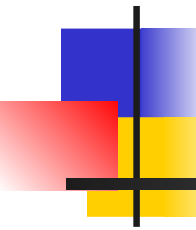
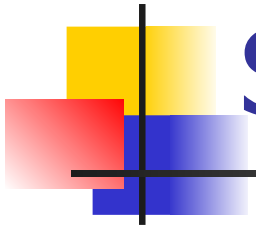


SCC0221 – Introdução à Ciência da Computação I

Prof.: Dr. Rudinei Goularte

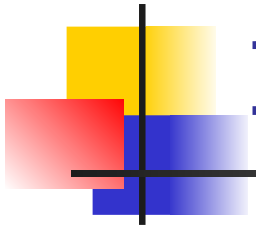
Alocação Dinâmica

Instituto de Ciências Matemáticas e de Computação – ICMC
Sala 4-229



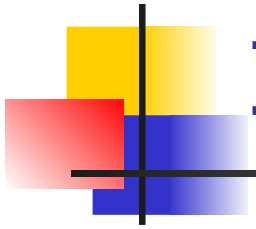
Sumário

- 1. Typedef
- 2. Divisão da Memória
- 3. Alocação Dinâmica
- 4. Exemplo: Alocação de Matriz



1. typedef

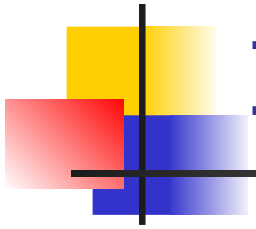
- Permite que se associe explicitamente um tipo a um identificador.
  - Composição de novos tipos de dados a partir de tipos pré-existentes.
  - Não cria um novo tipo de dado.



1. typedef

- Forma geral:
  - **typedef** *tipo identificador*;
  - *tipo*:qualquer tipo válido em C.
  - *identificador*: um identificador válido em C.

- Exemplos:
  - typedef char letra;
  - typedef int polegadas, METROS;

