

Subalgoritmos

Aula 20



Subalgoritmos

PROBLEMA: Escreva um algoritmo para exibir a seguinte tela.

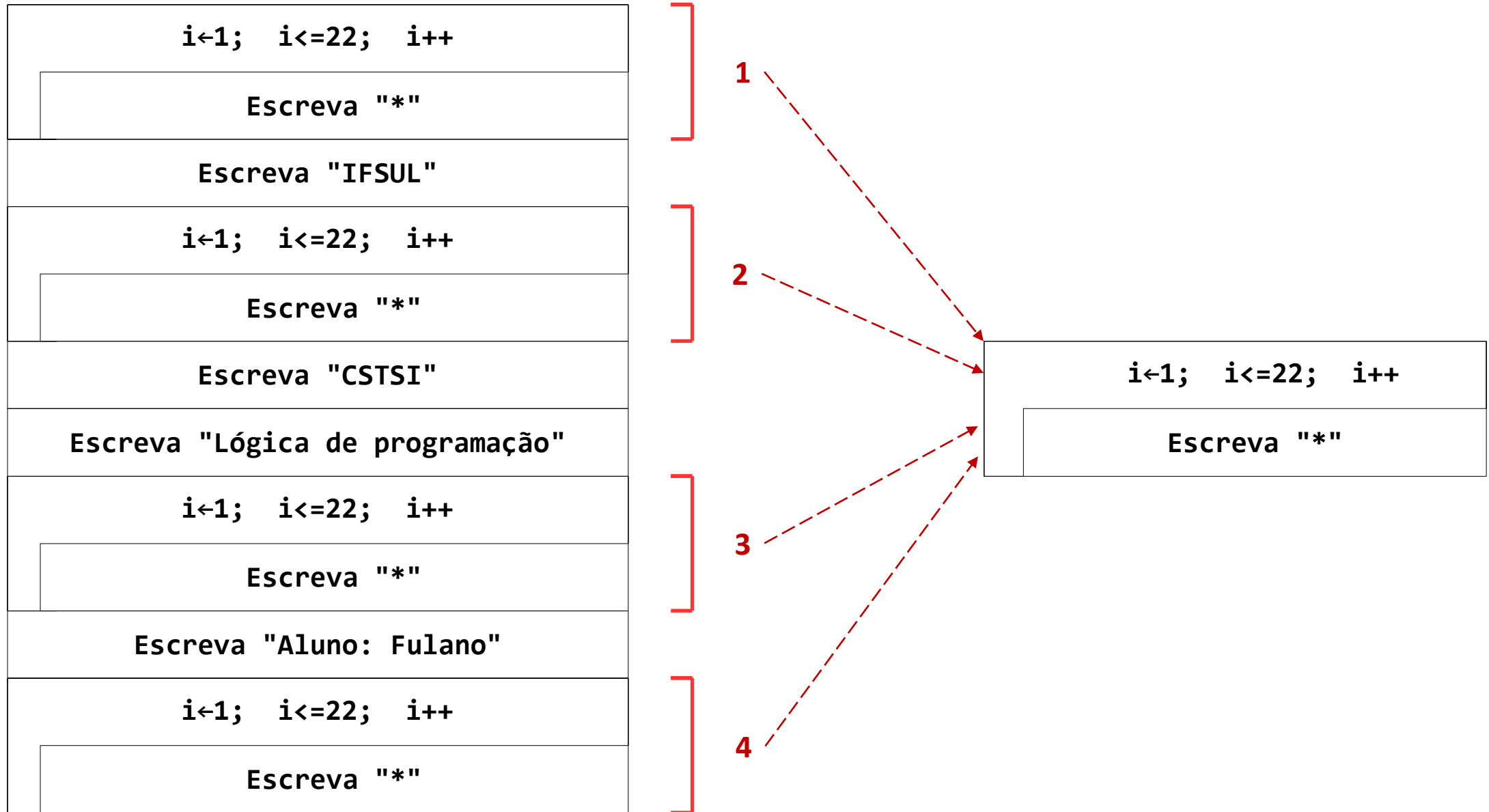
IFSUL

CSTSI

Lógica de programação

Aluno: Fulano

Cada linha que contém 22 asteriscos
deve ser impressa com repetição.

