Recursividade

Prof. Andrei Braga
Prof. Geomar Schreiner

Recursividade

- Processo de definir algo utilizando a si mesmo.
  - Podemos dizer que o conceito de algo recursivo está dentro de si mesmo, sendo assim, esta dentro de si mesmo, e assim por diante

Recursividade

- Processo de definir algo utilizando
  - Podemos dizer que o conceito de algo
    dentro de si mesmo, e assim por diant

Recursividade

- Processo de definir algo utilizando a si mesmo.
  - Podemos dizer que o conceito de algo recursivo está dentro de si mesmo, sendo assim, esta dentro de si mesmo, e assim por diante

- Porque?
  - Programas recursivos são, em geral, mais simples de escrever, analisar e entender

Recursividade

- A dificuldade em utilizar a recursão está envolvida na forma com que o computador se comporta com ela, temos que entender o comportamento interno para poder criar algoritmos bons.
- A memória de um programa se comporta como uma pilha
  - Não vamos ver ela com todos os detalhes, mas, em outras matérias isso vai acontecer.
- A memória vai conter uma pilha onde cada elemento representa uma função.
  - Assim que o programa for iniciado, a função *main* é empilhada
    - Além da pilha de funções existe um ponteiro para onde o programa está executando, área para armazenamento das variáveis (globais e locais)

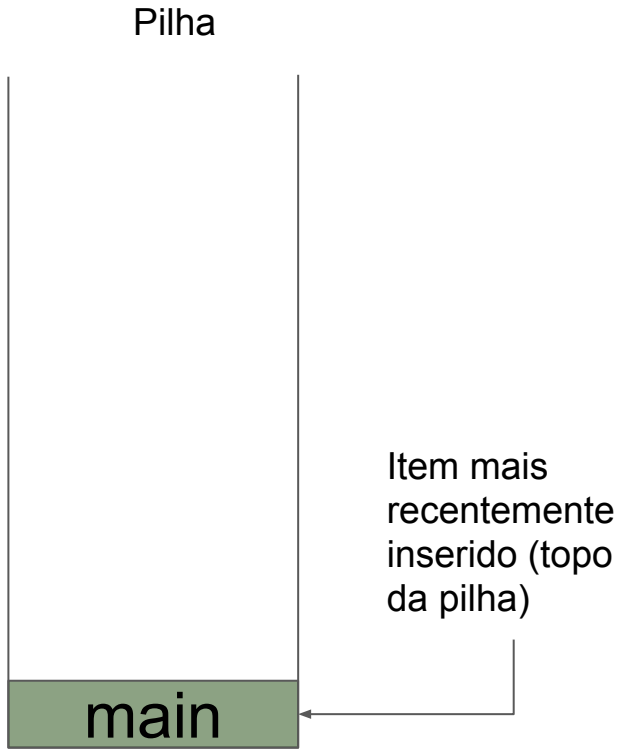
Recursividade

- A dificuldade em utilizar a recursão está envolvida na forma com que o computador se comporta com ela, temos que entender o comportamento interno para poder criar algoritmos bons.
- A memória de um programa se comporta como uma pilha
  - Não vamos ver ela com todos os detalhes, mas, em outras
- A memória vai conter uma pilha onde cada elemer
  - Assim que o programa for iniciado, a função *main* é empilh
    - Além da pilha de funções existe um ponteiro para on área para armazenamento das variáveis (globais e lo

Recursividade

- Exemplo

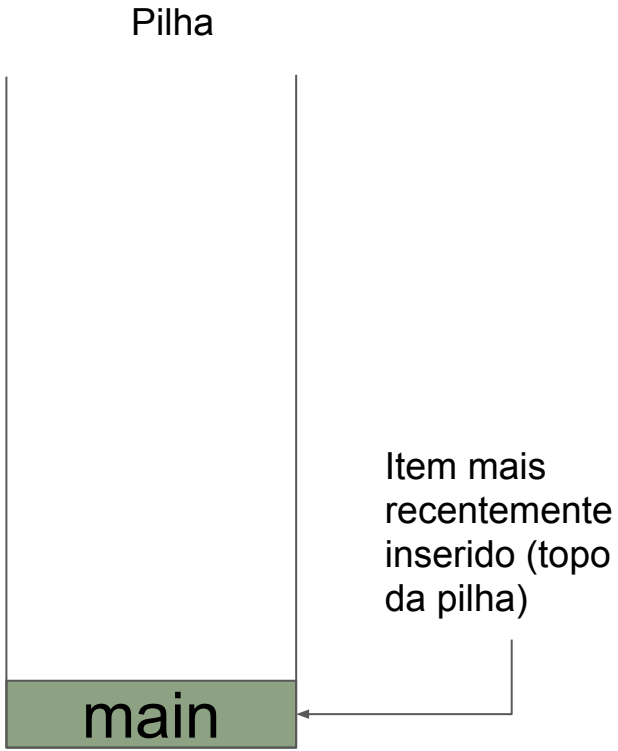
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int soma, dif, meleca;
   soma = soma(3,2);
   dif = dif(3,2);
   meleca = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