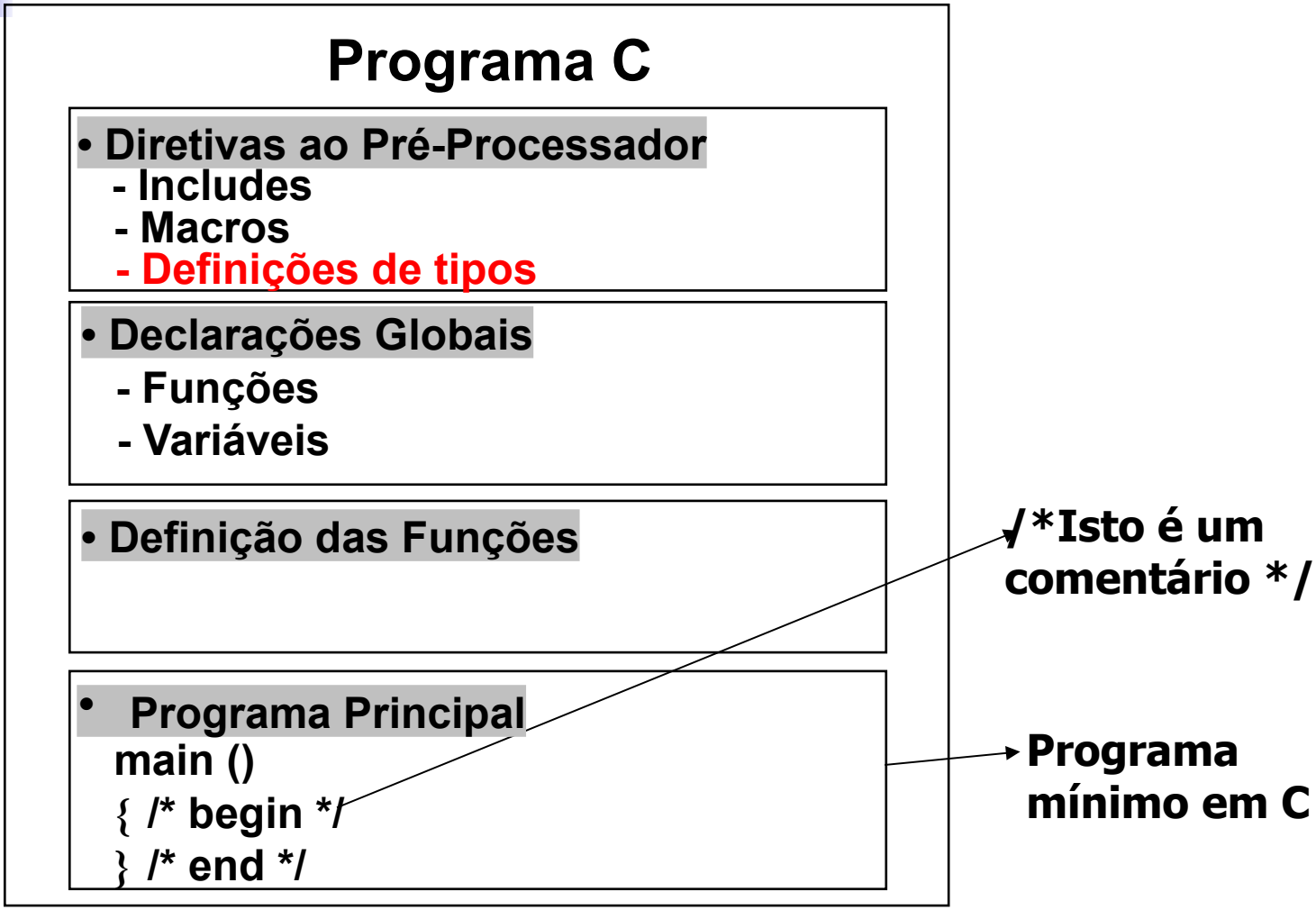


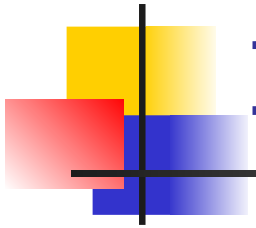
1. typedef

- Os identificadores de cada definição de tipo podem ser usados para declarar variáveis ou funções.

- Exemplos
  - letra u;
  - polegadas altura, largura;

1. typedef

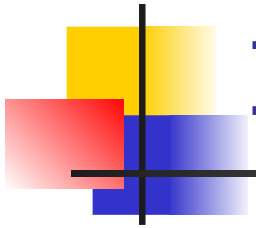

Programa C
• Diretivas ao Pré-Processador
- Includes
- Macros
- Definições de tipos
• Declarações Globais
- Funções
- Variáveis
• Definição das Funções
• Programa Principal
main ()
{ /* begin */
} /* end */
/*Isto é um comentário */
Programa mínimo em C



1. typedef

- Exemplos

```
#include <stdio.h>
typedef float num_real;
typedef int medida;
typedef medida altura;
int main (void) {
        altura alt=20;
        int x=4, i;
        i = alt / x;
        return(0);
}
```

