Curso Técnico

de Programação de Jogos Digitais

Aula 04

índice

[1. Arrays e Coleções em C 3](#_Toc169072819)

[2. Arrays Multidimensionais 3](#_Toc169072820)

[3. Coleções Genéricas 4](#_Toc169072821)

[4. Coleções Não Genéricas 5](#_Toc169072822)

[5. Referencias 7](#_Toc169072823)

# Arrays e Coleções em C

Arrays e coleções são estruturas de dados que permitem armazenar e manipular vários valores de um mesmo tipo ou de tipos diferentes. Em C, existem dois tipos principais de arrays e coleções: os arrays simples e os arrays multidimensionais, e as coleções genéricas e as coleções não genéricas.

Um array simples é uma estrutura de dados que armazena uma sequência de valores de um mesmo tipo, em uma única dimensão. Um array simples tem um tamanho fixo, que deve ser especificado na sua declaração ou inicialização. Um array simples pode ser acessado por meio de um índice, que representa a posição do valor no array, começando em zero.

Para declarar um array simples em C, é preciso seguir a seguinte sintaxe:

tipo[] nome = new tipo[tamanho];

Onde tipo é o tipo de dado dos valores do array, nome é o identificador do array e tamanho é o número de elementos do array. O operador new é usado para criar uma instância do array na memória.

Para inicializar um array simples em C, é possível atribuir os valores dos elementos na sua declaração, usando a seguinte sintaxe:

tipo[] nome = {valor1, valor2, ..., valorN};

Onde tipo é o tipo de dado dos valores do array, nome é o identificador do array e valor1, valor2, ..., valorN são os valores dos elementos do array. O tamanho do array será determinado pelo número de valores fornecidos. Para acessar um elemento de um array simples em C, é preciso usar o nome do array seguido pelo índice do elemento entre colchetes, usando a seguinte sintaxe:

nome[índice]

Onde nome é o identificador do array e índice é a posição do elemento no array, começando em zero. O índice deve ser um valor inteiro ou uma expressão que resulte em um valor inteiro.

# Arrays Multidimensionais

Um array multidimensional é uma estrutura de dados que armazena uma sequência de valores de um mesmo tipo, em mais de uma dimensão. Um array multidimensional tem um tamanho fixo para cada dimensão, que deve ser especificado na sua declaração ou inicialização. Um array multidimensional pode ser acessado por meio de vários índices, que representam a posição do valor no array em cada dimensão.

Para declarar um array multidimensional em C, é preciso seguir a seguinte sintaxe:

tipo[,] nome = new tipo[tamanho1,tamanho2];

Onde tipo é o tipo de dado dos valores do array, nome é o identificador do array e tamanho1 e tamanho2 são os números de elementos do array nas primeiras e segundas dimensões, respectivamente. O operador new é usado para criar uma instância do array na memória. Para inicializar um array multidimensional em C, é possível atribuir os valores dos elementos na sua declaração, usando a seguinte sintaxe:

tipo[,] nome = {{valor11, valor12, ..., valor1N}, {valor21, valor22, ..., valor2N}, ..., {valorM1, valorM2, ..., valorMN}};

Onde tipo é o tipo de dado dos valores do array, nome é o identificador do array e valor11, valor12, ..., valor1N são os valores dos elementos do array na primeira linha da primeira dimensão; valor21, valor22, ..., valor2N são os valores dos elementos do array na segunda linha da primeira dimensão; e assim por diante. O tamanho do array será determinado pelo número de linhas e colunas fornecidas. Para acessar um elemento de um array multidimensional em C, é preciso usar o nome do array seguido pelos índices do elemento entre colchetes, separados por vírgulas, usando a seguinte sintaxe:

nome[índice1,índice2]

Onde nome é o identificador do array e índice1 e índice2 são as posições do elemento no array nas primeiras e segundas dimensões, respectivamente. Os índices devem ser valores inteiros ou expressões que resultem em valores inteiros.

# Coleções Genéricas

Uma coleção genérica é uma estrutura de dados que armazena uma sequência de valores de um tipo específico ou genérico, que pode ser definido na sua declaração ou inicialização. Uma coleção genérica tem um tamanho variável, que pode ser alterado dinamicamente durante a execução do programa. Uma coleção genérica pode ser acessada por meio de um índice, que representa a posição do valor na coleção, começando em zero, ou por meio de um iterador, que permite percorrer os valores da coleção.

Para declarar uma coleção genérica em C, é preciso seguir a seguinte sintaxe:

Classe<Tipo> nome = new Classe<Tipo>();

Onde Classe é o nome da classe que representa a estrutura da coleção, Tipo é o tipo de dado dos valores da coleção e nome é o identificador da coleção. O operador new é usado para criar uma instância da coleção na memória.

Para inicializar uma coleção genérica em C, é possível atribuir os valores dos elementos na sua declaração, usando a seguinte sintaxe:

Classe<Tipo> nome = new Classe<Tipo>() {valor1, valor2, ..., valorN};

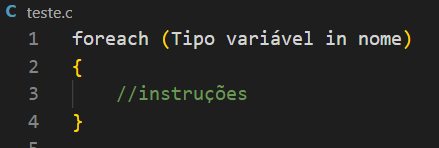
Onde Classe é o nome da classe que representa a estrutura da coleção, Tipo é o tipo de dado dos valores da coleção, nome é o identificador da coleção e valor1, valor2, ..., valorN são os valores dos elementos da coleção.

