SCC0222 - Laboratório de Introdução à Ciência de Computação I

Recursão

Prof.: Leonardo Tórtoro Pereira

leonardop@usp.br

Recursão [1]

- É quando uma função chama a ela mesma
  - Essa chamada pode ser direta ou indireta
- Pode resolver com facilidade alguns problemas
  - Mas, para outros, um laço de iteração é melhor
- Normalmente é dividida em um caso base, que retorna um valor para a solução, e o caso maior, que é expressado em função de problemas menores

Recursão [1]

```c
void recursiveFunction(int input)
{
    printf("%d\n", input);
    recursiveFunction(--input);
}

int main()
{
    recursiveFunction(5);
    return 0;
}
```

Recursão [1]

➔ Quando o caso base não é alcançado, pode retornar *stack overflow*

Recursão [1]

```c
void recursiveFunction(int input){
    if(input > 0)
    {
        printf("%d\n", input);
        recursiveFunction(--input);
    }
}

int main(){
    recursiveFunction(5);
    return 0;
}
```

Recursão [1]

- O método de recursão costuma ser usado quando a solução depende da solução de instâncias menores do mesmo problema

Recursão [1]

- Alguns problemas bem conhecidos que podemos usar recursão:
  - Fatorial
  - Sequência de Fibonacci (e outras sequências no geral)
  - Busca binária
  - Maior divisor comum
  - Torre de Hanoi
  - ...

Fatorial

Recursão [1]

```c
int fact(int n){
    int answer;
    if(n <= 1)
        return 1;
    else{
        answer = n*fact(n-1);
        return answer;
    }
}

int main(){
    printf("%d\n",fact(5));
}
```

Fibonacci

Recursão [1]

```c
int fib(int n){
    if (n == 0)
        return 0;
    if (n == 1 || n == 2)
        return 1;
    else
        return (fib(n - 1) + fib(n - 2));
}
int main(){
    for (int i = 0; i < NUMBER; i++)
        printf("%d ", fib(i));
    return 0;
}
```

Maior Divisor Comum

Recursão [1]

```c
int mdc(int x, int y){
  if (y == 0)
     return x;
  else
     return mdc(y, x % y);
}
int main(){
    printf("MDC: %d\n", mdc(4, 6));
}
```

Busca Binária

Recursão [1]

```c
#define NOT_FOUND -1
int binarySearch(int begin, int end, int key, int vector[]){
    int middle = (begin + end) / 2;
    if (begin > end)
        return NOT_FOUND;
    if (key == vector[middle])
        return middle;
    else if (key < vector[middle])
        return binarySearch(begin, middle-1, key, vector);
    else if (key > vector[middle])
        return binarySearch(middle+1, end, key, vector);
    return NOT_FOUND;
}
int main(){
    int vetor[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    printf("Index: %d\n", binarySearch(0, 9, 10, vetor));
    return 0;
}
```

Referências

1. https://www.geeksforgeeks.org/recursion/
2. https://pt.wikipedia.org/wiki/Recursividade_(ci%C3%AAncia_da_computa%C3%A7%C3%A3o)
3. https://en.wikipedia.org/wiki/Recursion_(computer_science)
4. Ziviani, N. Projeto de Algoritmos. Thomson, 2ª Edição, 2004.