Programando com recursão
Professor Wladimir A. Tavares

Recursão

Até agora, aprendemos a resolver problemas utilizando estruturas de repetição, como os laços for e while. No entanto, existe outra técnica poderosa que podemos usar para resolver problemas: a recursão.

A recursão consiste em abordar um problema complexo dividindo-o em um ou mais subproblemas menores do mesmo tipo. Esse processo continua até que os subproblemas sejam simples o suficiente para serem resolvidos diretamente.

Escrita dos números de 1 até n

Implemente uma função que escreve todos os números de 1 até n:

```
int escreve(int n){
    for(int i = 1; i <= n; i++)
        printf("%d\n", i);
}
```

Note que o problema da escrita de 1 até n pode ser decomposto no problema da escrita de 1 até n-1 seguido da escrita de n.

```
int escreve(int n){
    if(n==1)
        printf("1\n");
    else{
        escreve(n-1);
        printf("%d\n", n);
    }
}
```

Escrita dos números de n até 1

```
int escreve(int n){
    if(n==1)
        printf("1\n");
    else{
        printf("%d\n", n);
        escreve(n-1);
    }
}
```

Somatório de 1 até n

```
int somatorio(int n){
    if(n==1)
        return 1;
    else{
        return somatorio(n-1) + n;
    }
}
```

Fatorial de 1 até n

```
int fatorial(int n){
    if(n==1)
        return 1;
    else{
        return fatorial(n-1) * n;
    }
}
```

Recursão com padrões

Triangulo Refletido

```
****
***
**
*
*
**
***
****
```

```
void triangulo(int n){
    if(n==1){
        printf("*\n");
    }else{
        for(int i = 1; i <= n; i++){
            printf("*");
        }
        printf("\n");
        triangulo(n-1);
        for(int i = 1; i <= n; i++){
            printf("*");
        }
        printf("\n");
    }
}
```

Triangulo2

```
n = 4
* k = 1
** k = 2
*** k = 3
**** k = 4
*** k = 3
** k = 2
* k = 1
```

```
void __triangulo2(int n, int k){
    if(n == k){
        for(int i = 1; i <= n; i++)
            printf("*");
        printf("\n");
    }else{
        for(int i = 1; i <= k; i++)
            printf("*");
        printf("\n");
        __triangulo2(n, k+1);
        for(int i = 1; i <= k; i++)
            printf("*");
        printf("\n");
    }
}

void triangulo2(int n){
    __triangulo2(n, 1);
}
```

Triangulo 3

```
**** k = 0, n = 4
 *** k = 1, n = 3
  ** k = 2, n = 2
   * k = 3, n = 1
  ** k = 2, n = 2
 *** k = 1, n = 3
**** k = 0, n = 4
```

```c
void __triangulo3(int n, int k){
    if(n==1){
        for(int i = 1; i <= k; i++) printf(" ");
        printf("*\n");
    }else{
        for(int i = 1; i <= k; i++) printf(" ");
        for(int i = 1; i <= n; i++) printf("*");
        printf("\n");
        __triangulo3(n-1, k+1);
        for(int i = 1; i <= k; i++) printf(" ");
        for(int i = 1; i <= n; i++) printf("*");
        printf("\n");
    }
}
void triangulo3(int n){
    __triangulo3(n, 0);
}
```

Triangulo 4

```
*******
 *****
  ***
   *
  ***
 *****
*******
```

```
void __triangulo4(int n, int k){
    if(n==1){
        for(int i = 1; i <= k; i++) printf(" ");
        printf("*\n");
    }else{
        for(int i = 1; i <= k; i++) printf(" ");
        for(int i = 1; i <= 2*n-1; i++) printf("*");
        printf("\n");
        __triangulo4(n-1, k+1);
        for(int i = 1; i <= k; i++) printf(" ");
        for(int i = 1; i <= 2*n-1; i++) printf("*");
        printf("\n");


    }
}

void triangulo4(int n){
    __triangulo4(n, 0);
}
```

Losango

```
N = 3
  *         asteriscos 2k-1
 ***        espacos n-k
*****
*****
 ***
  *
```

```
void losango(int n, int k) {
    if(k>n){
    return ;
    }else{
        for(int i = 1; i <= n-k; i++)
            printf(" ");
        for(int i = 1; i <= 2*k-1; i++)
            printf("*");
        printf("\n");
        losango(n, k+1);
        for(int i = 1; i <= n-k; i++)
            printf(" ");
        for(int i = 1; i <= 2*k-1; i++)
            printf("*");
        printf("\n");
    }
}
```

DesenhaX=5

```
N = 5
*       *    K = 5 espacoAntes = 0 espacoMeio = 7
 *     *     K = 4 espacoAntes = 1 espacoMeio = 5
  *   *      K = 3 espacoAntes = 2 espacoMeio = 3
   * *       K = 2 espacoAntes = 3 espacoMeio = 1
    *        K = 1 espacoAntes = 4
   * *       K = 2 espacoAntes = 3 espacoMeio = 1
  *   *      K = 3 espacoAntes = 2 espacoMeio = 3
 *     *     N = 4 espacoAntes = 1 espacoMeio = 5
*       *    N = 5 espacoAntes = 0 espacoMeio = 7

espacoMeio = 2*k-3
```

```
void desenhaAux(int K, int espacoAntes, int espacoMeio);


void desenhaX(int N){
    desenhaAux(N, 0, 2*N-3);
}

void desenhaAux(int K, int espacoAntes, int espacoMeio){

    if(K == 1){
        for(int i = 1; i <= espacoAntes; i++)
            printf(" ");
        printf("*\n");
    }else{

        for(int i = 1; i <= espacoAntes; i++)
            printf(" ");
        printf("*");
        for(int i = 1; i <= espacoMeio; i++)
            printf(" ");
        printf("*\n");

        desenhaAux(K-1, espacoAntes+1, espacoMeio-2);

        for(int i = 1; i <= espacoAntes; i++)
            printf(" ");
        printf("*");
        for(int i = 1; i <= espacoMeio; i++)
            printf(" ");
        printf("*\n");
    }
}
```

Recursão com Vetores

Encontrar o maior

```
int maior(int v[], int n){
    if(n==1){
        return v[0];
    }else{
        int m = maior(v, n-1);
        return m > v[n-1] ? m : v[n-1];
    }
}
```

Encontrar o maior e o segundo maior

```
typedef struct pair{
    int x, y;
} pair;

pair segundo_maior(int v[], int n){
    pair p;
    if(n==2){
        if(v[0]>= v[1]){
            p.x = v[0];
            p.y = v[1];
        }else{
            p.x = v[1];
            p.y = v[0];
        }
        return p;
    }else{
        pair p = segundo_maior(v, n-1);

        if(v[n-1] >= p.x){
            p.x = v[n-1];
        }else{
            if(v[n-1] >= p.y){
                p.y = v[n-1];
            }
        }

        return p;

    }
}
```

Busca

```
int busca(int vet[], int n, int k){
    if(n==1){
        return (vet[0] == k);
    }else{
        if( busca(vet, n-1, k) == 1){
            return 1;
        }else{
            return vet[n-1] == k;
        }
    }
}
```

Ordena

```
void ordena(int vet[], int n){
    if(n>1){
        int pos = maior_pos(vet, n);
        //troca o maior com o ultimo
        int t = vet[n-1];
        vet[n-1] = vet[pos];
        vet[pos] = t;
        ordena(vet, n-1);
    }
}
```