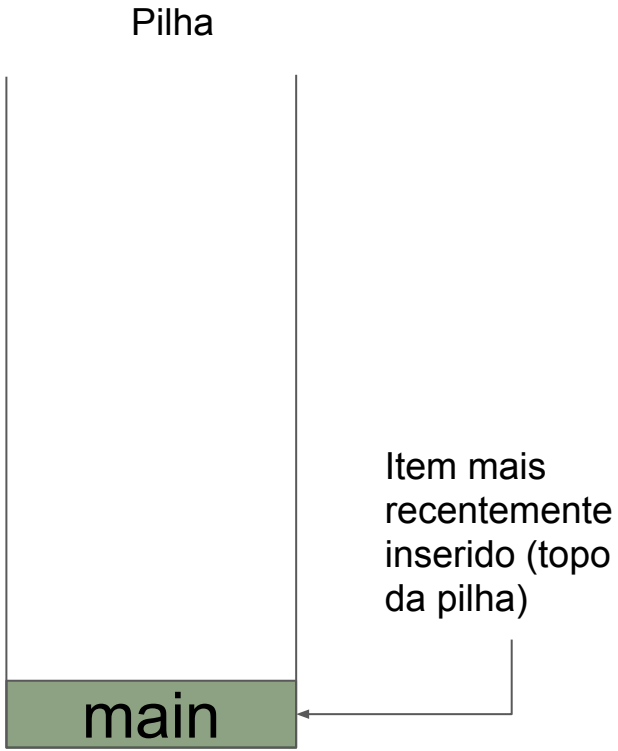
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

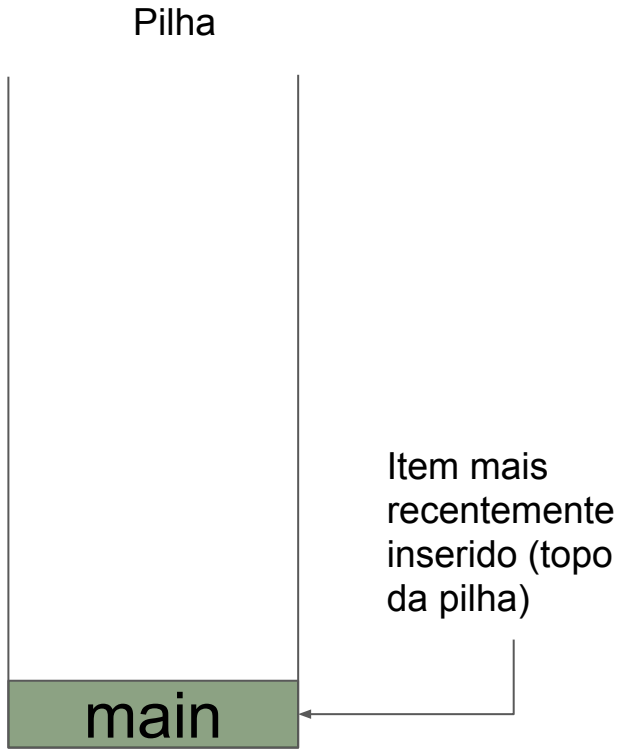
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

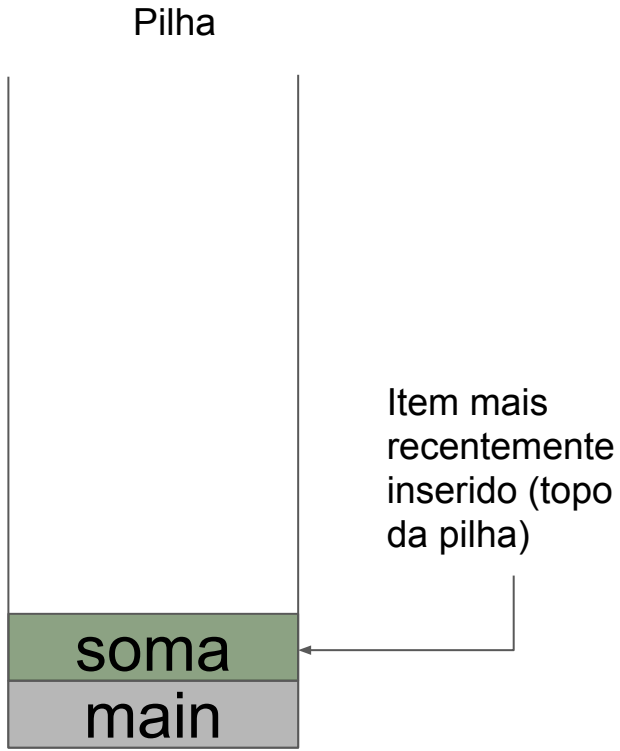
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

