

Programação 2

Ponteiros e Alocação Dinâmica

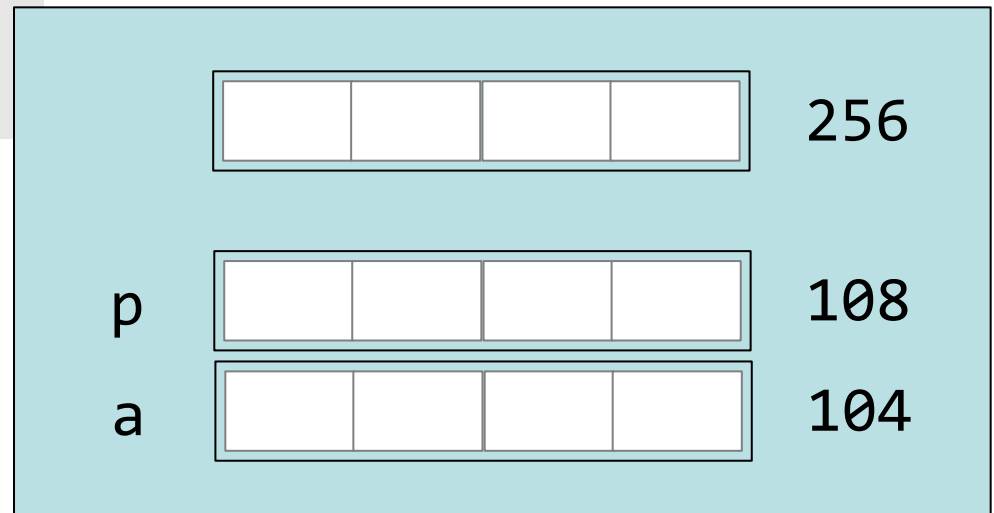
Rivera

Ponteiros

```
int main ( void )  
{  
    int a;  
    int *p;  
    p = &a;  
    *p = 2;  
    printf(" %d ", a);  
    return 0;  
}
```

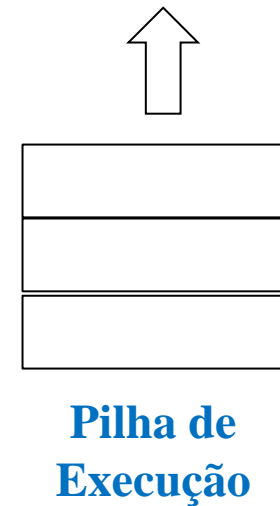
Operadores unários

- & (“endereço de”)
✓ &a
- * (“conteúdo de”)
✓ *p



Ponteiros: cuidados

```
int main ( void )  
{  
    int a, b, *p;  
    a = 2;  
    *p = 3;  
    b = a + (*p);  
    printf(" %d ", b);  
    return 0;  
}
```



Erro em $*p = 3$

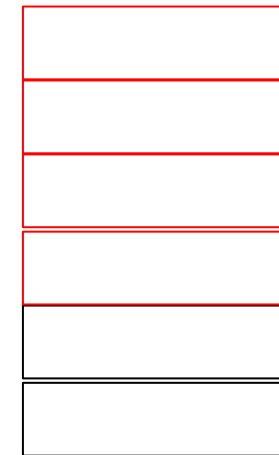
- Memória apontada por p não inicializada
- 3 armazenado num espaço de memória desconhecido

Funções que mudam valores de variáveis de outras

```
#include <stdio.h>
void troca(int a, int b);

int main (void)
{
    int a=10, b=20;
    troca(a,b);
    printf(" a=%d b= %d\n",a,b);
}

void troca(int a, int b)
{
    int tmp=b;
    b=a;
    a=tmp;
}
```



