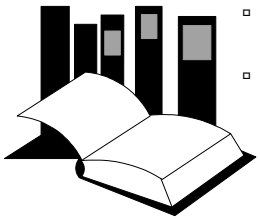
Pensamento Recursivo

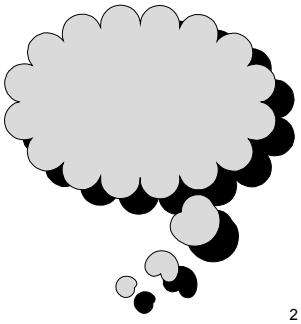


- Esta seção introduz a técnica de programação recursiva.
- Como você sabe, programação recursiva envolve ocorrências menores do mesmo problema.

Prof. Dr. José Augusto Baranauskas

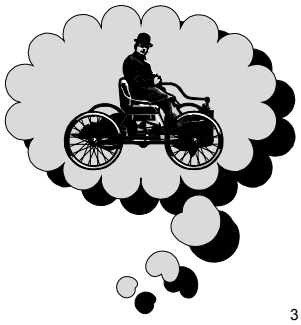
O Objeto Carro

- Para iniciar o exemplo, pense em seu carro favorito.



O Objeto Carro

- Para iniciar o exemplo, pense em seu carro favorito.

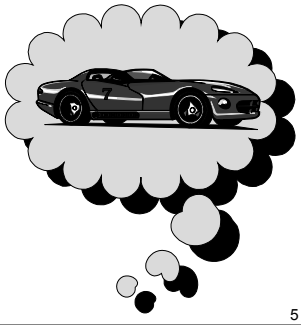


O Objeto Carro

- Para iniciar o exemplo, pense em seu carro favorito.

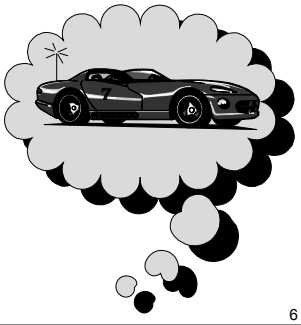
O Objeto Carro

- Para iniciar o exemplo, pense em seu carro favorito.



O Objeto Carro

- Para iniciar o exemplo, pense em seu carro favorito.
- Imagine que o carro é controlado por um sinal de rádio vindo de um computador.



O Objeto Carro

- Para iniciar o exemplo, pense em seu carro favorito.
- Imagine que o carro é controlado por um sinal de rádio vindo de um computador.
- Os sinais de rádio são gerados ativando-se os procedimentos de um Carro.

```
struct Car
{
 . . .

};
```

7

Procedimentos para o Objeto Carro

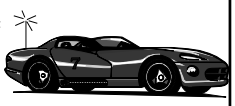
```
struct Car
{  // Nós não precisamos saber sobre campos!
     . . .
};

void MakeConnection(Car C, int CarNumber);
void Move(Car C);
void TurnAround(Car C);
boolean Blocked(Car C);
```

8

Exemplos de Chamada de Procedimento

```
int main
{ Car Racer;

  MakeConnection(Racer,7);

  . . .


}
```

Quando nós ativamos MakeConnection, o computador faz uma conexão ou link com o carro que tem um número exclusivo.

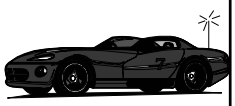


9

Exemplos de Chamada de Procedimento

```
int main
{ Car Racer;

  MakeConnection(Racer,7);
  TurnAround(Racer);
  . . .


}
```

Quando nós ativamos TurnAround, o sinal do computador avisa o carro para virar 180 graus.



10

Exemplos de Chamada de Procedimento

```
int main
{ Car Racer;

  MakeConnection(Racer,7);
  TurnAround(Racer);
  Move(Racer);
  . . .


}
```

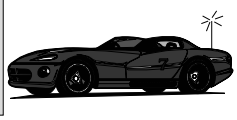
Quando nós ativamos Move, o sinal do computador diz ao carro para mover-se para a frente um metro.