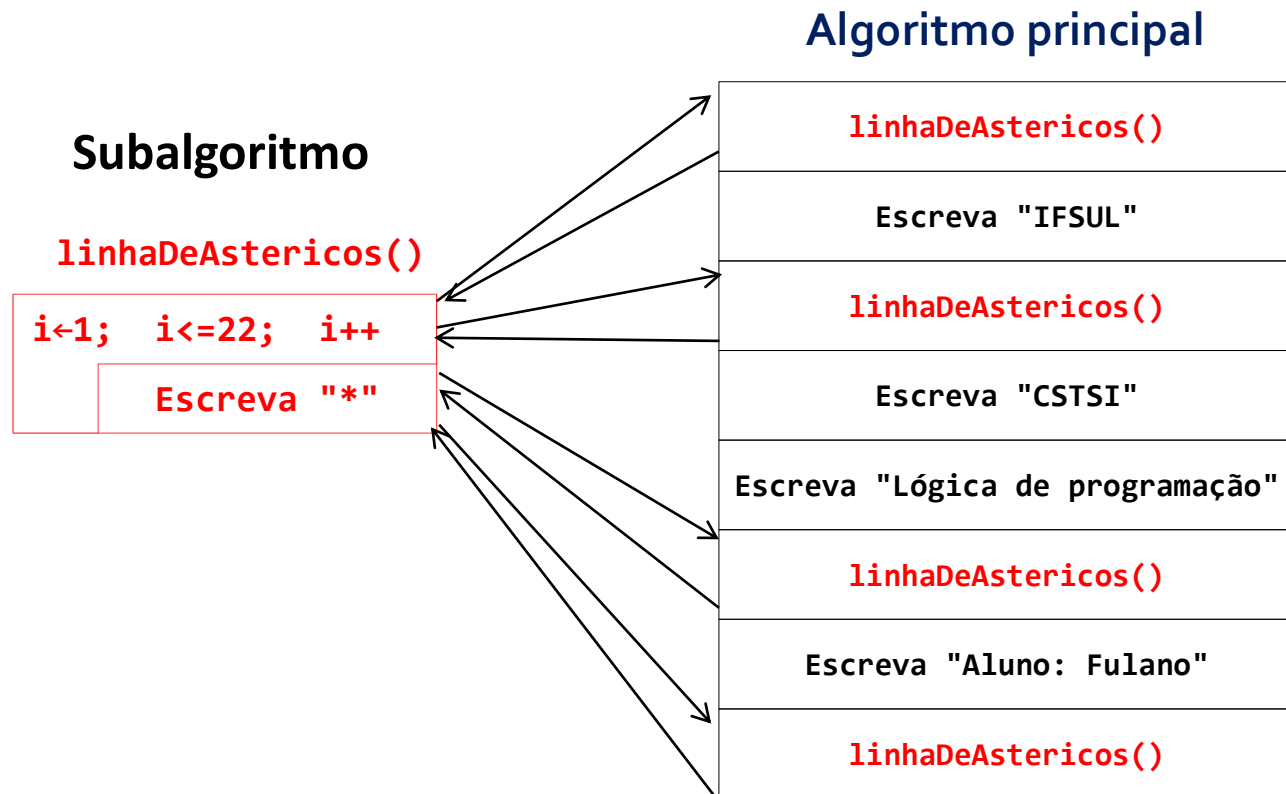




Subalgoritmos

É um trecho de algoritmo, com uma **função bem definida** (o mais **independente** possível do restante do algoritmo) que é **chamado dentro de um algoritmo principal**.