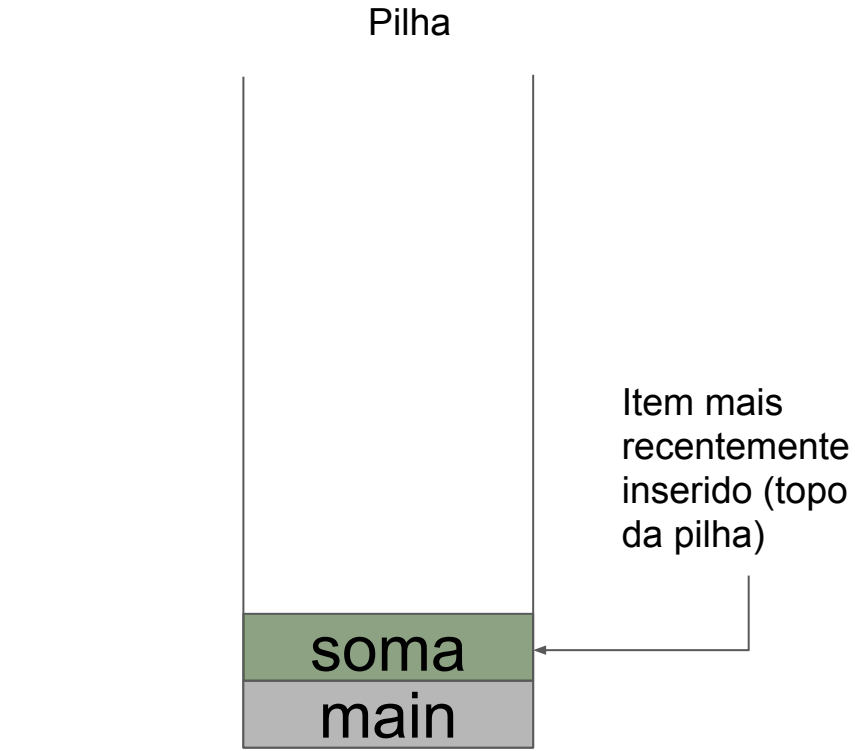
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

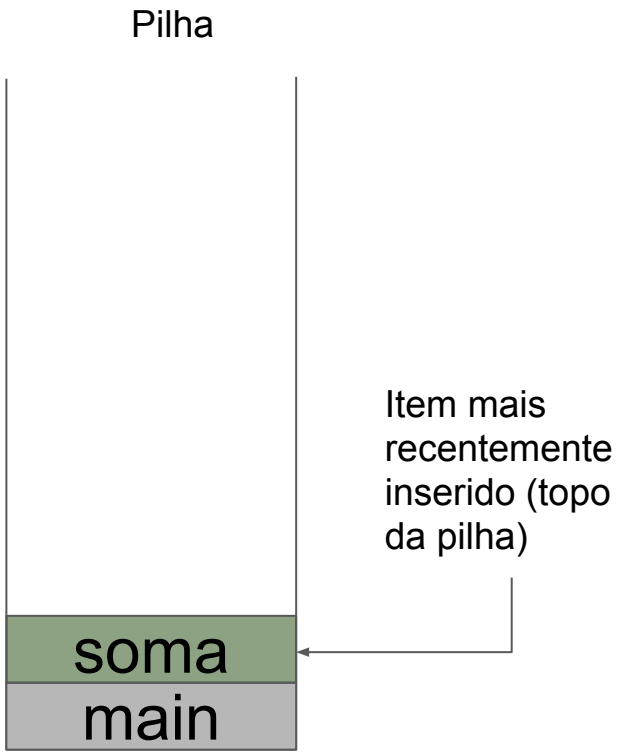
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

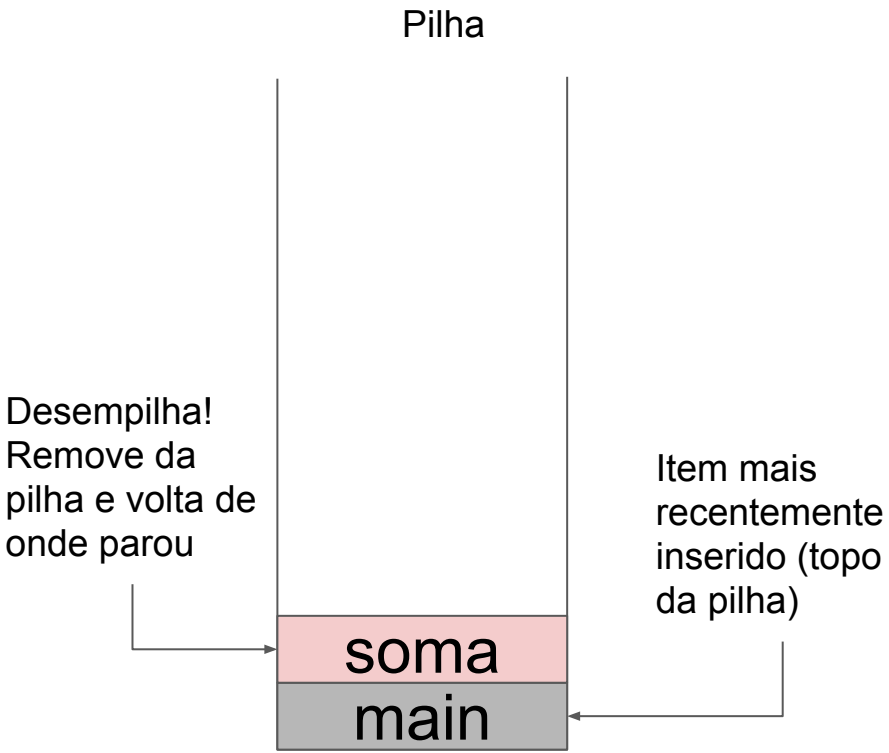
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

