|  |  |
| --- | --- |
| C++ | PORTUGOL |
| Linguagem compilada é uma linguagem de programação em que o código fonte, nessa linguagem, é executado diretamente pelo sistema operacional ou pelo processador, após ser traduzido por meio de um processo chamado compilação, usando um programa de computador chamado compilador, para uma linguagem de baixo nível, como linguagem de montagem ou código de máquina. | Linguagem interpretada é uma linguagem de programação em que o código fonte nessa linguagem é executado por um programa de computador chamado interpretador, que em seguida é executado pelo sistema operacional ou processador. Mesmo que um código em uma linguagem passe pelo processo de compilação, a linguagem pode ser considerada interpretada se o programa resultante não for executado diretamente pelo sistema operacional ou processador. |

cout << ; cin >> ;

int matriz[3][3] = {0};

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

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

cin >> matriz[linha][coluna];

       }

   }

int vetor[5] = {0};

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

       cin>> vetor[indice];

   }

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

       cout<< vetor[indice] << endl;

   }

int vetor[5] = {0};

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

       cin>> vetor[indice];

   }

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

       cout<< vetor[indice] << endl;

|  |
| --- |
| ORDENAÇÃO BUBBLE SORT |
| Esta comparação pode ser feita através de diferentes tipos de dados: inteiros (int), lógicos (bool), cadeias (string) e etc;  A ideia principal é percorrer o array diversas vezes e a cada passada comparar o elemento da posição n com o elemento da posição n + 1, após a comparação é realizada a operação de *swap* quando necessário. |

|  |
| --- |
| Funções |
| Organização de código/Divisão de tarefas para solução do problema principal; Facilidade na programação/O código da função é executado sempre que ela for chamada através de outra função/ Sempre que uma função é chamada, a execução da função que a chamou é parada/São executadas todas as instruções da função chamada/Ao final da função, ela **retorna** para o código que a chamou;   |  | | --- | | Exemplo de ordenação de vetor por bubble sort | | for (int i = 1; i < tamanho; i++) {  for (int j = 0; j < tamanho; j++) {  if (vetor[j] > vetor[j + 1]) {  aux = vetor[j];  vetor[j] = vetor[j + 1];  vetor[j + 1] =aux;  }  }  } | |  | |

|  |
| --- |
| Exemplo de ordenação de matriz por bubble sort |
| /-----TRANSFORMANDO MATRIZ EM VETOR------  int i=0,x, y;  for (x=0; x<N; x+) { //assumindo que a matriz tenha N colunas e N linhas  for (y=0; y<N; y++) {  vetor[i]=matriz[x][y];  i++;  }  }  //-----TRANSFORMANDO VETOR EM MATRIZ------  int i=0,x, y;  for (x=0; x<N; x+) { //assumindo que a matriz tenha N colunas e N linhas  for (y=0; y<N; y++) {  matriz[x][y]=vetor[i];  i++; |

A diferença principal entre passagem de parâmetros por valor e por referência (&) é que quando é passada uma variável por valor, a mesma pode ser alterada na função onde é chamada, mas ao final da execução da função não houve modificação do valor.

No caso de passagem por referência(&) qualquer alteração realizada na variável na função onde foi chamada, ao finalizar a execução da função, modifica o valor.

\*Vetores e Matrizes (arrays) já trabalham com passagem por referência, portanto, não é necessário colocar o &.

\* Pode-se retornar somente UM valor, então não é permitido retornar vetores e matrizes.

Retorno de Função

\* Uma função pode retornar somente UM valor.

\* O valor retornado pode ser qualquer tipo de dado,

exceto uma função, um vetor ou uma matriz.

\* Não se pode retornar Vetores e Matrizes pois

possuem mais que um valor, e uma função retorna

somente um valor.

\* Uma função pode ter vários return, mas tão logo

encontre o primeiro return retorna para a sentença

que originou a chamada.

Int main(){ FIBONACCI

int a, b, auxiliar, i, n

a = 0; b = 1;

cout <<"Digite um número: ";

cin >>n;

cout <<"Série de Fibonacci:\n";

cin>>b;

for(i = 0; i < n; i++) { auxiliar = a + b; a = b;b = auxiliar;

cout << auxiliar;

}

}