**Pilha de
Execução**

a = 10 b = 20
Press any key to continue

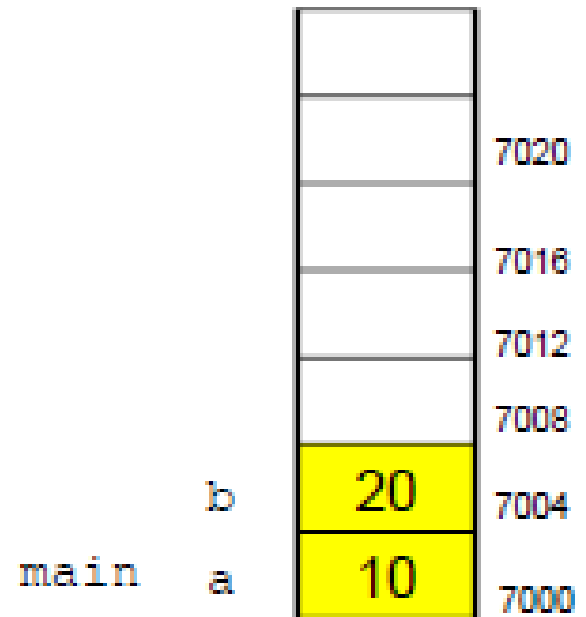
Funções que mudam valores de variáveis de outras

```
#include <stdio.h>
void troca(int a, int b);

int main (void)
{
    int a=10, b=20;
    troca(a,b);
    printf(" a=%d b= %d\n",a,b);
}

void troca(int a, int b)
{
    int tmp=b;
    b=a;
    a=tmp;
}
```

Pilha de memória



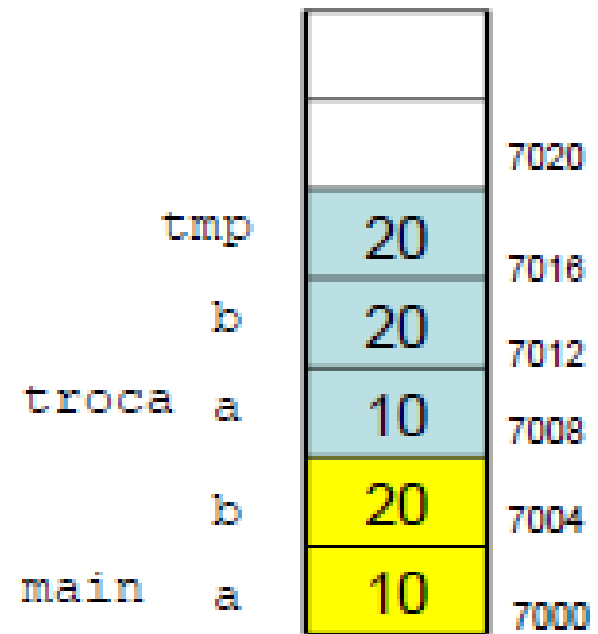
Funções que mudam valores de variáveis de outras

```
#include <stdio.h>
void troca(int a, int b);

int main (void)
{
    int a=10, b=20;
    troca(a,b);
    printf(" a=%d b= %d\n",a,b);
}

void troca(int a, int b)
{
    → int tmp=b;
    b=a;
    a=tmp;
}
```

Pilha de memória



Funções que mudam valores de variáveis de outras

```
#include <stdio.h>
void troca(int a, int b);

int main (void)
{
    int a=10, b=20;
    troca(a,b);
    printf(" a=%d b= %d\n",a,b);
}

void troca(int a, int b)
{
    int tmp=b;
    → b=a;
    a=tmp;
}
```

Pilha de memória

				7020
tmp		20		7016
b		10		7012
troca	a	10		7008
	b	20		7004
main	a	10		7000

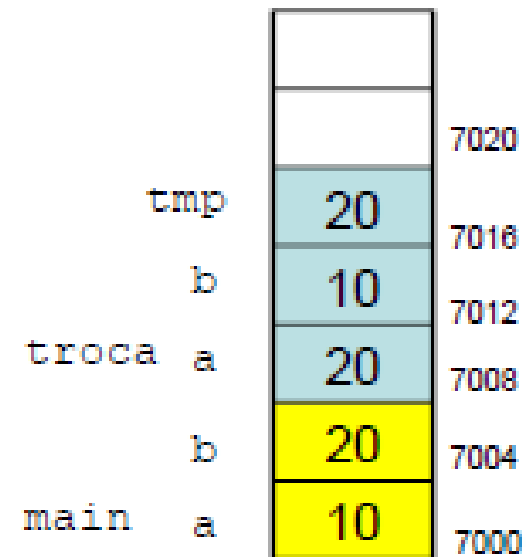
Funções que mudam valores de variáveis de outras

```
#include <stdio.h>
void troca(int a, int b);

int main (void)
{
    int a=10, b=20;
    troca(a,b);
    printf(" a=%d b= %d\n",a,b);
}

void troca(int a, int b)
{
    int tmp=b;
    b=a;
    a=tmp;
}
```