11

Exemplos de Chamada de Procedimento

```
int main
{ Car Racer;

  MakeConnection(Racer,7);
  TurnAround(Racer);
  Move(Racer);
  . . .


}
```

Quando nós ativamos Move, o sinal do computador diz ao carro para mover-se para a frente um metro.

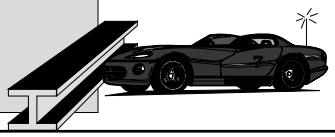


12

Exemplos de Chamada de Procedimento

```
int main
{ Car Racer;

  MakeConnection(Racer,7);
  TurnAround(Racer);
  Move(Racer);
  if (Blocked(Racer))
    cout << ”Impossível mover!";
  . . .

}
```

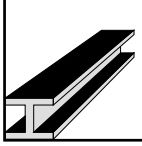
O método Blocked é uma função que detecta barreiras.

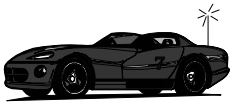


13

Sua Missão

- Escreva um procedimento que mova um carro para a frente até que ele encontre uma barreira ...

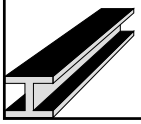


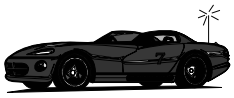


14

Sua Missão

- Escreva um procedimento que mova um carro para a frente até que ele encontre uma barreira ...

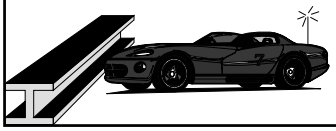




15

Sua Missão

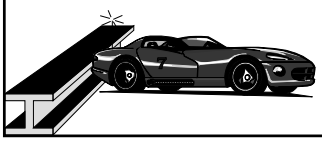
- Escreva um procedimento que mova um carro para a frente até que ele encontre uma barreira ...



16

Sua Missão

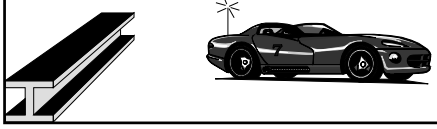
- Escreva um procedimento que mova um carro para a frente até que ele encontre uma barreira ...
- ... então o carro é virado novamente...



17

Sua Missão

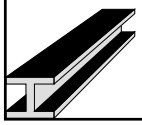
- Escreva um procedimento que mova um carro para a frente até que ele encontre uma barreira ...
- ... então o carro é virado novamente...
- ...e retorna à sua posição inicial percorrendo o caminho oposto.



18

Sua Missão

- Escreva um procedimento que mova um carro para a frente até que ele encontre uma barreira ...
- ... então o carro é virado novamente...
- ...e retorna à sua posição inicial percorrendo o caminho oposto.

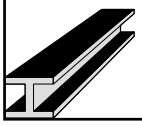


19

Sua Missão

```
void Ricochet(Car MovingCar);
```

- Escreva um procedimento que mova um carro para a frente até que ele encontre uma barreira ...
- ... então o carro é virado novamente...
- ...e retorna à sua posição inicial percorrendo o caminho oposto.



MovingCar

20

Pseudo-código para Ricochet

```
void Ricochet(Car MovingCar);
```

- Se Blocked(MovingCar) é verdadeiro, então o carro está próximo à barreira. Neste caso, apenas vire o carro.

21

Pseudo-código para Ricochet

```
void Ricochet(Car MovingCar);
```

- Se Blocked(MovingCar) é verdadeiro, então o carro está próximo à barreira. Neste caso, apenas vire o carro.
- Caso contrário, o carro ainda não chegou na barreira, então comece com:

```
Move(MovingCar);
. . .
```

22

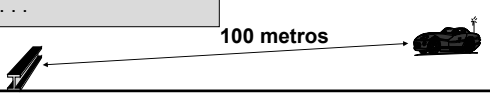
Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
. . .
```

Isso torna o problema um **pouco menor**. Por exemplo, se o carro começar a uma distância de 100 metros da barreira ...



23

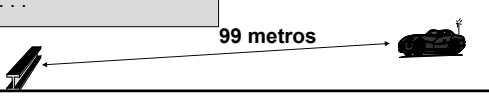
Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
. . .
```

Isso torna o problema um **pouco menor**. Por exemplo, se o carro começar a uma distância de 100 metros da barreira ... então após ativar Move um vez, a distância passa a ser 99 metros.