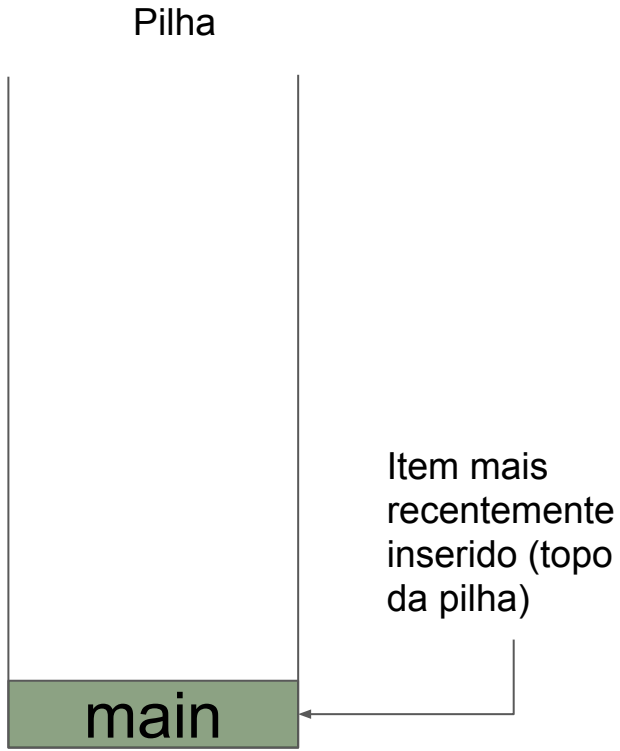
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

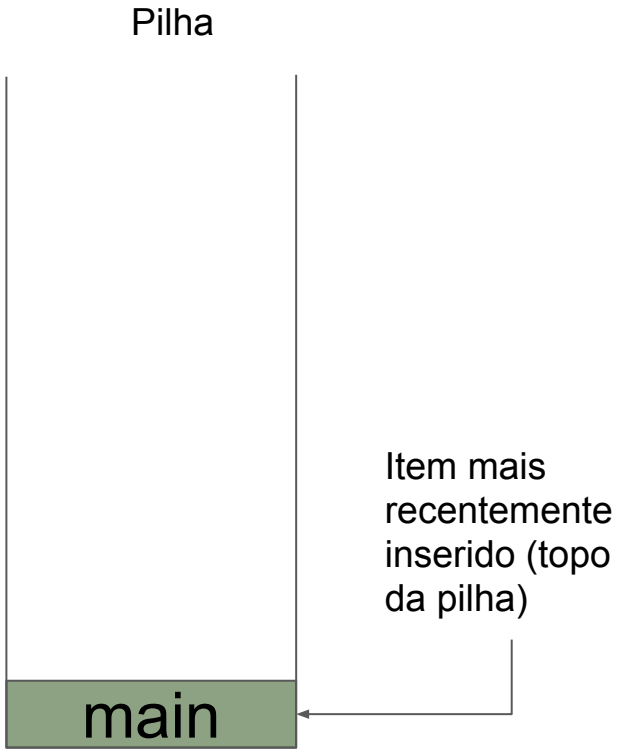
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

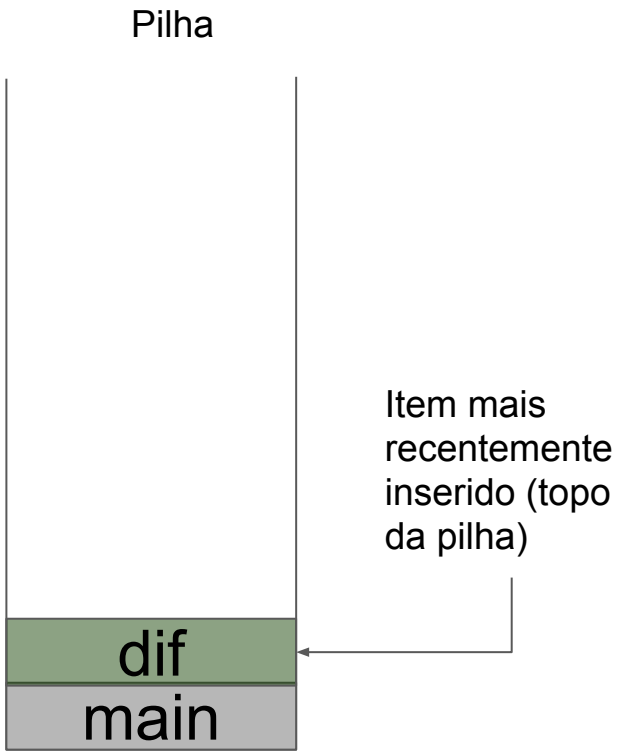
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

