Programação 2

Aula 5

Thiago Cavalcante – thiago.cavalcante@penedo.ufal.br

27 de novembro de 2019

Universidade Federal de Alagoas – UFAL
Campus Arapiraca
Unidade de Ensino de Penedo

Alocação dinâmica

Uma variável é uma posição na memória previamente* reservada que pode armazenar um dado

*previamente $\rightarrow$ toda variável deve ser declarada

Alocação dinâmica

Nem sempre é possível saber quanta memória um programa vai precisar (ex. funcionários)

A linguagem C permite alocar dinamicamente blocos de memórias usando ponteiros

Alocar $\rightarrow$ reservar

Dinamicamente $\rightarrow$ em tempo de execução

Alocação dinâmica

O programa requisita, **em tempo de execução**, um espaço de memória ao computador, o qual retorna um **endereço para o início** desse espaço (que é armazenado em um **ponteiro**)

Funções para alocação de memória (`stdlib.h`)

- `sizeof`
- `malloc`
- `calloc`
- `realloc`
- `free`

`sizeof`

Usada para saber quantos bytes ocupa **um único elemento** de um determinado tipo

```
printf("%d", sizeof(int)); // resultado: 4
```

malloc

```
void *malloc (unsigned int num);
```

Aloca uma quantidade **num** de bytes de memória e retorna um **ponteiro genérico (void*)** para o início do bloco

Programador **deve atribuir um tipo** ao ponteiro

Se houver erro a função retorna **NULL**

Alocação dinâmica

```c
int i;
int *p;

p = (int *) malloc(5 * sizeof(int));

for(i = 0; i < 5; i++) {
  p[i] = 10 + i;
  // *(p + i) = 10 + i;
}
```

⚠ Prestar atenção no tipo de dados e na quantidade

`calloc`

```
void *calloc (unsigned int num,
              unsigned int size);
```

Aloca uma quantidade **`num`** $\times$ **`size`** de bytes de memória e retorna um **ponteiro genérico (`void*`)** para o início do bloco

Inicializa todos os bits do bloco com **zero**

Alocação dinâmica

```c
int i;
int *p;

// p = (int *) malloc(5 * sizeof(int));
p = (int *) calloc(5, sizeof(int));

for(i = 0; i < 5; i++) {
  p[i] = 10 + i;
  // *(p + i) = 10 + i;
}
```

realloc

```
void *realloc (void *ptr,
               unsigned int num);
```

Modifica para **num** bytes o tamanho da memória previamente alocada e apontada por **ptr**

Alocação dinâmica

```c
int *p;
p = malloc(5 * sizeof(int));

//Diminui o tamanho do array
p = realloc(p, 3 * sizeof(int));

//Aumenta o tamanho do array
p = realloc(p, 10 * sizeof(int));
```

Quando o tamanho é maior, o bloco recém-alocado tem valor **indeterminado** (não ocorre inicialização)

`free`

```
void free (void *p);
```

A memória alocada dinamicamente (`malloc`, `calloc` ou `realloc`) **não é liberada automaticamente** pelo programa ⚠

Essa memória precisa ser liberada com a função **`free`**, que recebe o ponteiro para o início do bloco

Sempre libere a memória que não for mais utilizar e não deixe ponteiros "soltos"

```
free(p);
p = NULL;
```

Arrays multidimensionais

- Usando array unidimensional
- Usando ponteiro para ponteiro
- Usando ponteiro para ponteiro + array unidimensional