24

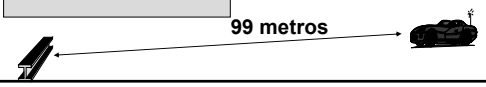
Pseudo-código para Ricochet

void Ricochet(Car Moving

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
. . .
```

Agora nós temos uma versão **menor** deste **mesmo problema**.



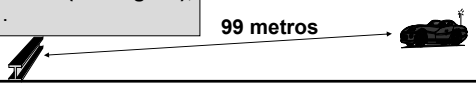
Pseudo-código para Ricochet

void Ricochet(Car Moving

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

Faça uma chamada recursiva para resolver o pequeno problema ...



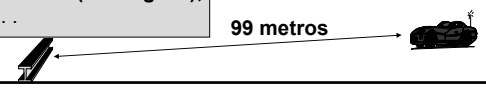
Pseudo-código para Ricochet

void Ricochet(Car Moving

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.



Pseudo-código para Ricochet

void Ricochet(Car Moving

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.

Pseudo-código para Ricochet

void Ricochet(Car Moving

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.

Pseudo-código para Ricochet

void Ricochet(Car Moving

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.

Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving(está próximo à barrei carro.
- Caso contrário, o car então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.

31

Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving(está próximo à barrei carro.
- Caso contrário, o car então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.

32

Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving(está próximo à barrei carro.
- Caso contrário, o car então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.

33

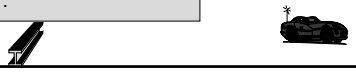
Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving(está próximo à barrei carro.
- Caso contrário, o car então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.



34

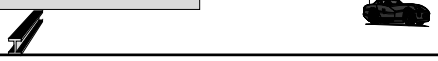
Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving(está próximo à barrei carro.
- Caso contrário, o car então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.



35

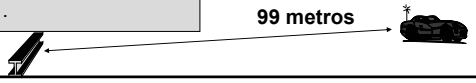
Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving(está próximo à barrei carro.
- Caso contrário, o car então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

A chamada recursiva resolverá o problema.



36

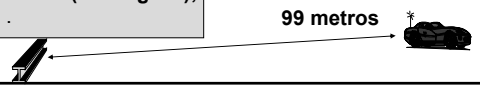
Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
. . .
```

Qual é o último passo necessário para retornar à nossa posição original?



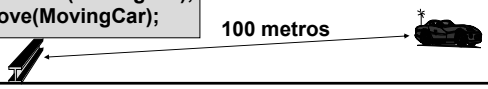
Pseudo-código para Ricochet

```
void Ricochet(Car Moving
```

- Se Blocked(Moving
  está próximo à barrei
  carro.
- Caso contrário, o car
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
Move(MovingCar);
```

Qual é o último passo necessário para retornar à nossa posição original?



Pseudo-código para Ricochet

```
void Ricochet(Car MovingCar);
```

- Se Blocked(MovingCar) é verdadeiro, então o carro está próximo à barreira. Neste caso, apenas vire o carro.
- Caso contrário, o carro ainda
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
Move(MovingCar);
```

Este procedimento recursivo segue um padrão comum que você deve reconhecer.

Pseudo-código para Ricochet

```
void Ricochet(Car MovingCar);
```

- Se Blocked(MovingCar) verdadeiro, então o carro está próximo à barreira. Neste caso, apenas vire o carro.
- Caso contrário, o carro ainda
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
Move(MovingCar);
```

Quando o problema é simples, resolva-o sem chamadas recursivas. Este é o **caso base**.

Pseudo-código para Ricochet

```
void Ricochet(Car MovingCar);
```

- Se Blocked(MovingCar) é verdadeiro, então o carro está próximo à barreira. Neste caso, apenas vire o carro.
- Caso contrário, o carro ainda
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
Move(MovingCar);
```

Quando o problema é mais complexo, comece criando uma versão **menor** do **mesmo problema**...