Pilha de memória



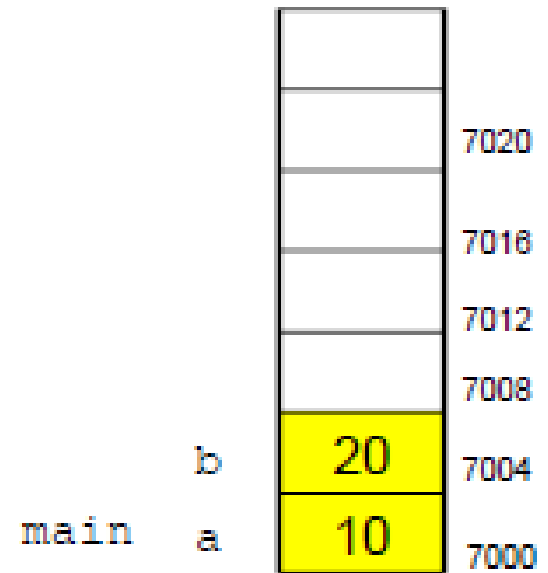
Funções que mudam valores de variáveis de outras

```
#include <stdio.h>
void troca(int a, int b);

int main (void)
{
    int a=10, b=20;
    troca(a,b);
    printf(" a=%d b= %d\n",a,b);
}

void troca(int a, int b)
{
    int tmp=b;
    b=a;
    a=tmp;
}
```

Pilha de memória



Ponteiros

- Passagem de ponteiros para funções:
 - ♦ Função g chama função f
 - f não pode alterar diretamente valores de variáveis de g

```
#include <stdio.h>
void troca(int *, int *);
int main (void)
{
    int a=10, b=20;
    troca(&a,&b);
    printf(" a = %d b = %d \n", a, b);
}
```

```
void troca(int *pa, int *pb)
{
    int tmp = *pb;
    *pb = *pa;
    *pa = tmp;
}
```

a=20 b=10

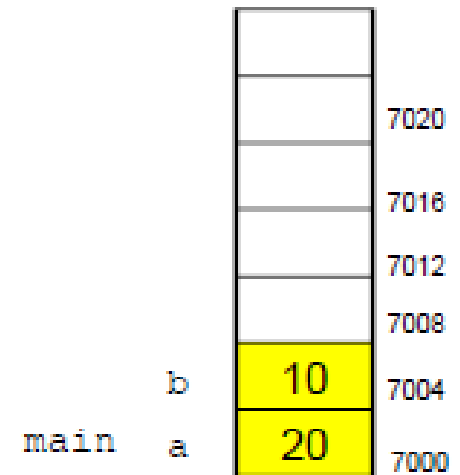
Press any key to continue

Ponteiros

```
#include <stdio.h>
void troca(int *pa, int *pb);
int main (void)
{
    int a=10, b=20;
    → troca(&a,&b);
    printf(" a = %d b = %d \n", a, b);
}


void troca(int *pa, int *pb) {
    int tmp=*pb;
    *pb=*pa;
    *pa=tmp;
}
```

Pilha de memória



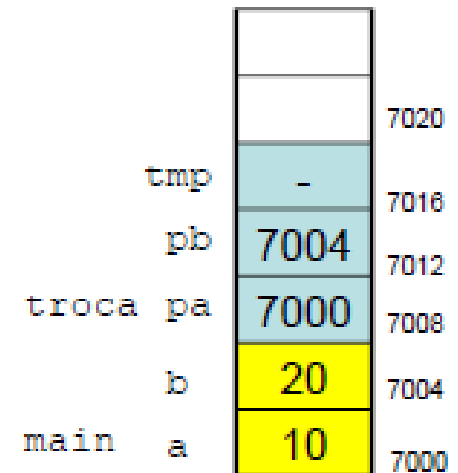
Ponteiros

```
#include <stdio.h>
void troca(int *pa, int *pb);
int main (void)
{
    int a=10, b=20;
    troca(&a,&b);
    printf(" a = %d b = %d \n", a, b);
}
```



```
void troca(int *pa, int *pb) {
    int tmp=*pb;
    *pb=*pa;
    *pa=tmp;
}
```