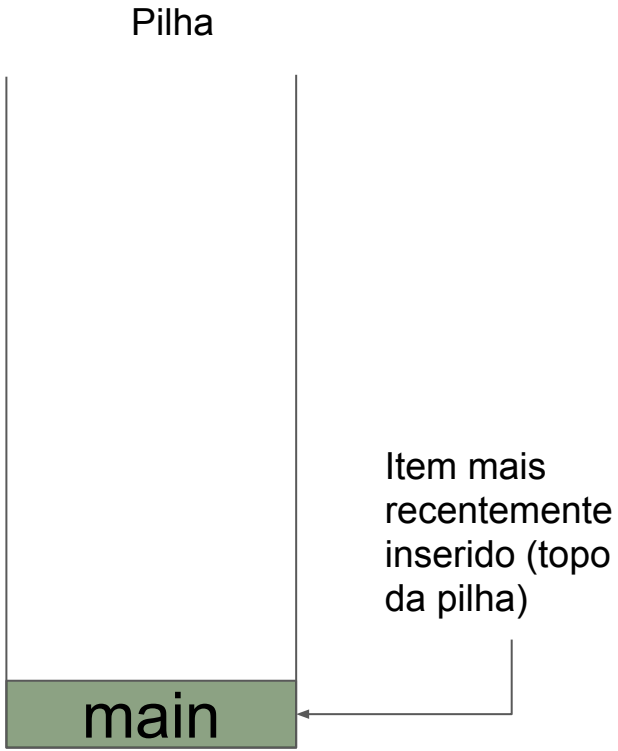
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

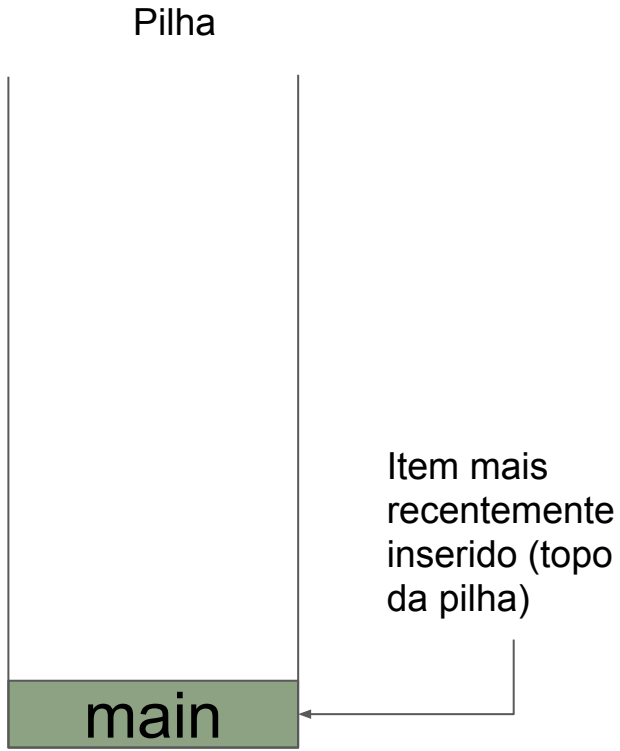
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```

Recursividade

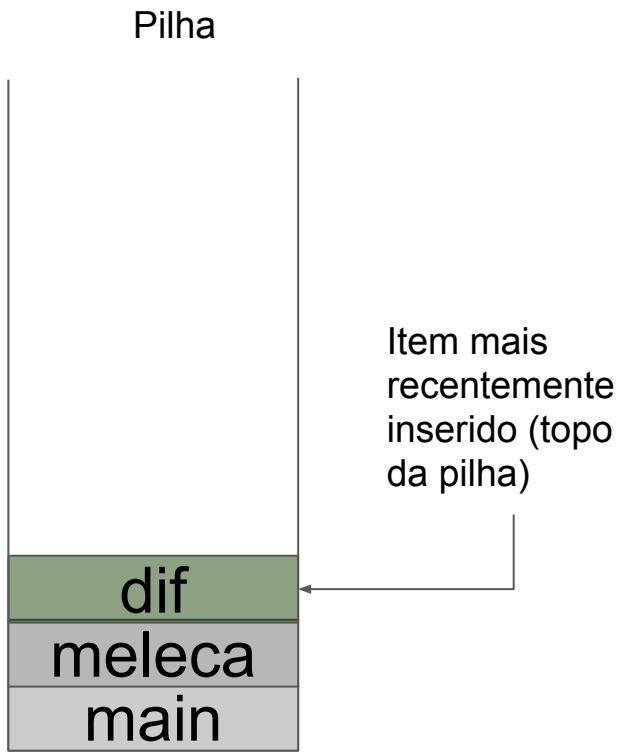
- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```

Recursividade

- Exemplo

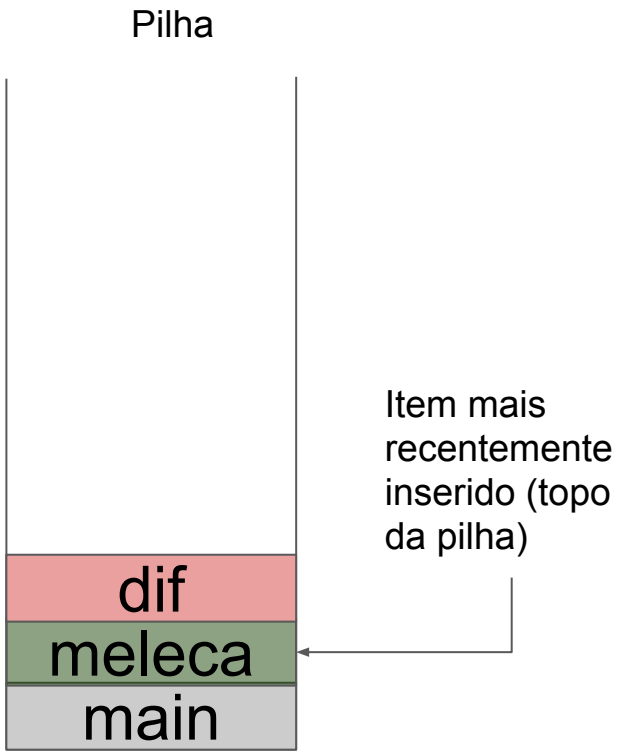
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