Pseudo-código para Ricochet

```
void Ricochet(Car MovingCar);
```

- Se Blocked(MovingCar) é verdadeiro, então o carro está próximo à barreira. Neste caso, apenas vire o carro.
- Caso contrário, o carro ainda
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
Move(MovingCar);
```

...e uma **chamada recursiva** para resolver totalmente o problema menor...

Pseudo-código para Ricochet

```
void Ricochet(Car MovingCar);
```

- Se Blocked(MovingCar) é verdadeiro, então o carro está próximo à barreira. Neste caso, apenas vire o carro.
- Caso contrário, o carro ainda
  então comece com:

```
Move(MovingCar);
Ricochet(MovingCar);
Move(MovingCar);
```

... e finalmente faça quaisquer operações que sejam necessárias para **completar a solução do problema original**

Implementação de Ricochet

```
void Ricochet(Car MovingCar)
{
   if (Blocked(MovingCar))          // caso base
      TurnAround(MovingCar);
   else                             // caso recursivo
   {  Move(MovingCar);
      Ricochet(MovingCar);
      Move(MovingCar);
   }
}
```

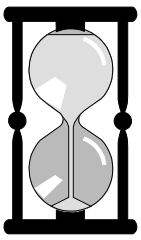

Um Exercício

Escreva um procedimento recursivo para calcular o fatorial de um número inteiro. Lembre-se:

0! = 1! = 1

n! = n * (n-1)!, se n > 1

```
int fatorial(int n)
{


}
```



Você tem 2 minutos para escrever a declaração.

Um Exercício

Uma Solução Recursiva

```
int fatorial(int n)
{
   if (n <= 1)          // caso base
      return 1;
   else                 // caso recursivo
      return n * fatorial(n-1);

}
```

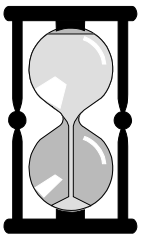

Outro Exercício

Escreva um procedimento **não** recursivo para calcular o fatorial de um número inteiro. Lembre-se:

0! = 1! = 1

n! = n * (n-1)!, se n > 1

```
int fatorial(int n)
{


}
```



Você tem 3 minutos para escrever a declaração.

Outro Exercício

Uma Solução Não Recursiva

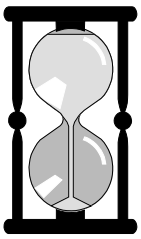
```
int fatorial(int n)
{  int i, produto = 1;

   for (i = 2; i <= n; i++)
      produto *= i;

   return produto;

}
```

Um Desafio

Um algoritmo muito conhecido para determinar o maior divisor comum de dois inteiros é o algoritmo de Euclides

Escreva um procedimento **recursivo** e outro **não recursivo** para calcular o MDC

$$MDC(m,n) = \begin{cases} MDC(n,m) & \text{se } n > m \\ m & \text{se } n = 0 \\ MDC(n, m \% n) & \text{se } n > 0 \end{cases}$$



Você tem 5 minutos para escrever a declaração.

Um Desafio

Uma Solução Recursiva

```
int mdc(int m, int n)
{
  if (n == 0)          // caso base
    return m;
  else                 // caso recursivo
    if (n > m)
      return mdc(n,m);
    else
      if (n > 0)
        return mdc(n, m % n);
}
```


Um Desafio

Uma Solução Não Recursiva

```
int mdc(int m, int n)
{
                    ?
}
```

Para a solução não recursiva deste problema, você precisa conhecer o TAD pilha!

Resumo

- Recursão é um recurso útil quando o problema a ser solucionado é definido de forma recursiva
- Programas recursivos são *elegantes* na sua definição, expressando concisamente o problema em termos de subproblemas
- Todo programa recursivo pode ser escrito de forma não recursiva (iterativa) e vice-versa
- Nos casos mais complexos, para transformar um programa recursivo num não recursivo é necessário o uso do TAD pilha
- O uso da recursão em problemas inerentemente iterativos pode gerar programas ineficientes

THE END