Pilha de memória



Ponteiros

```
#include <stdio.h>
void troca(int *pa, int *pb);
int main (void)
{
    int a=10, b=20;
    troca(&a,&b);
    printf(" a = %d b = %d \n", a, b);
}
```

→

```
void troca(int *pa, int *pb) {
    int tmp=*pb;
    *pb=*pa;
    *pa=tmp;
}
```

Pilha de memória

			7020
tmp	20		7016
pb	7004		7012
troca pa	7000		7008
b	20		7004
main a	10		7000

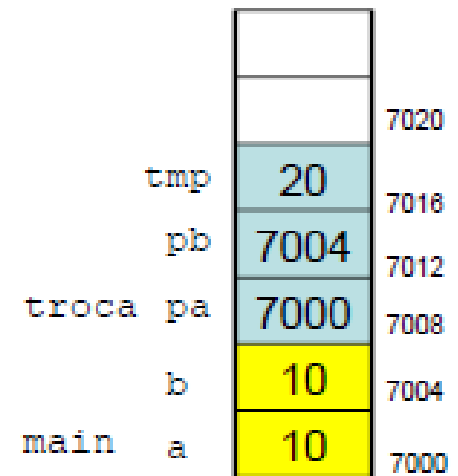
Ponteiros

```
#include <stdio.h>
void troca(int *pa, int *pb);
int main (void)
{
    int a=10, b=20;
    troca(&a,&b);
    printf(" a = %d b = %d \n", a, b);
}
```

```
void troca(int *pa, int *pb) {
    int tmp=*pb;
    *pb=*pa;
    *pa=tmp;
}
```



Pilha de memória



Ponteiros

```
#include <stdio.h>
void troca(int *pa, int *pb);
int main (void)
{
    int a=10, b=20;
    troca(&a,&b);
    printf(" a = %d b = %d \n", a, b);
}
```

```
void troca(int *pa, int *pb) {
    int tmp=*pb;
    *pb=*pa;
    *pa=tmp;
}
```



Pilha de memória

			7020
tmp	20		7016
pb	7004		7012
troca pa	7000		7008
b	10		7004
main a	20		7000

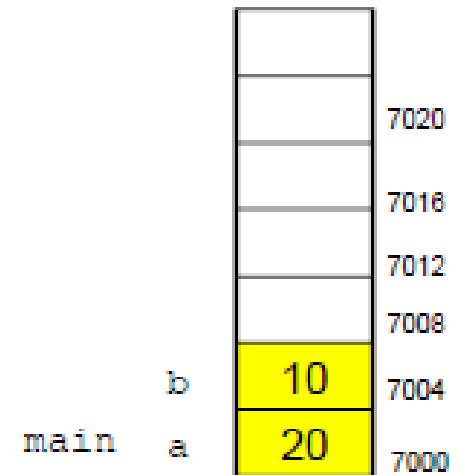
Ponteiros

```
#include <stdio.h>
void troca(int *pa, int *pb);
int main (void)
{
    int a=10, b=20;
    troca(&a,&b);
    printf(" a = %d b = %d \n", a, b);
}
```



```
void troca(int *pa, int *pb) {
    int tmp=*pb;
    *pb=*pa;
    *pa=tmp;
}
```

Pilha de memória



Trabalho

- Implementar um programa que faça:
 - Defina um valor para n (número de elementos)
 - Digite os n elementos inteiros
(A) → Cada elemento digitado deve ser colocada na sequencia certa (de menor a maior) por uma função “ColacaNaPosiçãoCerta”
 - Mostrar o resultado ordenado
- Um programa anterior com 2 vetores
 - V1: cada valor digitado em sequencia normal
 - V2: vetor de índice que ordena V1 por índices do vetor
(B) Modificar (A) para colocar os índices na posicao certa para ordenas V1.