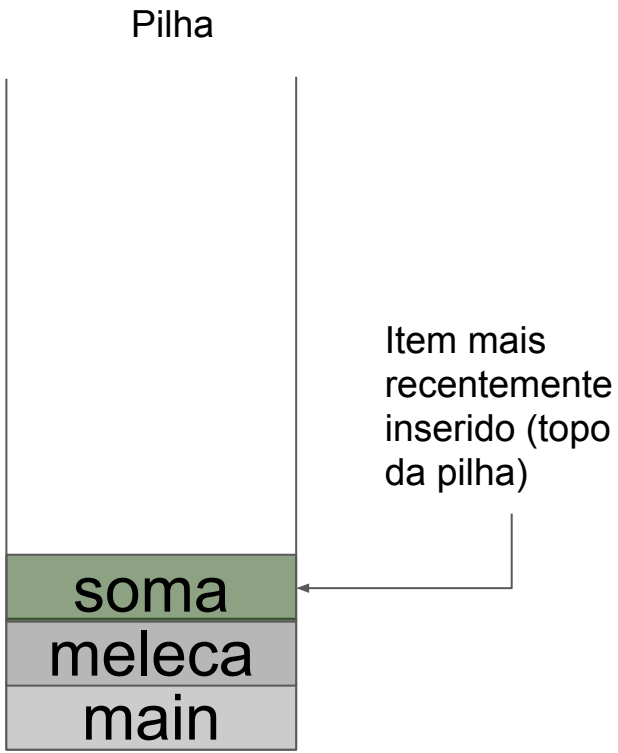
- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```

Recursividade

- Exemplo

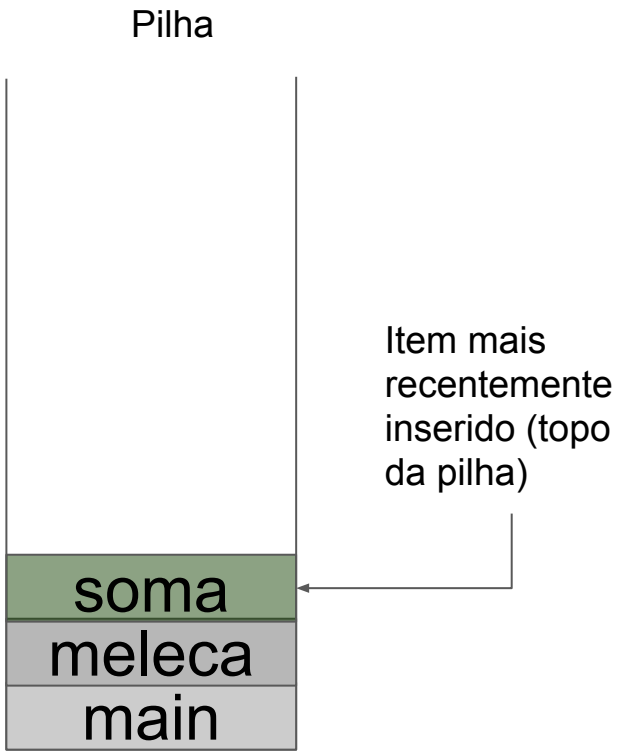
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

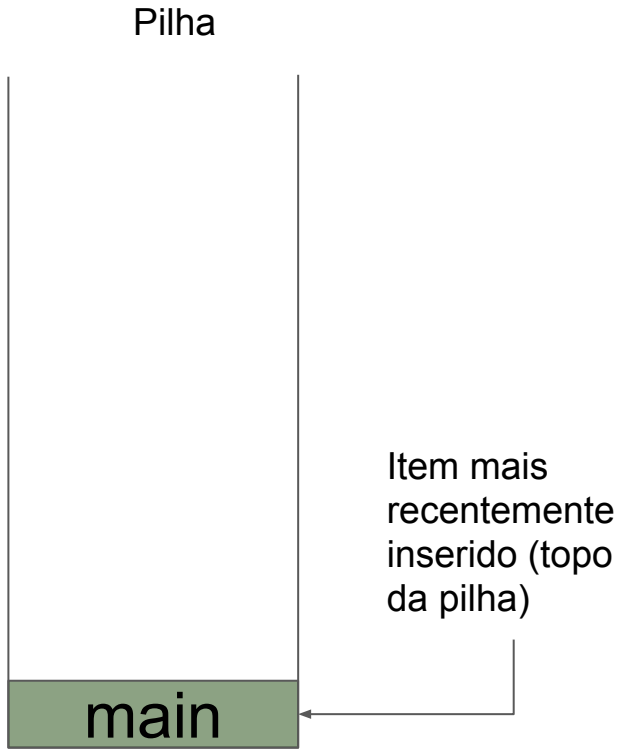
- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```