i←1; i≤22; i++
Escreva "*"
Escreva "IFSUL"
i←1; i≤22; i++
Escreva "*"
Escreva "CSTSI"
Escreva "Lógica de programação"
i←1; i≤22; i++
Escreva "*"
Escreva "Aluno: Fulano"
i←1; i≤22; i++
Escreva "*"





Vantagens dos Subalgoritmos

- ✓ **Evita** que um trecho de comandos necessário em vários locais tenha que ser **escrito repetidamente**;
- ✓ **Divide e estrutura** um algoritmo em partes logicamente coerentes;
- ✓ Aumenta a **legibilidade** do algoritmo;
- ✓ Facilita a **detecção de erros**;
- ✓ Permite a **reutilização de software**; e
- ✓ **Divide** um problema grande em vários menores.

```
#include <stdio.h>
```

```
void linhaDeAsteriscos(void);
```

← Protótipo da função

```
main()
```

← Programa principal
(função principal)

```
linhaDeAsteriscos();
```

```
printf("\nIFSUL\n");
```

```
linhaDeAsteriscos();
```

```
printf("\nCSTSI\n\n");
```

```
printf("Lógica de programação\n");
```

```
linhaDeAsteriscos();
```

```
printf("\nAluno: Fulano\n");
```

```
linhaDeAsteriscos();
```

```
}
```

Chamadas da função

Definição da função
(corpo)

Retorno da função

Nome da função

Parâmetros da função

Cabeçalho da função

```
void linhaDeAsteriscos(void){
```

```
int i;
```

Variável local

```
for (i=1; i<=22; i++)
```

```
    printf("*");
```

```
}
```

PROBLEMA: Escreva um programa para ler um inteiro Q e imprimir Q linhas de 22 asteriscos.



Variáveis locais:

- São visíveis apenas no local onde são declaradas.
- São criadas quando a execução da função inicia e destruídas quando termina.

```
#include <stdio.h>

void linhaDeAsteriscos(void);

main(){
    int i, q; ← Variáveis locais

    printf("Informe Q:");
    scanf("%d", &q);
    for (i=1; i<=q; i++){
        linhaDeAsteriscos();
        printf("\n");
    }
}

void linhaDeAsteriscos(void){
    int i; ← Variável local

    for (i=1; i<=22; i++)
        printf("*");
}
```



Variáveis locais

```
#include <stdio.h>
void exibeNumero(void);
main(){
    int num;
    num = 30;
    exibeNumero();
    printf("Número (main):%d\n", num);
}

void exibeNumero(void){
    num = 40;
    printf("Número (função):%d\n", num);
}
```



*Esse programa não compila.
Por que??*

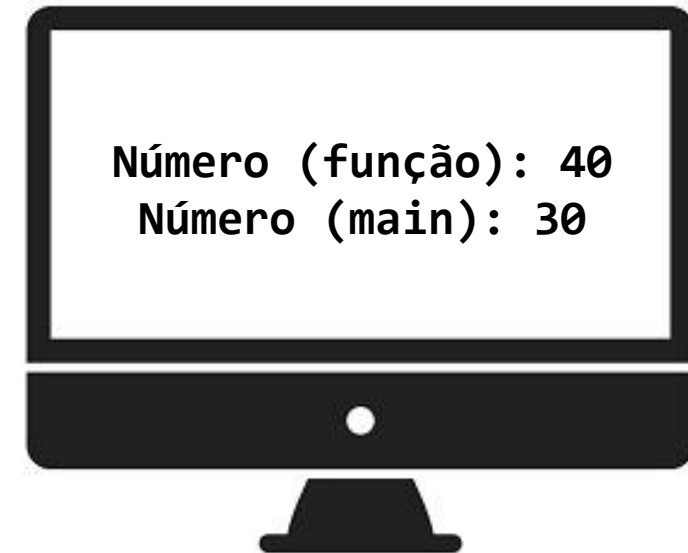
Variáveis locais estão isoladas dentro da função onde foram declaradas.