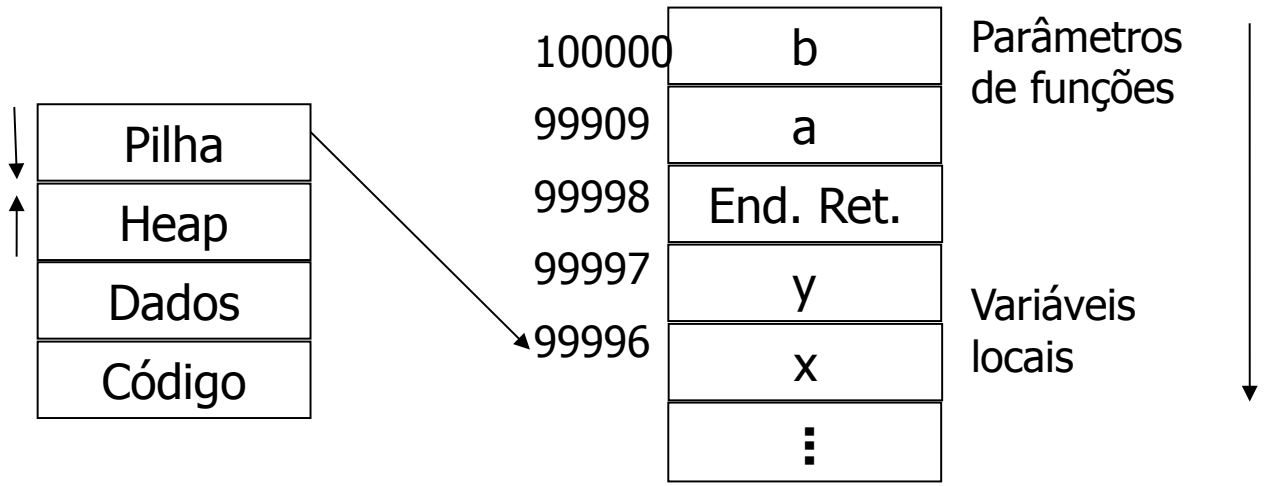


1. typedef

- Qual a vantagem de criar uma nova nomenclatura para um tipo existente?
  - Abreviar declarações longas.
  - Possuir nomes de tipos que refletem para que serão utilizados.
  - Facilitar a portabilidade.

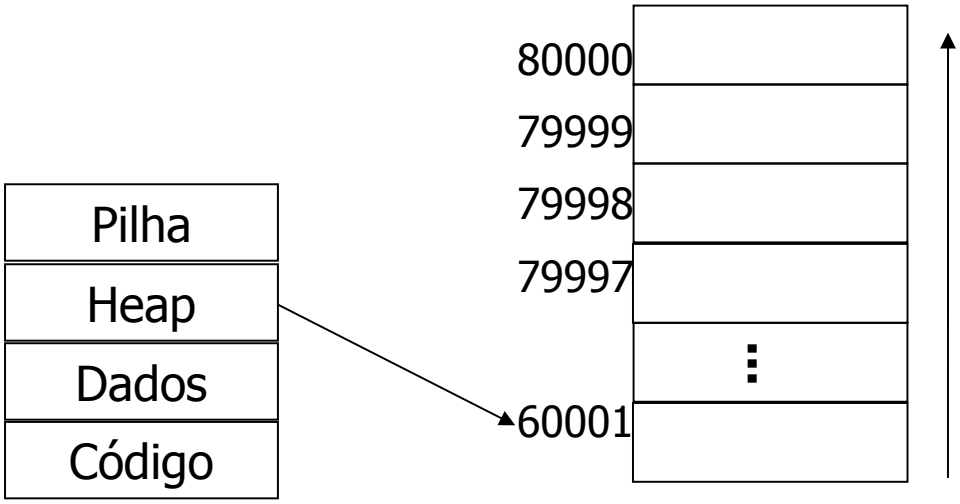
2. Divisão da Memória

- Stack (Pilha)
  - Estrutura de dados para armazenamento.
  - Stack cresce ao contrário do resto da memória.