Recursividade

- Exemplo

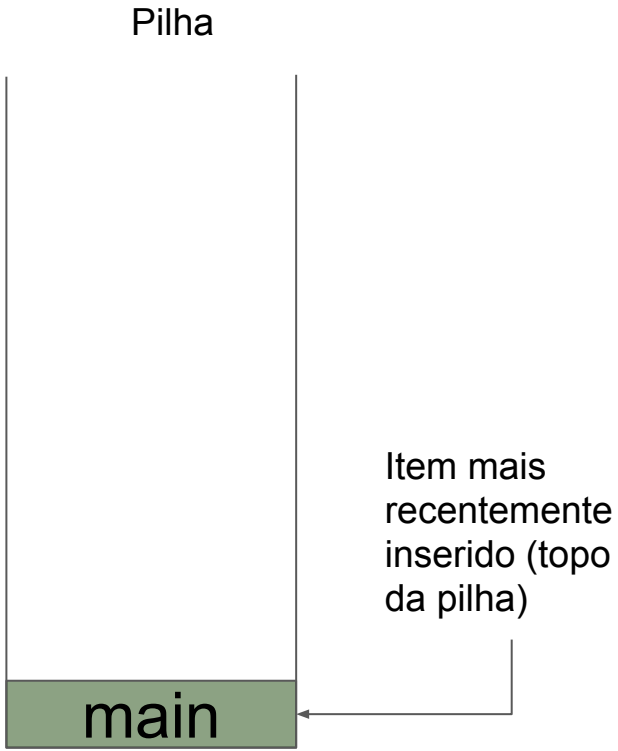
```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```



Recursividade

- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```

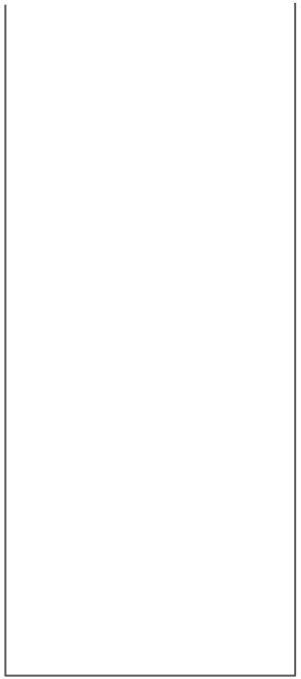


Recursividade

- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```

Pilha



Recursividade

- Exemplo

```c
#include <stdio.h>
int soma(int a, int b){
   int r = a+b;
   return r;
}
int dif(int a, int b){
   return a-b;
}
int meleca (int a, int b, int c){
   return soma(a, dif(b,c));
}
int main(){
   int s, d, m;
   s = soma(3,2);
   d = dif(3,2);
   m = meleca(3,1,5);
   return 0;
}
```

Recursividade

- Como usar isso para nosso benefício?

Recursividade

- Como usar isso para nosso benefício?
  - Quebramos o problema em partes menores, deixamos ele mais simples, e chamamos a função várias vezes até encontrar a forma mais simples

Recursividade

- Como usar isso para nosso benefício?
  - Quebramos o problema em
    ele até que esteja na sua

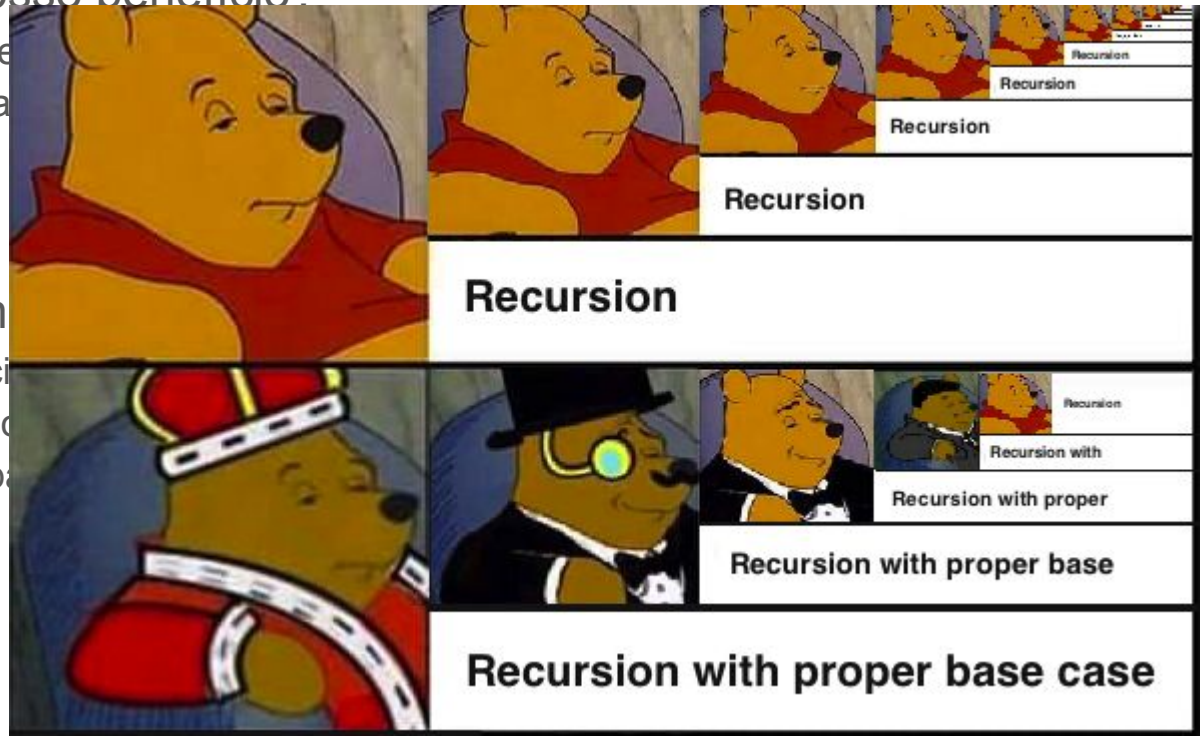
Recursividade

- Como usar isso para nosso benefício?
  - Quebramos o problema em partes menores, deixamos ele mais simples, e vamos reduzindo ele até que esteja na sua forma mais simples

- Podemos decompor uma recursão por
  - **Caso base**: uma instância do problema solucionada facilmente
  - **Chamadas recursivas**: onde a função é definida em termos de si própria, realizando uma redução para seu caso básico

Recursividade

- Como usar isso para nosso benefício?
  - Quebramos o problema e
    ele até que esteja na sua
- Podemos decompor um
  - **Caso base**: uma instânci
  - **Chamadas recursivas**: c
    redução para seu caso b



Recursividade

- Exemplo - Imprimir os números inteiros até X

