ESTUDO - PROVA

Conteúdo criado para revisão de material de prova segundo os slides apresentados em sala

➔ CONCEITOS BÁSICOS

◆ Sintaxe - Diretivas

- #include
  - Inclui o conteúdo de um arquivo de cabeçalho.

- #include <iostream>
  - Permite que você inclua funções de I/O.

- #include <vector>
  - Permite que você inclua

- #include <string>
  - Permite que você inclua

◆ Estrutura

- Int main(int argc, char** argv) {}
  - A função **main** é o ponto de entrada de um programa em C++. Quando você executa um programa, a primeira coisa que o sistema operacional faz é chamar esta função

- Int
  - É o tipo de retorno da função, retornando um valor inteiro para o S.O para indicar se o programa foi executado com sucesso.

- main
  - O nome padrão da função.

- int argc
  - Argument count (Contador de Argumento). É um inteiro que representa o número próprio de argumentos de linha de comando que são passados ao programa.

- char** argv[ ]
  - Argument vector (Vetor de argumentos). É uma array de strings que contém os próprios argumentos passados.

◆ Identificadores

- Case Sensitive
  - Isto significa que ela diferencia letras minúsculas de maiúsculas.

- Snake_Case
    - É uma conversão de nomenclatura onde palavras separadas por sublinhados (underline “ _ “).

◆ Namespace

- É um agrupamento de nomes, é usado para evitar conflitos entre nome de variáveis, funções e classes em bibliotecas diferentes.

- std
    - É o namespace padrão da biblioteca em C++.

- std :: cout
    - A sintaxe representa que você está acessando a função do sistema na linguagem (saída de console).

- std :: string
    - A sintaxe indica que você está usando a classe “String”, uma versão padrão da biblioteca.

➔ TIPOS PRIMITIVOS

◆ int
- Números inteiros

◆ double | float
- Números de pontos flutuantes

◆ bool
- Valores lógicos (**true** or **false**)

◆ char
- Caracteres únicos

➔ CONTROLE DE FLUXO CONDICIONAIS

◆ If Else
-
```
if (n > 0) {
    cout << "positivo\n";
} else {
    cout << "negativo\n";
}
```

◆ If Else / Else If

```
if (n > 0) {
    cout << "positivo\n";
} else if (n == 0) {
    cout << "zero\n";
} else {
    cout << "negativo\n";
}
```

◆ Operador Ternário

```
String resultado ( numero > 0  ? " POSITIVO " : " NEGATIVO " );
```

◆ Switch (escolha)

```
switch (dia) {
  case 1: cout << "Seg"; break;
  case 2: cout << "Ter"; break;
  case 3: cout << "Qua"; break;
  default: cout << "Outro"; // sem break no último é comum
}
```

➔ Estruturas de Repetição

◆ While

```
while (i <= 10) {
  cout << i << " ";
  i++;
}
```

◆ Do-While

```
do {
  cout << x << " ";
  x++;
} while (x < 1);
```

◆ For

```
for (int i = 1; i <= 10; i++) {
  cout << i << " ";
}
```

➔ VETORES (ARRAYS UNIDIMENSIONAIS)

◆ O vetor é a coleção de elementos dispostos em uma única linha

◆ Declaração

```
float salario[5];       // 5 posições: 0..4
salario[0] = 240.0f;
salario[4] = 1000.0f;
```

◆ Imprimir

```
for (int i = 0; i < N; i++) cin >> v[i];
for (int i = 0; i < N; i++) cout << v[i] << " ";
```

➔ MATRIZ (ARRAYS BIDIMENSIONAIS)

- É uma coleção de elementos dispostos em linhas e colunas

◆ Declaração

```
float x[3][5];     // 3 linhas, 5 colunas
x[0][0] = 240.0f;
x[2][3] = 1000.0f;
```

◆ Imprimir

```
for (int i = 0; i < L; i++)
  for (int j = 0; j < C; j++)
    cin >> m[i][j];

for (int i = 0; i < L; i++) {
  for (int j = 0; j < C; j++) cout << m[i][j] << " ";
  cout << "\n";
}
```

➔ FUNÇÕES / MÉTODOS

◆ É uma função que contém um bloco de código que executa uma tarefa específica

- Declaração e Definição
  - A declaração informa ao compilador sobre a função (contendo parâmetros e o tipo de retorno) e a definição contém o código real da função

- Parâmetros e Argumentos
  - Os parâmetros são variáveis locais da função que recebem os valores passados quando a função é chamada.

- Valor do retorno
  - Uma função pode retornar um valor de volta para o local onde foi chamado,utilizando a palavra “ **RETURN “**.

-

```
int soma( int A, int B ) {
  return A + B;
}
```

➔ FUNÇÃO RECURSIVA

- ◆ É uma função que chama a si mesma. Para ser considerada recursiva, deve obrigatoriamente chamar a si mesma uma vez. Pode não exigir uso de estruturas de repetição / condicionais, mas deve sempre haver uma saída para encerrar os loops.

-
```
int Fatorial( int X ){
    if ( X <= 1 ) {
        return 1;
    } else {
        return X * Fatorial( X - 1 );
    }
}
```

-
```
long long fib(int n) {
  if (n < 2) return n;          // 0, 1
  return fib(n-1) + fib(n-2);
}
```

➔ CLASSES E OBJETOS

◆ Classes

- São moldes para a criação do objeto, definindo seus atributos e métodos que o objeto terá

◆ Objetos

- São instâncias de classes que representam entidades do mundo real e são interativas entre si.