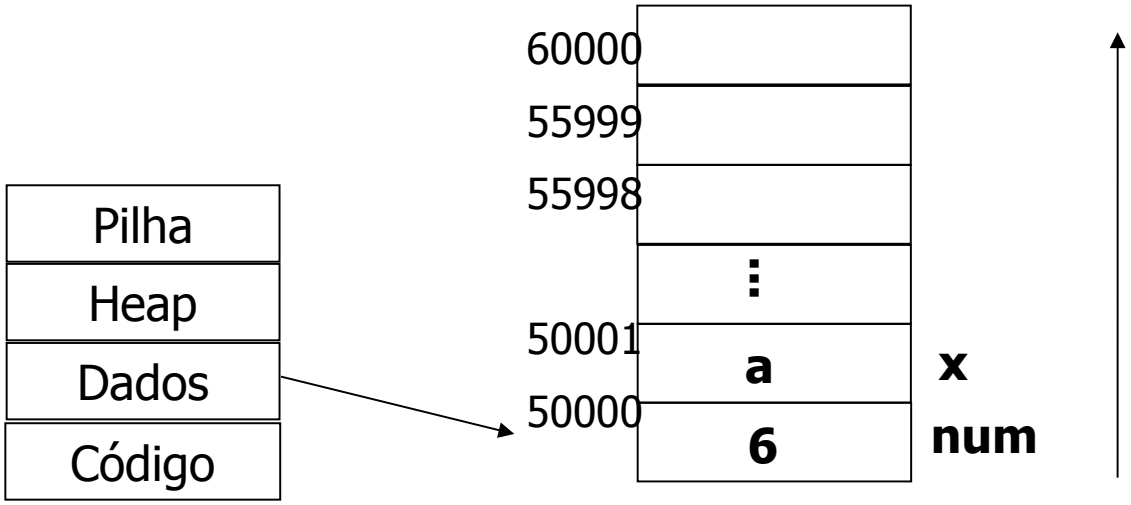
2. Divisão da Memória

- Heap
  - Memória livre para alocação



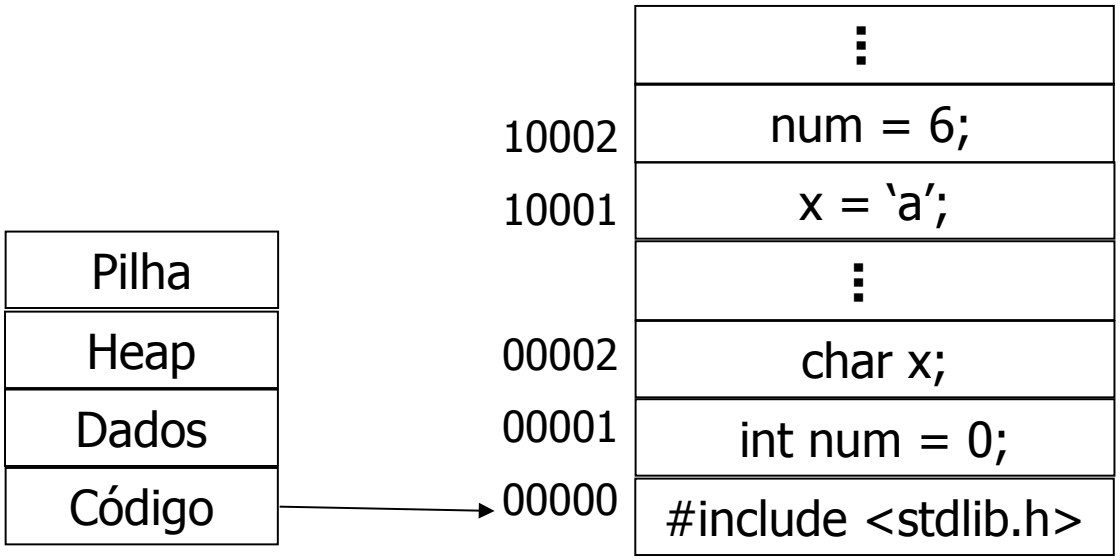
2. Divisão da Memória

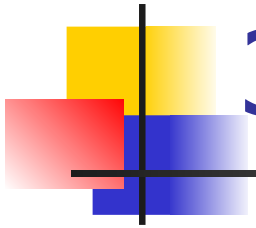
- Dados
  - Área da memória dedicada às declarações e definições globais.



