Prática 6
Introdução à programação C/C++
2017/2

Funções e Ponteiros - 2
Universidade Federal do Rio de Janeiro
Prof. Eduardo Mangeli

Nos próximos exercícios usaremos vetores com tamamnos variáveis. Para isso vamos usar funções para reserver memória e quando esta espaço não for mais necessários usar uma função para liberá-lo.

Essas funções são as seguintes: `malloc()`, `calloc()` e `free()`.

Protótipos

```c
#include <stdlib.h>

void *calloc (size_t nmemb, size_t size);
void *malloc (size_t size);
void free (void *ptr);
```

Exemplo

Como exemplo de uso destas funções considere o problema de reservar *n* posições para armazenar variáveis do tipo **int**. Para isto usamos o trecho de programa mostrado Listagem 1. Observe que após alocar o espaço foi usada a notação de vetores comuns.

```c
#include<stdlib.h>
#include<stdio.h>

int main (void){
  int i, n, *pvetor;
  float aux;

  /* define um valor para n, scanf ou n = */
  puts("Entre o tamanho de n.");
  scanf("%d", &n);

  /* aloca espaço na memória */
  pvetor = (int *) malloc (n * sizeof(int));
  if (!pvetor){
    puts("Sem memória.");
    return 1;
  }

  /* A PARTIR DE AGORA VOLTAMOS PARA VETORES COMUNS */
  /* aqui usamos pvetor, vamos ler o vetor */
  for (i=0; i<n; i++){
    printf("Entre elemento %d: ", i);
    scanf("%d", &pvetor[i]);
  }

  /* fazemos alguma coisa */
  aux = 0.0;
  for (i = 0; i < n; i++){
    aux += pvetor[i];
  }
  printf ("media: %f\n", aux/n);

  /* aqui não precisamos mais de pvetor */
  free(pvetor);
  return 0;
}
```

Listagem 1: Exemplo

Exercícios

Exercício 1. Execute o programa da Listagem 1.

Exercício 2. Agora vamos usar somente ponteiros. Execute o programa da Listagem 2.

```c
#include<stdlib.h>
#include<stdio.h>

int main (void){
  int i, n, *pvetor;
  float aux;

  /* define um valor para n, scanf ou n = */
  puts("Entre o tamanho de n.");
  scanf("%d", &n);

  /* aloca espaço na memória */
  pvetor = (int *) malloc (n * sizeof(int));
  if (!pvetor){
    puts("Sem memória.");
    return 1;
  }

  /* A PARTIR DE AGORA VOLTAMOS PARA VETORES COMUNS */
  /* aqui usamos pvetor, vamos ler o vetor */
  for (i=0; i<n; i++){
    printf("Entre elemento %d: ", i);
    scanf("%d", pvetor + i);
  }

  /* fazemos alguma coisa */
  aux = 0.0;
  for (i = 0; i < n; i++){
    aux += *(pvetor + i);
  }
  printf ("media: %f\n", aux/n);

  /* aqui não precisamos mais de pvetor */
  free(pvetor);
  return 0;
}
```

Listagem 2: Exemplo só com notação de ponteiros

Exercício 3. Escreva um programa que descubra qual é o maior segmento de memória que é possível reservar no computador.

Dica: Faça um loop que tente reservar espaço, se conseguir libere-o e tente reservar um espaço maior. Para liberar o espaço reservado use free(). Faça isto até verificar que o programa não consegue mais alocar memória.

Dica da dica: Ir aumentando de um em um byte vai fazer o programa demorar muito.

Exercício 4. Construa um programa para o registro de um anúncio de jornal. Nesse jornal o cliente paga por um sistema que considera o número de linhas do anúncio, a quantidade total de caracteres em cada linha, e a quantidade de dígitos no anúncio. Seu programa deve perguntar o número de linhas que o anúncio terá e a quantidade de caracteres para cada uma das linhas (pode ser diferente). Seu programa precisa usar a **notação de ponteiros** (e só de ponteiros) para alocar a memória para o anúncio, receber as linhas informadas pelo usuário, percorrer as linhas e informar ao final o **número de linhas**, **o número de caracteres em cada linha**, e a **quantidade total de dígitos** no anúncio.

A contagem dos dígitos nas linhas deve ser realizada por uma função desenvolvida especialmente para isso.

Para capturar as linhas informadas pelo usuário, use a função `fgets`. A Listagem 3 mostra um exemplo de uso dessa função.

```c
char *frase;
fgets(frase, 30, stdin); //lê até 30 caracteres (incluindo '\0') da entrada padrão
__fpurge(stdin); //limpa o buffer da entrada padrão e elimina os caracteres a mais
```

Listagem 3: Exemplo fgets