Para acessar um elemento de uma coleção genérica em C, é preciso usar o nome da coleção seguido pelo índice do elemento entre colchetes, usando a seguinte sintaxe:

nome[índice]

Onde nome é o identificador da coleção e índice é a posição do elemento na coleção, começando em zero. O índice deve ser um valor inteiro ou uma expressão que resulte em um valor inteiro.

Para percorrer os elementos de uma coleção genérica em C, é possível usar uma estrutura de repetição foreach, que usa um iterador para obter cada valor da coleção, usando a seguinte sintaxe:



Onde Tipo é o tipo de dado dos valores da coleção, variável é o nome da variável que recebe cada valor da coleção, nome é o identificador da coleção e instruções são as linhas de código que executam alguma ação com cada valor da coleção.

Existem várias classes que representam diferentes tipos de coleções genéricas em C, como List<T>, Dictionary<TKey,TValue>, Queue<T>, Stack<T>, HashSet<T>, LinkedList<T>, entre outras. Cada classe tem suas próprias características, métodos e propriedades que permitem manipular os valores da coleção de forma adequada.

# Coleções Não Genéricas

Uma coleção não genérica é uma estrutura de dados que armazena uma sequência de valores de tipos diferentes, em uma única dimensão. Uma coleção não genérica tem um tamanho variável, que pode ser alterado dinamicamente durante a execução do programa. Uma coleção não genérica pode ser acessada por meio de um índice, que representa a posição do valor na coleção, começando em zero, ou por meio de um iterador, que permite percorrer os valores da coleção.

Para declarar uma coleção não genérica em C, é preciso seguir a seguinte sintaxe:

Classe nome = new Classe();

Onde Classe é o nome da classe que representa a estrutura da coleção e nome é o identificador da coleção. O operador new é usado para criar uma instância da coleção na memória. Para inicializar uma coleção não genérica em C, é possível atribuir os valores dos elementos na sua declaração, usando a seguinte sintaxe:

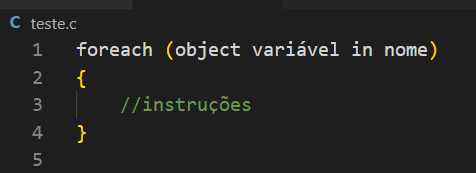
Classe nome = new Classe() {valor1, valor2, ..., valorN};

Onde Classe é o nome da classe que representa a estrutura da coleção, nome é o identificador da coleção e valor1, valor2, ..., valorN são os valores dos elementos da coleção. Para acessar um elemento de uma coleção não genérica em C, é preciso usar o nome da coleção seguido pelo índice do elemento entre colchetes, usando a seguinte sintaxe:

nome[índice]

Onde nome é o identificador da coleção e índice é a posição do elemento na coleção, começando em zero. O índice deve ser um valor inteiro ou uma expressão que resulte em um valor inteiro.

Para percorrer os elementos de uma coleção não genérica em C, é possível usar uma estrutura de repetição foreach, que usa um iterador para obter cada valor da coleção, usando a seguinte sintaxe:



Onde object é o tipo de dado genérico que representa qualquer tipo de valor, variável é o nome da variável que recebe cada valor da coleção, nome é o identificador da coleção e instruções são as linhas de código que executam alguma ação com cada valor da coleção.

Existem várias classes que representam diferentes tipos de coleções não genéricas em C, como ArrayList, Hashtable, Queue, Stack, SortedList, entre outras. Cada classe tem suas próprias características, métodos e propriedades que permitem manipular os valores da coleção de forma adequada.

No entanto, as coleções não genéricas têm algumas desvantagens em relação às coleções genéricas, como:

* Elas requerem mais conversões de tipo (casting) para acessar os valores da coleção, o que pode causar erros em tempo de execução se o tipo não for compatível.
* Elas não têm verificação de tipo em tempo de compilação, o que pode permitir a inserção de valores de tipos diferentes na mesma coleção, causando inconsistências e dificuldades na manipulação dos dados.
* Elas têm um desempenho inferior às coleções genéricas, pois usam mais memória e processamento para armazenar e acessar os valores da coleção.

Por esses motivos, as coleções genéricas são preferíveis às coleções não genéricas na maioria dos casos. As coleções não genéricas devem ser usadas apenas quando for necessário armazenar valores de tipos diferentes na mesma estrutura.

# Referencias

1. Unity Technologies. (2020). **Unity User Manual.** Unity Technologies.
2. [C#, 2020] **Visual C# Developer Center**, Microsoft Docs.

Microsoft Corporation. (2020).

1. Geig, M. (2018). **Unity 2018 Game Development in 24 Hours**, Sams Teach Yourself. Sams Publishing.
2. Hocking, J. (2015). **Unity in Action: Multiplatform Game Development in C#.** Manning Publications.