Aula 13

Prof: Henrique Augusto Maltauro

Desenvolvendo Algoritmos

Estruturas de Condição

Estruturas de Condição

As estruturas de condição definem blocos de código com vários caminhos possíveis, a serem executados com base no valor de uma expressão.

Dentro do C#, assim como na maior parte das linguagens de programação, nós temos três estruturas de condição:

- if
- else
- switch

Estruturas de Condição

- C#: if

A estrutura if define um bloco de código a ser executado com base no valor de uma expressão booleana.

Estruturas de Condição

- C#: if

```
public static void Executar(int numero)
{
    if (numero > 10)
    {
        // Bloco de código a ser executado se a expressão for verdadeira
    }
}
```

Estruturas de Condição

- C#: else

Uma estrutura if pode ser combinada com a estrutura else para definir dois blocos de códigos distintos, que serão executados com base na expressão booleana.

Estruturas de Condição

- C#: else

```
public static void Executar(int numero)
{
    if (numero > 10)
    {
        // Bloco de código a ser executado se a expressão for verdadeira
    }
    else
    {
        // Bloco de código a ser executado se a expressão for falsa
    }
}
```

Estruturas de Condição

- C#: switch

A estrutura switch define uma lista de blocos de códigos a serem executados com base em uma expressão de igualdade.

Estruturas de Condição

- C#: switch

```
public static void Executar(int numero)
{
    switch (numero)
    {
        case 1:
            // Bloco de código a ser executado se o numero for igual a 1
            break;
        case 2:
            // Bloco de código a ser executado se o numero for igual a 2
            break;
        default:
            // Bloco de código a ser executado se o numero não for igual a nenhuma das opções
            break;
    }
}
```

Estruturas de Repetição

Estruturas de Repetição

As estruturas de repetição definem um bloco de código que vai ser executado várias vezes.

Dentro do C#, assim como na maior parte das linguagens de programação, nós temos quatro estruturas de repetição:

- while
- do
- for
- foreach

Estruturas de Repetição

- C#: while

A estrutura while define um bloco de código que é executado zero ou mais vezes, de acordo com uma expressão booleana.

Estruturas de Repetição

- C#: while

```
public static void Executar(int numero)
{
    while (numero > 10)
    {
        // Bloco de código
    }
}
```

Estruturas de Repetição

- C#: do

A estrutura do define um bloco de código que é executado uma ou mais vezes, de acordo com uma expressão booleana.

Estruturas de Repetição

- C#: do

```
public static void Executar(int numero)
{
    do
    {
        // Bloco de código
    } while (numero > 10);
}
```

Estruturas de Repetição

- C#: for

A estrutura for define um bloco de código que é executado enquanto uma expressão booleana especificada é avaliada como verdadeira.

Essa estrutura é composta de três elementos:

- Inicializador
- Condição
- Iterador

Estruturas de Repetição

- C#: for (inicializador)

O elemento inicializador é executado apenas uma vez, antes de entrar na estrutura.

Normalmente, é declarado e inicializado uma variável local que será utilizada no elemento da condição.

Estruturas de Repetição

- C#: for (condição)

O elemento condição define uma expressão booleana, que determina se a próxima repetição da estrutura deve ser executada.

Se ela for avaliada como verdadeira, a próxima repetição será executada, caso contrário, a repetição será encerrada.

Estruturas de Repetição

- C#: for (iterador)

O elemento iterador define o que acontece após cada execução do bloco de código da estrutura.

Estruturas de Repetição

- C#: for

```
public static void Executar(int numero)
{
    for (int i = 0; i <= 10; i++)
    {
        // Bloco de código
    }
}
```

Estruturas de Repetição

- C#: foreach

Essa estrutura NÃO vai ser explicada aqui, e vou deixar para abordar essa estrutura mais pra frente quando formos estudar Listas e Arrays no C#.

Exercício

Exercício

1 - Receber dois números double, um para representar a vida do personagem, e o outro para representar o ataque.

Apresentar no console se o ataque matou ou não o personagem.