Alocação Dinâmica

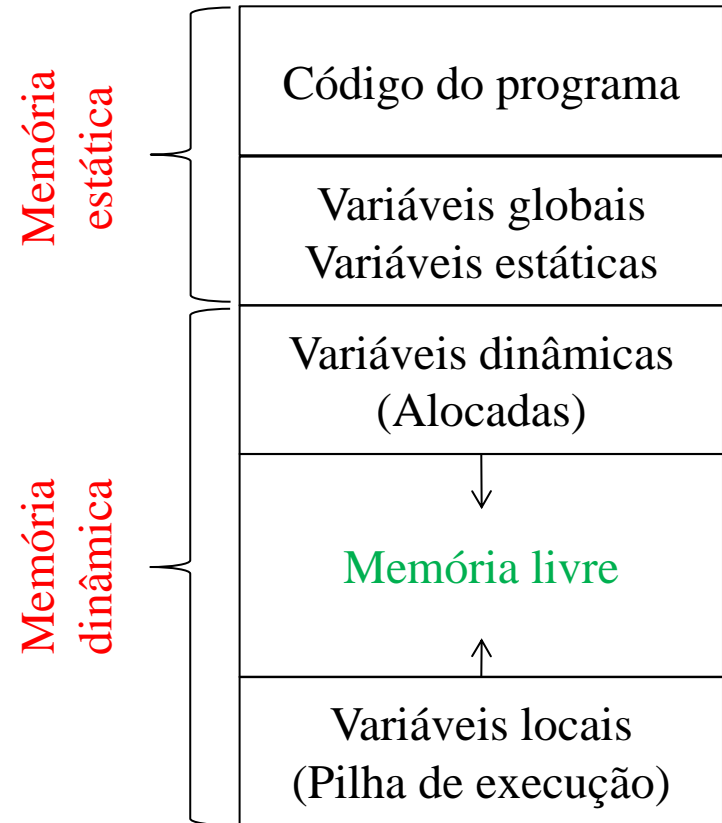
- Uso da memória
 - ◆ Uso de variáveis globais (e estáticas)
 - Espaço reservado enquanto o programa estiver ativo
 - ◆ Uso de variáveis locais
 - Espaço reservado apenas enquanto a função proprietária estiver ativa
 - ◆ Variáveis globais ou locais
 - Simples ou vetores
 - Vetor precisa número máximo de elemento
 - Compilador precisa calcular o espaço para reservar

Alocação Dinâmica

- Uso da memória
 - ♦ Alocação dinâmica
 - Espaço de memória requisitado em tempo de execução
 - Espaço permanece reservado até que seja explicitamente liberado
 - Espaço disponível depois de liberado
 - Espaço alocado e não liberado explicitamente
 - » Automaticamente liberado ao final da execução

Alocação Dinâmica

- Uso da memória
 - ♦ Memória estática
 - Código do programa
 - Variáveis globais
 - Variáveis estáticas
 - ♦ Alocação dinâmica
 - Variáveis dinâmicas (alocadas)
 - Memória livre
 - Variáveis locais



