Estruturas de Dados II

Aula 2

Prof. Patrícia Noll de Mattos

Revisão de Subrotinas

- Dividir um problema em **partes menores**
- Unidades **autônomas** de código
- Possuem uma **assinatura:** identificador, tipo de retorno, tipo e quantidade de parâmetros.
- Pode ser **executado várias vezes** quando **chamado** pelo seu nome
- Após sua chamada, o **controle da execução** passa para a **subrotina**, ao encerrar, retorno para o próximo comando após sua chamada.
- **Fluxo de execução**.

.

Chamada

- **Procedimentos**, não retornam valor
- **Funções:** retornam valor, podem ser utilizadas como se fossem os valores que retornam.

```
void mostra_vetor(int vet[], int i){
  . . .
}

int soma_vetor(int vet[], int i){
  . . .
}

int vet[10]; int i=10;
mostra_vetor(vet, i);
soma = soma_vetor(vet, i);
```

Protótipo de Função

```
#include <stdio.h>
#include <conio.h>
int soma(int v1, int v2);

int soma(int v1, int v2){
      int s;
      s = v1+v2;
      return s;
};
```

```
int main(){
      int valor1, valor2;
      printf("Digite um valor:");
      scanf("%i", &valor1); fflush(stdin);

      printf("Digite um valor:");
      scanf("%i", &valor2); fflush(stdin);

      printf("Soma=%i\n", soma(valor1,
      valor2));
      getch();
      return 0;
};
```

Passagem de Parâmetro

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

void troca(int v1, int v2);

void troca(int v1, int v2){
      int t;
      t = v1;
      v1 = v2;
      v2=t;
};
```

```
int main(){
      int valor1, valor2;
      printf("Digite um valor:");
      scanf("%i", &valor1); fflush(stdin);

      printf("Digite um valor:");
      scanf("%i", &valor2); fflush(stdin);

      troca(valor1, valor2);

      printf("Valor1=%i\n", valor1);
      printf("Valor1=%i\n", valor2);
      getch();

      return 0;
};
```

Passagem de Parâmetro

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

void troca(int *v1, int *v2);

void troca(int *v1, int *v2){
	int t;
	t = *v1;
	*v1 = *v2;
	*v2=t;
};
```

```
int main(){
	int valor1, valor2;
	printf("Digite um valor:");
	scanf("%i", &valor1); fflush(stdin);

	printf("Digite um valor:");
	scanf("%i", &valor2); fflush(stdin);

	troca(&valor1, &valor2);

	printf("Valor1=%i\n", valor1);
	printf("Valor1=%i\n", valor2);
	getch();

	return 0;
};
```

Recursividade:

- Uma função recursiva consiste em uma função que chama ela mesma!

```
int main()
{
    int valor;
    printf(“ Digite um valor:”);
    scanf(“%i”, &valor); fflush(stdin);
    printf(“Fatorial de %i e %i!\n”, valor, fatorial(valor));
    getch();
    return 0;
}
```

