return 0;

}

### **Atividade 2: Soma dos Elementos de uma Matriz**

**Descrição:** Crie um programa que leia uma matriz 3x3 de inteiros e calcule a soma de todos os seus elementos.

**Instruções:**

1. Leia os elementos de uma matriz 3x3.
2. Calcule a soma de todos os elementos da matriz.
3. Exiba a soma total.

#include <iostream>

using namespace std;

int main() {

int matriz[3][3];

int soma = 0;

// Leitura dos elementos da matriz

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

cout << "Digite o elemento [" << i << "][" << j << "]: ";

cin >> matriz[i][j];

soma += matriz[i][j];

}

}

// Exibindo a soma total

cout << "A soma dos elementos da matriz é: " << soma << endl;

return 0;

}

### **Atividade 3: Maior e Menor Elemento em um Vetor**

**Descrição:** Crie um programa que leia 5 números inteiros e os armazene em um vetor. Determine o maior e o menor número no vetor utilizando if-else.

**Instruções:**

1. Leia 5 números inteiros e armazene-os em um vetor.
2. Percorra o vetor para encontrar o maior e o menor número.
3. Exiba o maior e o menor número.

#include <iostream>

using namespace std;

int main() {

int numeros[5];

int maior, menor;

// Leitura dos números

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

cout << "Digite o número " << i+1 << ": ";

cin >> numeros[i];

}

// Inicialização do maior e menor

maior = menor = numeros[0];

// Determinação do maior e menor número

for (int i = 1; i < 5; i++) {

if (numeros[i] > maior) {

maior = numeros[i];

}

if (numeros[i] < menor) {

menor = numeros[i];

}

}

// Exibindo os resultados

cout << "O maior número é: " << maior << endl;

cout << "O menor número é: " << menor << endl;

return 0;

}

### **Atividade 4: Diagonal Principal de uma Matriz**

**Descrição:** Crie um programa que leia uma matriz 3x3 e exiba os elementos da diagonal principal.

**Instruções:**

1. Leia os elementos de uma matriz 3x3.
2. Exiba os elementos da diagonal principal.

#include <iostream>

using namespace std;

int main() {

int matriz[3][3];

// Leitura dos elementos da matriz

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

cout << "Digite o elemento [" << i << "][" << j << "]: ";

cin >> matriz[i][j];

}

}

// Exibindo a diagonal principal

cout << "Elementos da diagonal principal: ";

for (int i = 0; i < 3; i++) {

cout << matriz[i][i] << " ";

}

cout << endl;

return 0;

}

### **Atividade 5: Contagem de Números Negativos em uma Matriz**

**Descrição:** Crie um programa que leia uma matriz 3x3 e conte quantos números negativos há na matriz.

**Instruções:**

1. Leia os elementos de uma matriz 3x3.
2. Conte quantos números negativos há na matriz.
3. Exiba a contagem de números negativos.

#include <iostream>

using namespace std;

int main() {

int matriz[3][3];

int contagemNegativos = 0;

// Leitura dos elementos da matriz

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

cout << "Digite o elemento [" << i << "][" << j << "]: ";

cin >> matriz[i][j];

if (matriz[i][j] < 0) {

contagemNegativos++;

}

}

}

// Exibindo a contagem de números negativos

cout << "A quantidade de números negativos na matriz é: " << contagemNegativos << endl;

return 0;

}

Estas atividades ajudam os alunos a praticar a leitura e manipulação de vetores e matrizes, além de aplicar a estrutura de controle if-else para resolver problemas específicos.