Universidade Federal Rural de Pernambuco
(UFRPE)
- Unidade Acadêmica de Garanhuns-

Encapsulamento

Introdução a Linguagem Java Parte III

Prof. Priscilla Kelly Machado Vieira

2019.1 – Aula 4

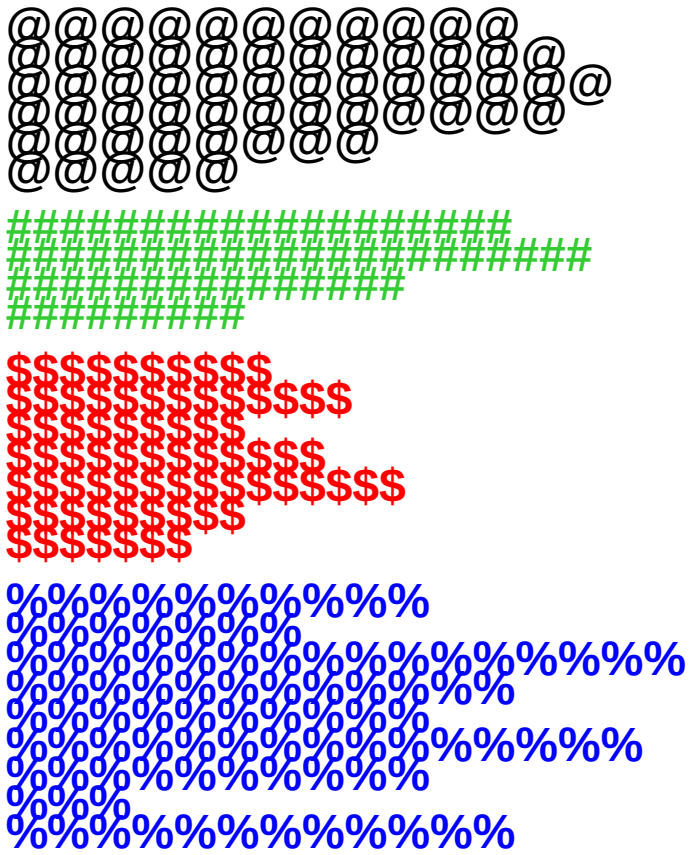
Apresentação do Capítulo

- Introdução
- Funções
- Recursividade
- Exercícios

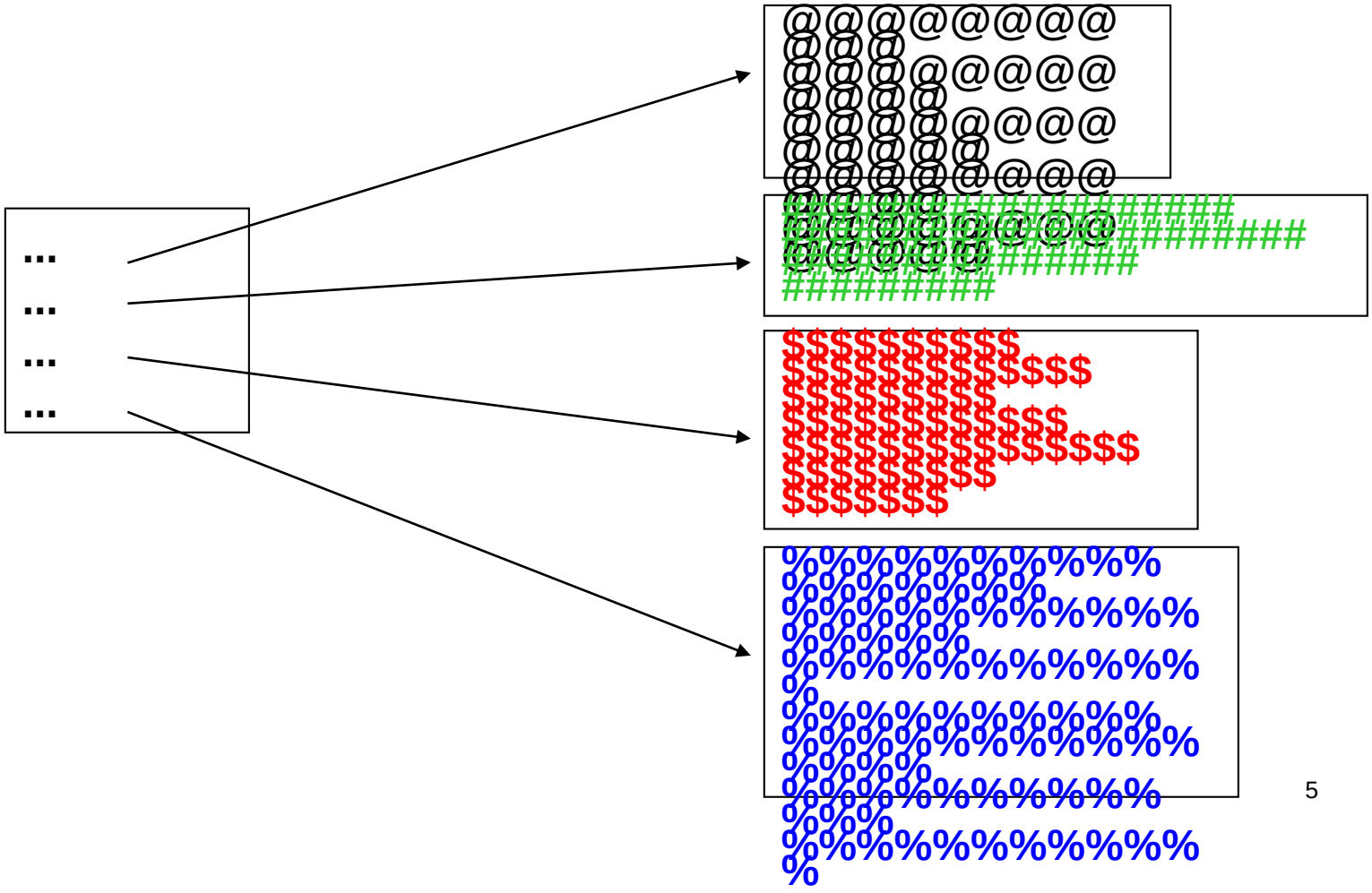
Introdução

- Como implementar um sistema computacional?
- No que pensar primeiro?
  - Estruturas de dados?
  - Operações?