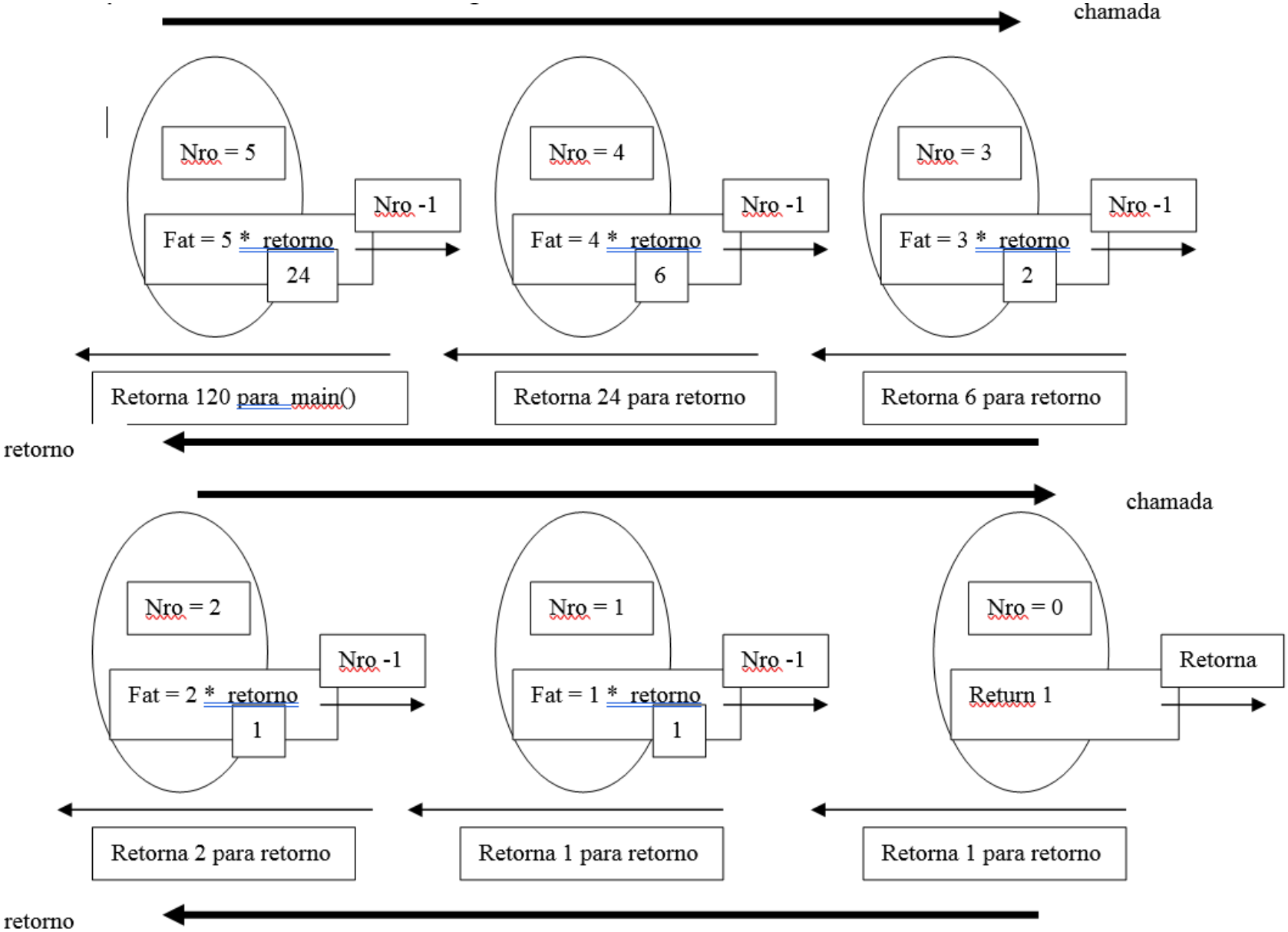
Caracteríticas:

- Condição de recursão e condição de parada.

```
int  fatorial ( int nro)
 {
     int fat=1;
     if ( nro == 0 )  return 1;
     else fat = nro *  fat (nro - 1);
     return fat;
  }
```

- Cada chamadas recursivas de uma função gera uma nova execução da função (réplica);
- a cada nova chamada recursiva, o contexto atual é empilhado, e, ao final da recursão, são desempilhados, gerando o resultado.
- Ou seja, o resultado vai sendo formado quando a recursão para e os valores começam a retornar!

Execução:


chamada
Nro = 5
Nro -1
Fat = 5 * retorno
24
Nro = 4
Nro -1
Fat = 4 * retorno
6
Nro = 3
Nro -1
Fat = 3 * retorno
2
Retorna 120 para main()
Retorna 24 para retorno
Retorna 6 para retorno
retorno
chamada
Nro = 2
Nro -1
Fat = 2 * retorno
1
Nro = 1
Nro -1
Fat = 1 * retorno
1
Nro = 0
Retorna
Return 1
Retorna 2 para retorno
Retorna 1 para retorno
Retorna 1 para retorno
retorno

Exercícios:

- Criar uma função recursiva que calcule a potência de um valor por outro, ambos recebidos por parâmetro.

- Criar um programa contendo uma função recursiva que percorra um vetor de 10 posições inteiras e apresente na tela seus elementos.