Sistemas para Internet
UFSM

Recursividade

Prof. Leandro O. Freitas
leandro@politecnico.ufsm.br

Retomando...

Sistemas para Internet
UFSM

- Pilhas:
  - Conceito de LIFO: último a entrar na pilha deve ser o primeiro a ser manipulado.

- Filas:
  - Conceito de FIFO: primeiro elemento a entrar na fila deve ser o primeiro a ser manipulado.

Pilha de funções

Sistemas para Internet
UFSM

- Conceito utilizado quando temos uma função que chama outra função, que por sua vez chama uma terceira função e assim sucessivamente;

- A última função a ser chamada, será a primeira a terminar sua execução *(LIFO).*

Pilha de funções

Sistemas para Internet
UFSM

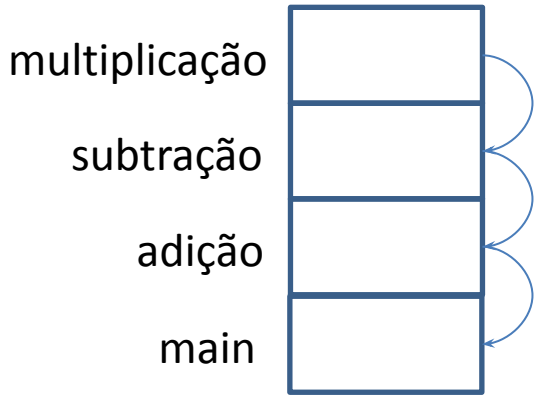
```
void multiplicacao (int x, int y) {
     int mult;
     mult = x * y;
     printf("%d \n", mult);
     }

void subtracao (int x, int y) {
     int sub;
     multiplicacao(x, y);
     sub = x - y;
     printf("%d \n", sub);
     }

void adicao (int x, int y) {
    int soma;
    subtracao(x, y);
    soma = x + y;
    printf("%d \n", soma);
    }

 int main(){
      int x = 10, y = 5;
      adicao(x, y);
      return 0;
      }
```

```
50
5
15
```



Função recursiva

Sistemas para Internet
UFSM

- Abordagem para resolver problemas na qual a solução depende de soluções para instâncias menores do mesmo problema.
  - Princípio de “dividir para conquistar”, mas onde a “divisão” resulta em um problema que é apenas uma versão menor do original;
  - Caracteriza-se por ser uma tipo de função que faz chamadas a si própria.

- Vantagens:
  - Reduz o tamanho do código;
  - Permite descrever algoritmos de forma mais clara e concisa (dependendo do nível de profundidade).

- Desvantagens:
  - Reduz desempenho da execução devido a gerenciamento das chamadas;
  - Depuração de erros pode se tornar mais complexa se a recursividade for muito profunda.

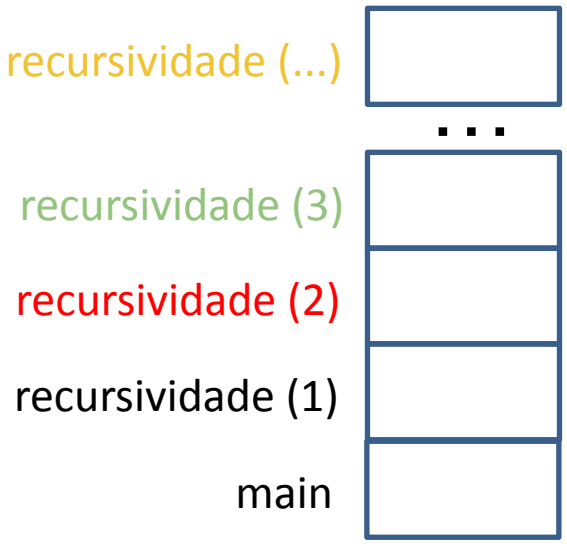
- Utiliza-se uma pilha para armazenar os dados usados em cada chamada de uma função que não terminou.

Exemplo de função recursiva

Sistemas para Internet
UFSM

```
1  #include <stdio.h>
2
3  void recursividade(int a) {
4      a++;
5      printf("%d \n", a);
6      recursividade(a);
7
8
9  }
10
11
12 int main()
13 {
14     recursividade(5);
15
16     return 0;
17 }
```

Qual é o problema desse exemplo?



Funções recursivas

Sistemas para Internet
UFSM

- **Atenção!**
  - A recursividade pode não ter fim (chamadas à função “infinitamente”);
  - Isto deve ser tratado durante o desenvolvimento do programa.

- Para evitar isso, sempre deve haver uma condição na função que, quando for aceita, encerra a recursão;

- O momento que encerramos a recursão, é chamado de **CASO BASE**.

- Caso base:
  - Permite que a função deixe de ser executada;
  - A função deve ter pelo menos um caso base para cada algoritmo recursivo, o que significa a finalização da função.

Funções recursivas

Sistemas para Internet
UFSM

- Ao desenvolver um algoritmo recursivo, deve-se:
  - Definir pelo menos um **caso base** (condição de terminação);
  - Fazer o **teste de finitude,** isto é, certificar-se de que as chamadas recursivas levam obrigatoriamente, em algum momento, ao(s) caso(s) base(s).

```c
#include <stdio.h>

int Exemplo(int a){
    if (a > 3)
        return a;

    else {
        a++;
        a = Exemplo(a);
    }
    return a;
}

int main(){
    int x = 1, resultado;
    resultado = Exemplo(x);
    printf("%d", resultado);
    return 0;
}
```

```c
int NaoRec (int a)
{
    while (a < 4)
        a++;
    return a;
}
```

Toda função recursiva bem definida pode ser escrita de forma não recursiva.

Exercício

Sistemas para Internet
UFSM

- Criar uma função recursiva para calcular o somatório dos números positivos até o valor N recebido por parâmetro. A função deve retornar o resultado ao programa principal, onde será impresso.

Exercício

- Solução:
- Faça a depuração do algoritmo;
- Altere o algoritmo para mostrar o somatório apenas dos valores pares.

```c
#include<stdio.h>

int soma(int x)
{
    if (x == 1)
    {
        return 1;
    }
    else
    {
        x = x + soma(x - 1);
        return x;
    }
}

int main(){
{
    int n;
    scanf("%d", &n);
    n = soma(n);
    printf("\n%d\n", n);
    return 0;
}
```