Abordagem não modular

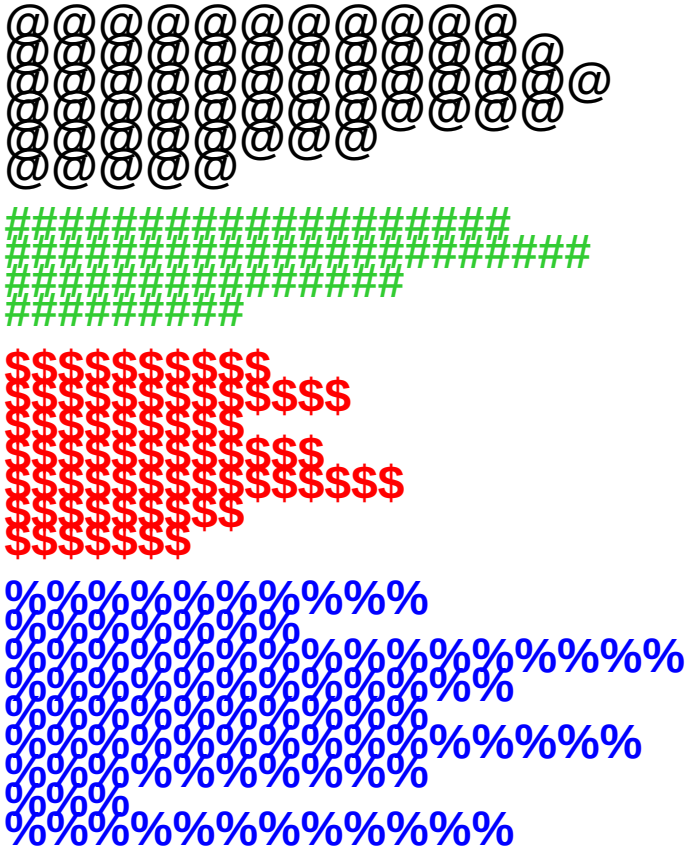


Abordagem Modular



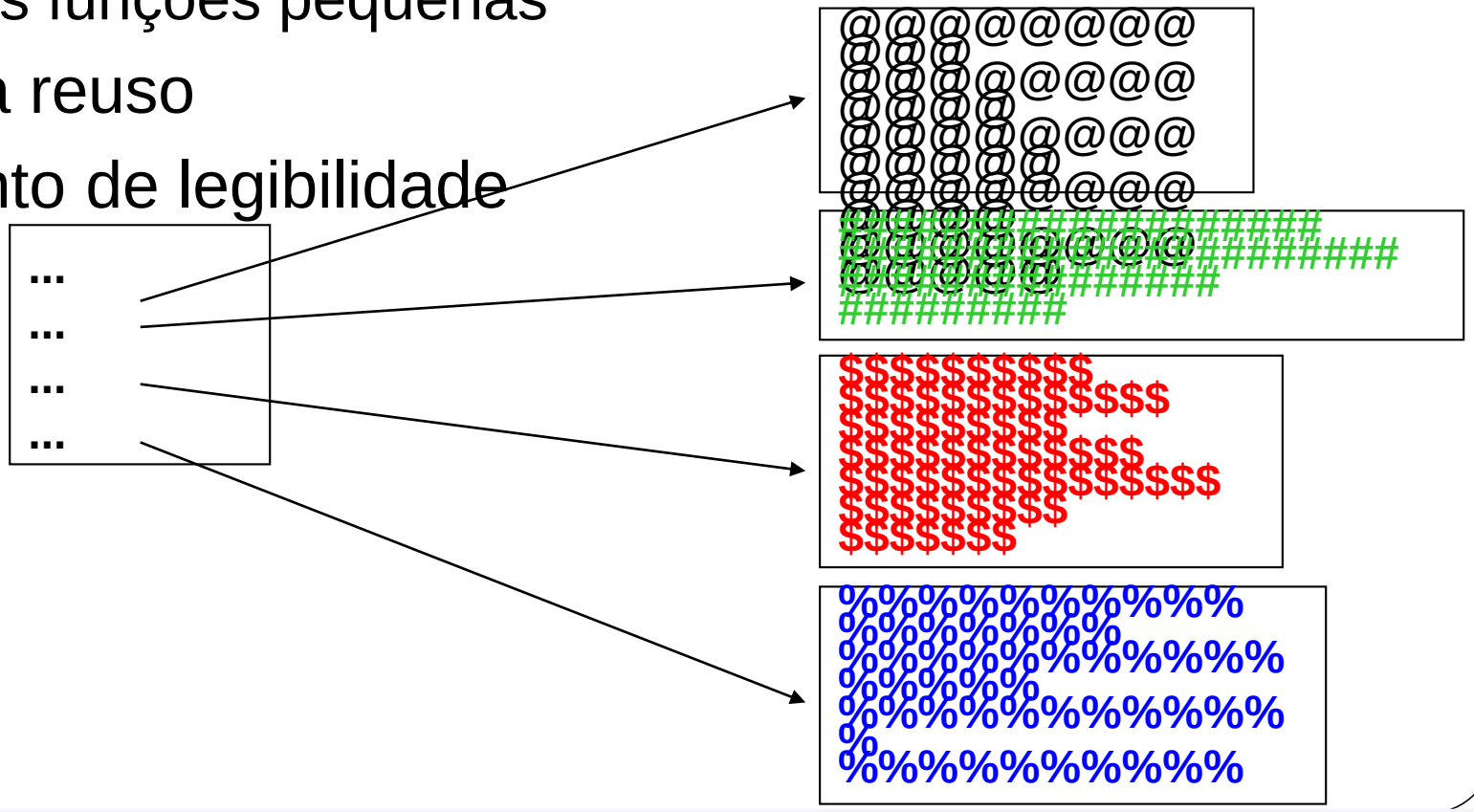
Abordagem não modular

- Main complexo
- Reduz o reuso



Abordagem Modular

- Main mais simples
- Facilita codificação:
  - Várias funções pequenas
- Facilita reuso
- Aumento de legibilidade



Exemplo


```
#include <stdio.h>

int main(){
  int n;
  printf("Digite o valor de n: ");
  scanf("%d",&n);
  int res;
  int i;
  res = 1;
  for(i=1;i<=n;i++){
    res = res*i;
  }

  printf("O resultado eh: %d", res);
}
```

Exemplo


```
#include <stdio.h>

int main(){
  int n;
  printf("Digite o valor de n:");
  scanf("%d",&n);

  int res;
  int i;
  res = 1;
  for(i=1;i<=n;i++){
    res = res*i;
  }

  printf("O resultado eh: %d", res);
}
```

Exemplo


```
#include <stdio.h>

int main(){
  int n;
  printf("Digite o valor de n:");
  scanf("%d",&n);
  int res;
  int i;
  res = 1;
  for(i=1;i<=n;i++){
    res = res*i;
  }

  printf("O resultado eh: %d", res);
}
```