2. Divisão da Memória

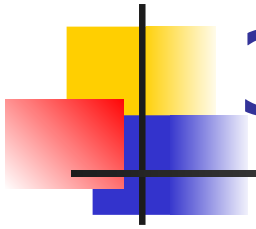
- Código
  - Área da memória onde está armazenado, em modo binário, o programa sendo executado.





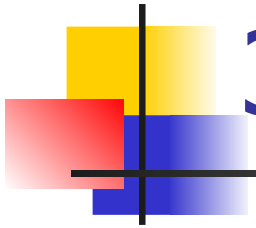
3. Alocação Dinâmica

- É um meio pelo qual um programa pode obter memória enquanto está em execução.
- C fornece funções para realizar alocação dinâmica de memória.
- A memória alocada dinamicamente é obtida do *heap*.



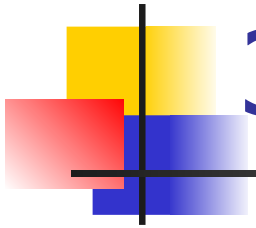
3. Alocação Dinâmica

- Funções calloc(), malloc(), realloc() e free().
  - Funções ANSI, disponíveis no header stdlib.h.
  - calloc() e malloc() criam espaço na memória.
  - free() libera espaço alocado com calloc() ou malloc().
  - realloc() realoca espaço previamente alocado.



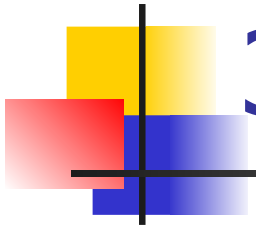
3. Alocação Dinâmica

- Observações:
  - NULL.
    - Valor nulo.
  - Ponteiro NULL.
    - Ponteiro para um tipo específico que, por ora, aponta para nada. Não tem valor.
  - Ponteiro void.
    - Ponteiro genérico, aponta para qualquer tipo de dado.



3. Alocação Dinâmica

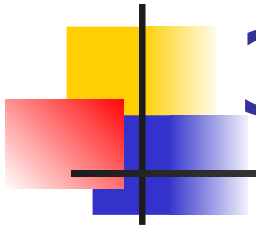
- void* calloc(int n, size_t tam);
  - Aloca espaço contíguo na memória para **n** elementos, com cada elemento tendo **tam** bytes.
  - O espaço é inicializado com todos os bits setados para zero.
  - Sucesso retorna um ponteiro void para o endereço base do espaço alocado.
  - Falha retorna NULL.



3. Alocação Dinâmica

- Exemplo
  - Alocação de um inteiro

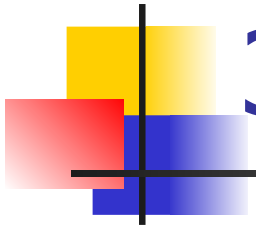
```
int *p;
p = (int *) calloc(1, 4); /* supondo inteiro de 4
                                                bytes */
ou
p = (int *) calloc(1, sizeof(int)); /* portabilidade */
```



3. Alocação Dinâmica

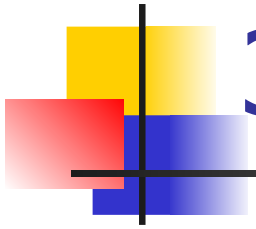
- Exemplo (Alocação de vetor)
  - Aloca espaço de memória para n inteiros

```
int n = 5, *p;
p = (int *) calloc(n, sizeof(int));
```



3. Alocação Dinâmica

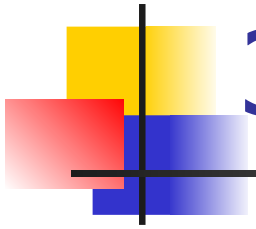
- void* malloc(size_t tam);
  - Aloca **tam** bytes de espaço na memória (no heap).
  - Diferente de calloc(), malloc() não inicializa o espaço de memória alocado.
  - Sucesso retorna um ponteiro void para o endereço base do espaço de memória alocado.
  - Falha retorna NULL.