Alocação Dinâmica

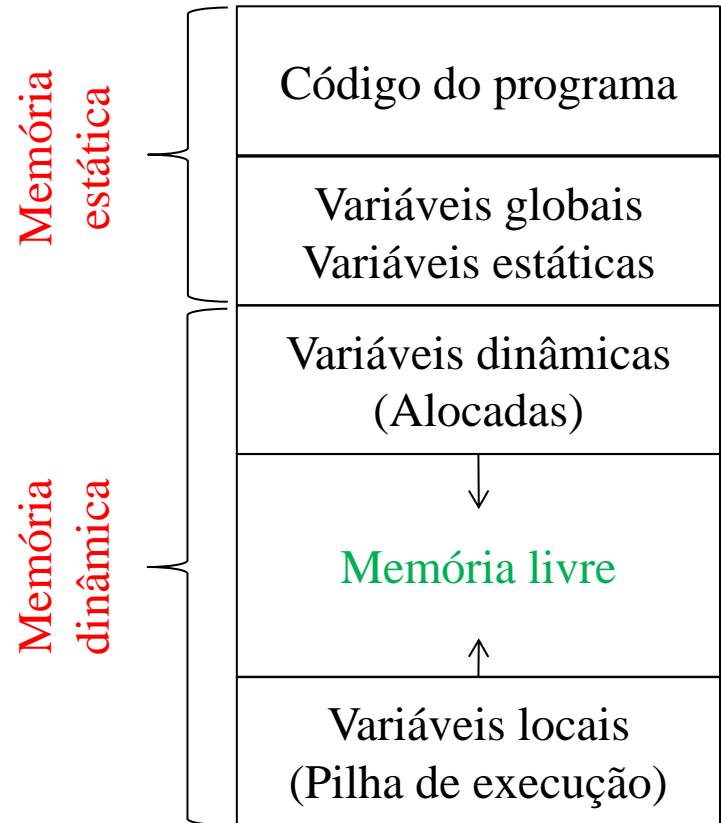
- Uso da memória

- ♦ Alocação dinâmica de memória

- Usa a memória livre
- Se espaço de memória livre for menor que o requisitado
 - Não aloca. Tratar erro.

- ♦ Pilha de execução

- Alocação de memória quando ocorre chamada de função
 - Sistema reserva o espaço para variáveis locais da função
 - Liberado o espaço quando termina a função
- Espaço não disponível (pilha cresce)
 - Programa abortado com erro



Alocação Dinâmica em C

- Funções da biblioteca padrão “stdlib.h”

- ♦ Contém funções pré-determinadas

- Funções para tratar alocação dinâmica de memória

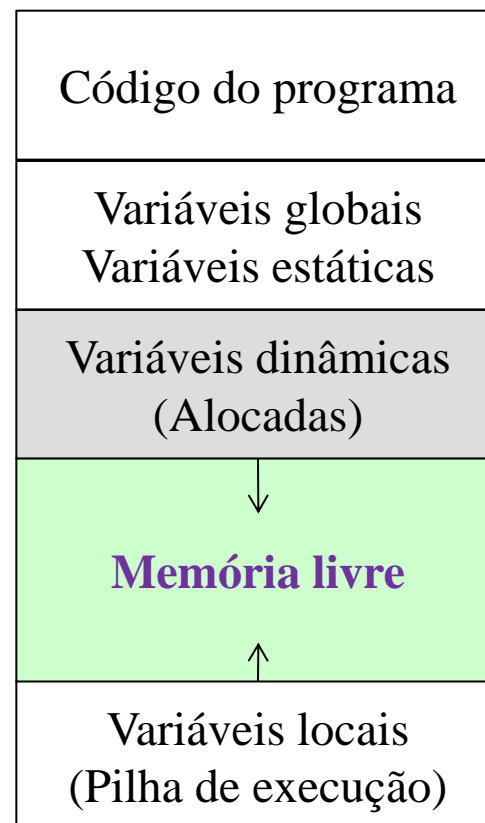
`void * malloc (int num_bytes);`

`void * calloc (int num, int num_bytes);`

`void free (v);`

- Constantes pré-determinadas

-



Alocação Dinâmica em C

- Função

`void * malloc (int num_bytes)`

- ♦ Retorna ponteiro genérico

- Ponteiro convertido para o tipo apropriado
 - `x = malloc (10 * sizeof (int)); // int * x;`
- Ponteiro convertido explicitamente
 - `x = (int *) malloc (10 * sizeof(int)); // int * x;`
- Retorna endereço nulo (`NULL`)
 - Se não houver espaço

- ♦ Operador “`sizeof`”