Exemplo


```
#include <stdio.h>

int main(){
  int n;
  printf("Digite o valor de n: ");
  scanf("%d",&n);int res;
  int i;
  res = 1;
  for(i=1;i<=n;i++){
    res = res*i;
  }
  printf("O resultado eh: %d", res);
}
```

Funções

- Vantagens de se utilizar?
  - Dividir uma tarefa complexa em tarefas menores, permitindo esconder detalhes de implementação.
  - Evita-se a repetição de um trecho código ao longo do programa.
  - Trabalho pode ser dividido
- Sintaxe:

```
tipo_de_retorno nome(parâmetros) {
        instruções locais;
}
```

Funções

- Os parâmetros recebidos por uma função são separados por vírgulas.
  - Quando uma função retorna um valor, é obrigatório o uso do comando *return*.
  - Funções que não retornam valores têm como tipo de retorno void.

Funções


```
#include <stdio.h>
void fat (int n);
int main ( ){
int n;
scanf ("%d", &n);
fat (n);
return 0;
}
void fat (int n){
int i;
int f = 1;
for(i=1; i<=n, i++){
    f*=i;
}
printf("Fatorial = %d\n, f");
}
```

Modularização em C

- Declarar a funcao antes da main() e implementar depois
- Implementar a funcao antes da main()
- Implementar a funcao em um arquivo a parte e inclui-lo no arquivo da main()

Modularização em C

- Função que não recebe parâmetros e não retorna valor

```
#include <stdio.h>

void desenha();

void main(){
    printf(”Usando funções.");
    desenha( );
}
void desenha(){
  int i;
  for (i = 0; i <= 10; i++)
    printf("--\n");
}
```

Assinatura, protótipo ou declaracao da funcao

Implementação da função

Modularização em C

- Função que recebe parâmetros e retorna valor
- Implementação antes da main()

```
#include <stdio.h>

int soma(int a, int b)
{
   return (a + b);
}
void main()
{
   printf(”Usando funções.");
   printf(“soma = %d”, soma(2, 2);
}
```

Modularização em C

- Função que recebe parâmetros e retorna valor
- Implementação em arquivo separado(biblioteca)

```
#include <stdio.h>
#include “funcoes.h”

void main()
{
    printf("Usando funções.");
    printf("soma = %d", soma(2, 2));
}
```

funcoes.h

```
int soma(int a, int b)
{
    return (a + b);
}
```

Parâmetros

- Passagem de parâmetro por valor: uma cópia daquele valor real / variável é feita para ser manipulada dentro da função.
  - O que é uma variável local?
  - O que é uma variável global?

As variáveis locais (inclusive os parâmetros) definidas dentro do corpo de uma função só existem dentro dela

```
#include <stdio.h>

int soma(int a, int b);

main(){
  int a = 2;
  int b = 2;
  printf("soma = %d\n, soma(a, b));
  printf("a = %d\n", a);
}
```

```
int soma(int a, int b)
{
  a = a + b;
  return (a);
}
```

Parâmetros

- Passagem de parâmetro por referência: Veremos na aula sobre ponteiros ...

Variáveis Globais

- Visível a todas as funções subsequentes
- Existem enquanto o programa estiver sendo executado

```
#include <stdio.h>
int s, p; /*variáveis globais*/
void somaProduto (int a, int b){
s= a+b;
p= a*b;
}
int main (void){
int x, y;
scanf ("%d %d", &x, &y);
somaProduto(x, y);
printf("Soma= %d produto = %d\n", s, p)
return 0;}
```

Recursividade

- O que é recursão?
- Qual a diferença entre recursão e iteração?
- Recursão e iteração tem o mesmo poder computacional?

Iteração x Recursão


```
float fatorial(int n) {
 int i;
 float f = 1;
 for(i = 2; i <= n; i++){
    f = f * i;
 }
 return f;
}
```

```
float fatorial(int n) {
 float resp = 1;
 if(n > 0) {
   resp = n * fatorial(n - 1);
 }
 return resp;
}
```

Exercício

- Laboratório 03
  - Construa um algoritmo que leia 3 números inteiros A, B e C e que, utilizando funções, mostre-os:
    - A + B + C
    - (A*B*C) * MIN (A, B, C)
    - (A*B*C)*MAX (A,B,C)
  - Utilize um arquivo externo para ser chamado dentro do programa com a main
  - Utilize funções