QUESTÕES

Resolução de exercícios

➔ VETORES

- Fazer um programa que leia vários números inteiros e positivos. A leitura se encerra quando encontrar um número negativo ou quando o vetor ficar completo. Gere e imprime um vetor onde cada elemento é o inverso do correspondente do vetor original

```cpp
#include <iostream>

int main() {
    const int TAMANHO = 10;
    int vetor[TAMANHO];
    int count = 0;

    std::cout << "Digite ate 10 numeros inteiros e positivos (negativo para parar):" << std::endl;
    for (int i = 0; i < TAMANHO; ++i) {
        int num;
        std::cin >> num;
        if (num < 0) {
            break;
        }
        vetor[i] = num;
        count++;
    }

    double vetorInverso[TAMANHO];
    std::cout << "\nVetor Original:" << std::endl;
    for (int i = 0; i < count; ++i) {
        std::cout << vetor[i] << " ";
        vetorInverso[i] = 1.0 / vetor[i];
    }

    std::cout << "\n\nVetor Inverso:" << std::endl;
    for (int i = 0; i < count; ++i) {
        std::cout << vetorInverso[i] << " ";
    }
    std::cout << std::endl;

    return 0;
```

```
}
```

- Fazer um programa que digite vários números no vetor de tamanho máximo de 100 elementos, até digitar o número “ 0 “. Imprima quantos números são iguais aos últimos números lidos. O limite de números é 100, sem considerar o “ 0 “ como último número

```
#include <iostream>

int main() {
    const int TAMANHO_MAX = 100;
    int vetor[TAMANHO_MAX];
    int ultimoNumero = -1;
    int count = 0;

    std::cout << "Digite numeros (maximo de 100). Digite 0 para parar." << std::endl;

    for (int i = 0; i < TAMANHO_MAX; ++i) {
        int num;
        std::cin >> num;
        if (num == 0) {
            break;
        }
        vetor[i] = num;
        ultimoNumero = num;
        count++;
    }

    int ocorrencias = 0;
    for (int i = 0; i < count; ++i) {
        if (vetor[i] == ultimoNumero) {
            ocorrencias++;
        }
    }

    std::cout << "O ultimo numero valido digitado foi " << ultimoNumero << "." << std::endl;
    std::cout << "Ele apareceu " << ocorrencias << " vezes no vetor." << std::endl;

    return 0;
}
```

- Crie um vetor com 8 elementos e ordená-los.

```
#include <iostream>
```

```cpp
#include <vector>
#include <algorithm>

int main() {
    const int TAMANHO = 8;
    int vetor[TAMANHO];

    std::cout << "Digite 8 numeros inteiros para o vetor:" << std::endl;
    for (int i = 0; i < TAMANHO; ++i) {
        std::cin >> vetor[i];
    }

    // Bubble Sort
    for (int i = 0; i < TAMANHO - 1; ++i) {
        for (int j = 0; j < TAMANHO - i - 1; ++j) {
            if (vetor[j] > vetor[j+1]) {
                // Troca os elementos
                int temp = vetor[j];
                vetor[j] = vetor[j+1];
                vetor[j+1] = temp;
            }
        }
    }

    std::cout << "\nVetor ordenado:" << std::endl;
    for (int i = 0; i < TAMANHO; ++i) {
        std::cout << vetor[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

- Criar um programa que crie um vetor com 10 posições e carregue com uma palavra, depois imprima o vetor de uma maneira que exiba a palavra ao contrário
    -

```cpp
 #include <iostream>
#include <string>

int main() {
    std::string palavra;
    std::cout << "Digite uma palavra (max 10 caracteres): ";
    std::cin >> palavra;

    std::cout << "Palavra ao contrario: ";
    for (int i = palavra.length() - 1; i >= 0; --i) {
        std::cout << palavra[i];
    }
    std::cout << std::endl;

    return 0;
```

```
}
```

➔ FUNÇÃO RECURSIVA

◆ Múltipla Escolha

- DEVE TER APENAS UM FOR?
- DEVE TER AO MENOS UMA ESTRUTURA DE SELEÇÃO?
- CHAMA A SI MESMO UMA VEZ?

◆ Faça uma função que retorne o dia de semana, que calcule o dia da semana usando a fórmula conhecida como “ Congruência de Zeller “.

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

int diaSemana(int d, int m, int a) {
    if (m < 3) {
        m += 12;
        a -= 1;
    }
    int k = a % 100;
    int j = a / 100;
    int h = (d + 13*(m+1)/5 + k + k/4 + j/4 + 5*j) % 7;
    return h; // 0 = sábado, 1 = domingo, ..., 6 = sexta
}
```

◆ Faça um programa que receba e imprima o dia, o mês e o ano. Este código deve ser chamado por uma função criada.

```
int main() {
    int dia, mes, ano;
    cout << "Digite dia mes ano: ";
    cin >> dia >> mes >> ano;

    int semana = diaSemana(dia, mes, ano);
    cout << "Dia da semana (0=Sabado, 1=Domingo,...,6=Sexta): "
<< semana << endl;

    return 0;
}
```

- Faça uma função que retorne se o ano é bissexto. O ano bissexto é divisível por 4 e por 400, mas não é divisível por 100.

    -

```
bool bissexto(int ano) {
    if ((ano % 400 == 0) || (ano % 4 == 0 && ano % 100 != 0)) {
        return true;
    }
    return false;
}
```