- Retorna o número de bytes ocupado por um tipo

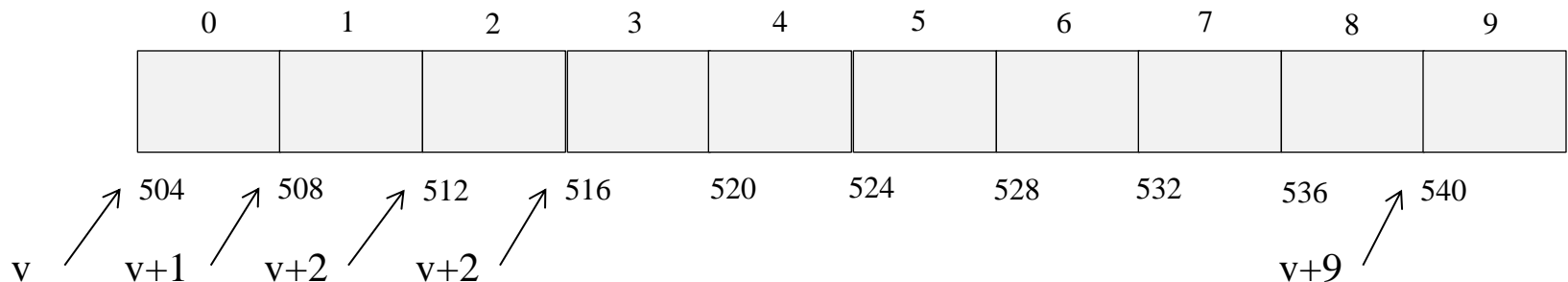
- ♦ Função “`free`”

- Parâmetro ponteiro de memória

Alocação Dinâmica

- Exemplo (em C):
 - ♦ Alocação dinâmica de um vetor de 10 elementos
 - Malloc retorna o endereço da área alocada
 - Ponteiro recebe endereço inicial do espaço alocado

```
int *v;  
v = (int *) malloc ( 10 * sizeof (int))  
If ( v == NULL)  
{  
    printf (“ erro na alocação de 10 * int);  
    exit();  
}
```



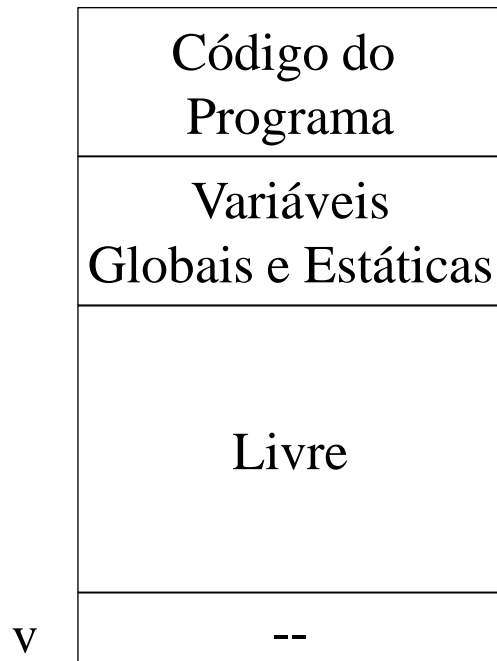
Alocação Dinâmica

- Exemplo (cont.):

```
v = (int *) malloc ( 10 * sizeof (int))
```

1: Declaração **int *v**

Abre-se espaço na pilha
para o ponteiro (var local)



2: Alocação:

v = (int *) malloc (10*sizeof(int))

Reserva espaço de memória da área
livre e atribui endereço à variável



```
#include <stdlib.h>
int main ( void )
{
    float *v;
    float med, var;
    int i,n;
    printf("Entre n e depois os valores\n");
    scanf("%d",&n);
    v = (float *) malloc(n*sizeof(float));
    if (v==NULL) { printf("Falta memoria\n"); exit(1); }

    for ( i = 0; i < n; i++ )
        scanf("%f", &v[i]);

    med = media(n,v);
    var = variancia(n,v,med);

    printf ( "Media = %f Variancia = %f \n", med, var);
    free(v);
    return 0;
}
```

Exercícios:

1. Dado um conjunto de arquivos de texto. Estabelecer uma ordem de acesso aos arquivos dependendo do número de caracteres, exceto mais de dois espaços, que eles contêm.
2. Ler de um arquivo N nomes de pessoas e suas respectivas idades. Agrupar as pessoas por idade e mostrar nomes por cada idade.
3. Existem um conjunto de N pontos no espaço 3D que representam vértices de faces poligonais (4 lados) de um objeto. Deseja-se pintar as faces do objeto com uma cor (R,G,B) proporcional ao ângulo definido entre o vetor normal de cada face e o vetor de raio de luz que está na origem do sistema cartesiano. O vetor normal de uma face é computado como produto vetorial de dois vetores arestas que tem um ponto (vértice) de origem comum.