```c
#include <stdio.h>
void imprimeItem(int atual, int X){
   if (atual >= X)
       return;
   printf(' %d ', atual);
   imprimeItem(atual+1, X);
}

int main(){
   imprimeItem(0,20);
   return 0;
}
```

Recursividade

- Exemplo - Fatorial
  - !n = n*(n-1)*(n-2)*...*1;

```c
#include <stdio.h>

int fatorial(int n){
   if (n > 0)
       return 1;
   return n*fatorial(n-1);
}

int main(){
   printf("%d", fatorial(10));
   return 0;
}
```

Recursividade

- Exemplo - Printar uma lista encadeada

```
void imprimeLista(Funcionario *primeiro)
{
   if (primeiro == NULL)
       return;

   printf("REGISTRO  ID: %d \n nome:%s\n nascimento: %d/%d/%d\n salario:%lf\n",
              primeiro->id, primeiro->nome, primeiro->nascimento.dia, primeiro->nascimento.mes,
primeiro->nascimento.ano, primeiro->salario);

   imprimeLista(primeiro->proximo);
}
```

Recursividade

- Exemplo - Printar uma lista encadeada

```
void imprimeLista(Funcionario *primeiro)
{
   if (primeiro == NULL)
       return;

   printf("REGISTRO  ID: %d \n nome:%s\n nascimento: %d/%d/%d\n salario:%lf\n",
              primeiro->id, primeiro->nome, primeiro->nascimento.dia, primeiro->nascimento.mes, primeiro->nascimento.ano,
primeiro->salario);

   imprimeLista(primeiro->proximo);
}
```

```
void imprimeLista(Funcionario *primeiro)
{
   Funcionario *aux = primeiro;

   while(aux!= NULL){
       printf("REGISTRO  ID: %d \n nome:%s\n nascimento: %d/%d/%d\n salario:%lf\n",
              primeiro->id, primeiro->nome, primeiro->nascimento.dia, primeiro->nascimento.mes, primeiro->nascimento.ano,
primeiro->salario);
    }
}
```

Recursividade

- Exemplo - Printar uma lista encadeada

```
void imprimeLista(Funcionario *primeiro)
{
   if (primeiro == NULL)
       return;

   printf("REGISTRO  ID: %d \n nome:%s\n
              primeiro->id, primeiro->
primeiro->salario);

   imprimeLista(primeiro->proximo);
}
```

Algoritmos recursivos tendem a ser mais lentos que suas versões iterativas:

- Por utilizarem mais memória
- Por otimizações do processador

```
void imprimeLista(Funcionario *primeiro)
{
   Funcionario *aux = primeiro;

   while(aux!= NULL){
       printf("REGISTRO  ID: %d \n nome:%s\n nascimento: %d/%d/%d\n salario:%lf\n",
              primeiro->id, primeiro->nome, primeiro->nascimento.dia, primeiro->nascimento.mes, primeiro->nascimento.ano,
primeiro->salario);
   }
}
```

Exercícios (USANDO RECURSIVIDADE)

1) Faça um programa que calcule o somatório de todos os N primeiros números
   - Soma = n+(n-1)+(n-2)+...+1
2) Desenvolva um algoritmo que calcule o n-ésimo termo de uma série de fibonacci.
   - Série de fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 . . .
   - O n-ésimo termo é obtido a partir dos dois anteriores.
3) Desenvolva um programa que imprima os elementos de uma lista simplesmente encadeada (com a inserção sempre baseada no último elemento) na ordem inversa a de inserção.
   - Lista inserida: 1,2,3,4,5 deve imprimir 5,4,3,2,1

Cronograma das próximas aulas

- 17/12/2020 - Árvores
- **Data a ser definida - Tarefa Prática (SuSy)**
- 23/12/2020 - Árvores
- 20/01/2021 - Exercícios
- **21/01/2021 - Avaliação 2**
- 27/01/2021 - Exercícios
- **28/01/2021 - Avaliação de Recuperação 2**