Variáveis locais

```
#include <stdio.h>
void exibeNumero(void);
main(){
    int num;
    num = 30;
    exibeNumero();
    printf("Número (main):%d\n", num);
}

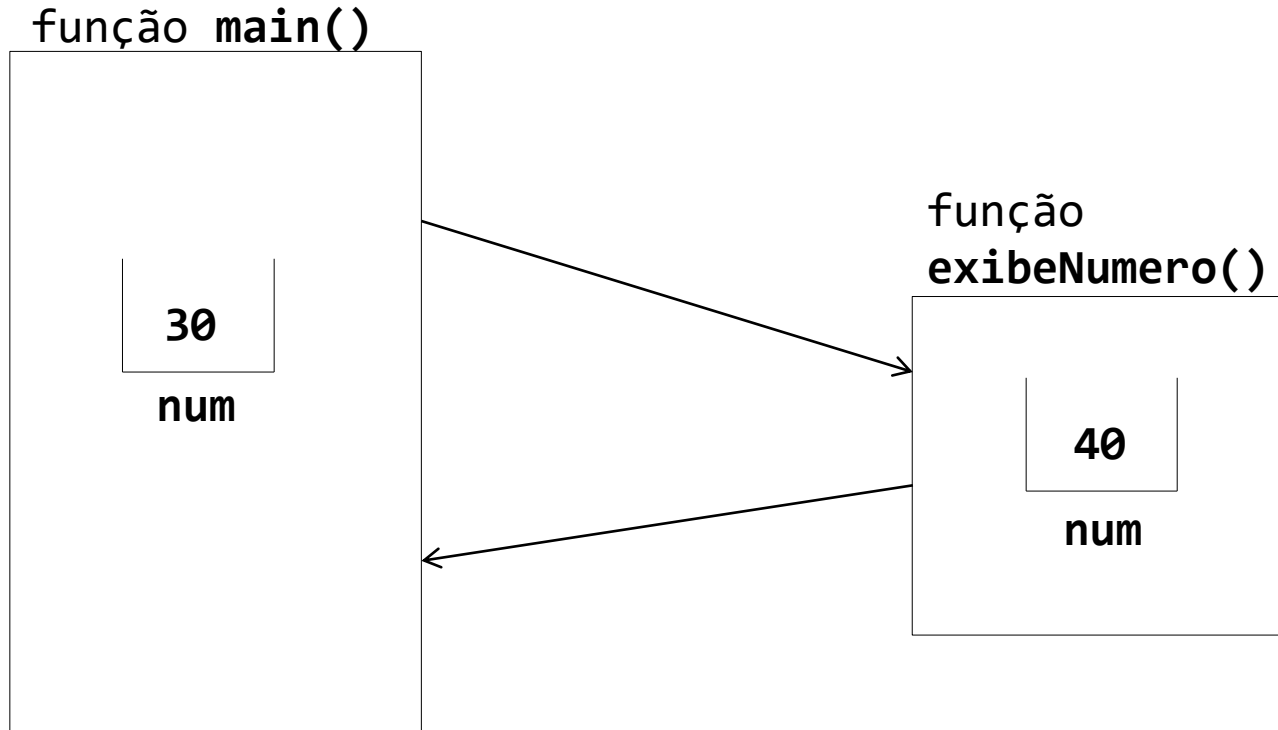
void exibeNumero(void){
    int num;
    num = 40;
    printf("Número (função):%d\n", num);
}
```





Variáveis locais

“São duas variáveis (num) diferentes com o **mesmo nome**”





Problema

Escreva um programa para exibir a tela abaixo.