3. Alocação Dinâmica

- Exemplo
  - Alocação de um inteiro

```
int *p;
p = (int *) malloc(sizeof(int));
```



3. Alocação Dinâmica

- Exemplo (Alocação de vetor)
  - Aloca espaço de memória para n inteiros

```
int n = 5, *p;
p = (int *) malloc(n * sizeof(int));

p[0] = 1;
p[1] = 2;
```

3. Alocação Dinâmica

- A memória disponível para ser alocada (*heap*) não é infinita.
- Modo correto de usar calloc() ou malloc():

```
if ((p=(int*) malloc(12)) == NULL) {
        printf("Sem memória");
        exit(1); /*ou outro método de tratar erro*/
}
```

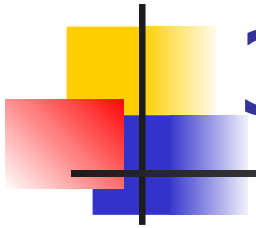
3. Alocação Dinâmica

- void *realloc(void *ptr, size_t size);
  - Tenta redimensionar o bloco de memória apontado por **ptr** previamente alocado com calloc() ou malloc().
  - **size** é o tamanho em bytes desejado para o novo bloco de memória.
  - Sucesso retorna um ponteiro void para o novo endereço base do espaço de memória alocado.
  - Falha retorna NULL.

3. Alocação Dinâmica

- Exemplo:

```
if ((p=(int*) malloc(12*sizeof(int)) ) == NULL) {
    printf("Sem memória");
    exit(1);
}
for(int i =0; i<12; i++)
   p[i] = i;
if ( (p=(int*) realloc(p, 12*sizeof(int) + 2) )== NULL){
   printf("Sem memória");
    exit(1);
}
p[12] = 12;
p[13] = 13;
```



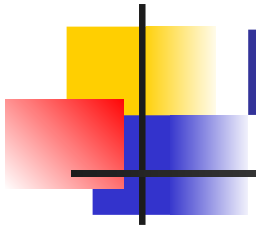
3. Alocação Dinâmica

- void* free(void *ptr);
  - Libera espaço de memória alocado por malloc() ou por calloc().
  - Memória alocada deve ser explicitamente devolvida ao SO usando free().
  - Caso contrário, só será devolvida quando o programa terminar.
  - **Boa prática de programação: usar free() para todo ponteiro quando o mesmo não for mais necessário.**

3. Alocação Dinâmica

- Exemplo:

```
int *p;
p = (int *) malloc(sizeof(int));
*p = 10;
printf("%d", *p);
free(p);
```



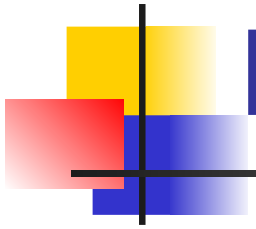
Exercício

- Desenvolva um programa em C que aloque dinamicamente um vetor de doubles com a dimensão (tamanho) e os elementos determinados pelo usuário.
- O programa deve ser modularizado – no mínimo uma função que realize a alocação dinâmica.
- Faça a documentação interna do seu programa.

4. Exemplo: Alocação de Matriz

```
int i, j, n;
double **m;
scanf ("%d", &n);
m = (double **) calloc(n, sizeof(double *));
for (i = 0; i < n; i++)
    m[i] = (double *) calloc(n, sizeof(double));
    /*não se verificou se a alocação foi bem sucedida.*/

for (i = 0; i < n; i++)
    for(j = 0; j < n; j++)
        m[i][j] = rand();
...
free(m); /* Liberando memória alocada.
            Deve-se liberar todos os ponteiros alocados, e
            não apenas "m". Memory Leak!*/
```



Exercício

- Modifique o exercício de multiplicação de matrizes de modo que as dimensões das matrizes sejam definidas dinamicamente. Isso irá economizar memória!