O programa deve implementar uma função chamada `retAsteriscos()` que exibe na tela um retângulo com 3 linhas e 22 colunas de asteriscos.

```
*****
```

```
*****
```

```
*****
```

```
IFSUL
```

```
*****
```

```
*****
```

```
*****
```

```
#include <stdio.h>
void linhaDeAsteriscos(void);
void retAsteriscos(void);
```

```
main(){
    retAsteriscos();

    printf("\n    IFSUL\n");
    retAsteriscos();
}
```

```
void linhaDeAsteriscos(){
    int i;
    for (i=1; i<=22; i++){
        printf("*");
    }
}
```

```
void retAsteriscos(){
    int i;
    for (i=1; i<=3; i++){
        linhaDeAsteriscos();

        printf("\n");
    }
}
```

```
#include <stdio.h>
void linhaDeAsteriscos(void);
void retAsteriscos(void);

main(){
    retAsteriscos();

    printf("\n    IFSUL\n");
    retAsteriscos();
}

void linhaDeAsteriscos(){
    int i;
    for (i=1; i<=22; i++)
        printf("*");
}

void retAsteriscos(){
    int i;
    for (i=1; i<=3; i++){
        linhaDeAsteriscos();

        printf("\n");
    }
}
```

The diagram illustrates the flow of function calls in the provided C code. It features three function definitions: `main()`, `linhaDeAsteriscos()`, and `retAsteriscos()`. Arrows indicate the sequence of calls: `main()` calls `retAsteriscos()` (indicated by a horizontal arrow from the first `retAsteriscos()` call to the start of its definition), `main()` calls `linhaDeAsteriscos()` (indicated by a horizontal arrow from the second `retAsteriscos()` call to the start of its definition), and `retAsteriscos()` calls `linhaDeAsteriscos()` (indicated by a horizontal arrow from the `linhaDeAsteriscos()` call inside `retAsteriscos()` to the start of its definition).

```
#include <stdio.h>
void linhaDeAsteriscos(void);
void retAsteriscos(void);

main(){
    retAsteriscos();

    printf("\n    IFSUL\n");
    retAsteriscos();
}

void linhaDeAsteriscos(){
    int i;
    for (i=1; i<=22; i++)
        printf("*");
}

void retAsteriscos(){
    int i;
    for (i=1; i<=3; i++){
        linhaDeAsteriscos();
        printf("\n");
    }
}
```

The diagram illustrates the flow of function calls in the provided C code. Arrows indicate the sequence of execution:

- An arrow from `retAsteriscos();` in `main()` points to the `retAsteriscos()` function definition.
- An arrow from `retAsteriscos()` points to the `linhaDeAsteriscos()` function definition.
- An arrow from `linhaDeAsteriscos()` points back to the `main()` function, specifically to the line following the first `retAsteriscos();` call.

```
#include <stdio.h>
void linhaDeAsteriscos(void);
void retAsteriscos(void);
```

```
main(){
    retAsteriscos();

    printf("\n    IFSUL\n");
    retAsteriscos();
}
```

```
void linhaDeAsteriscos(){
    int i;
    for (i=1; i<=22; i++)
        printf("*");
}
```

```
void retAsteriscos(){
    int i;
    for (i=1; i<=3; i++){
        linhaDeAsteriscos();

        printf("\n");
    }
}
```

```
graph TD
    main["main()"] --> retAsteriscos["retAsteriscos()"]
    main --> linhaDeAsteriscos["linhaDeAsteriscos()"]
    linhaDeAsteriscos --> retAsteriscos
    retAsteriscos --> main
```

```
#include <stdio.h>
void linhaDeAsteriscos(void);
void retAsteriscos(void);
```

```
main(){
    retAsteriscos();

    printf("\n    IFSUL\n");
    retAsteriscos();
}
```

```
void linhaDeAsteriscos(){
    int i;
    for (i=1; i<=22; i++)
        printf("*");
}
```

```
void retAsteriscos(){
    int i;
    for (i=1; i<=3; i++){
        linhaDeAsteriscos();
        printf("\n");
    }
}
```

```
graph TD
    main["main()"] --> retAsteriscos["retAsteriscos()"]
    main --> linhaDeAsteriscos["linhaDeAsteriscos()"]
    linhaDeAsteriscos --> retAsteriscos
    retAsteriscos --> main
```


Subalgoritmos

Aula 20