IP
Alocação Dinâmica

Hebert Coelho

Instituto de Informática
Universidade Federal de Goiás

Roteiro

- Alocação Dinâmica
- Exercícios
- Resolução de exercícios

sizeof

O operador unário sizeof calcula o tamanho de qualquer variável ou tipo Construído.

sizeof retorna um valor inteiro, assim seu resultado pode ser apresentado na tela com comando printf.

Exemplo:
int i;
printf("%d", sizeof(i));

- O printf acima, imprime o tamanho da variável inteira i.

Alocação Dinâmica

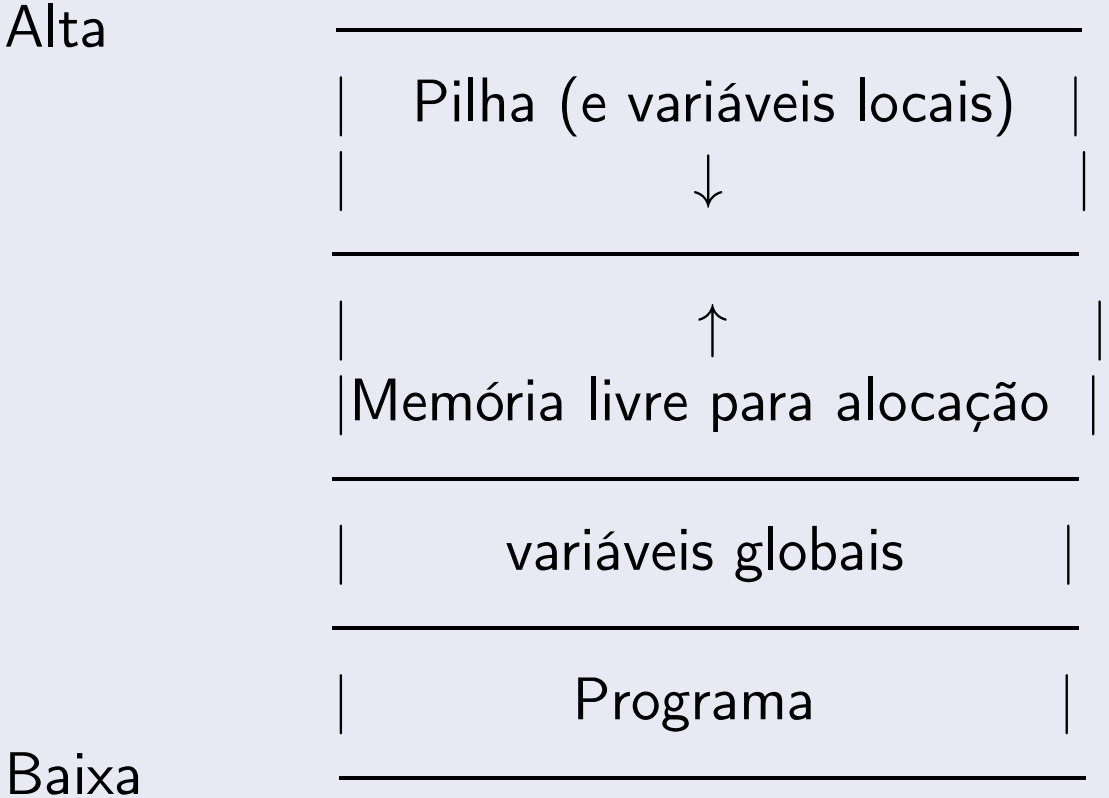
Existem 2 maneiras fundamentais de um programa em C armazenar informações na memória principal do computador.

- Variáveis locais e globais, incluindo matrizes e estruturas; Armazenamento fixo durante toda a execução do programa.

- Alocação Dinâmica
  Nesta maneira o programa pode obter espaço para armazenamento em tempo de execução.

Memória do sistema

O uso da memória de um programa em C



C ANSI

O padrão C ANSI especifica apenas quatro funções para o sistema de alocação dinâmica:

- calloc();
- malloc();
- free();
- realloc();

As funções de alocação dinâmica definidas pelo padrão C ANSI estão na biblioteca stdlib.h

calloc()

void *calloc(size_t num, size_t size);

- A função calloc() aloca uma quantidade de memória igual a num×size. Ou seja, calloc() aloca memória suficiente para uma matriz de num objetos de tamanho size;
- A função devolve um ponteiro para o primeiro byte da região alocada;
- Se não houver memória suficiente é devolvido um ponteiro nulo.

calloc - Exemplo de uso do calloc

```
1  #include<stdlib.h>
2  #include<stdio.h>
3
4  float *get_mem(void){
5       float *p;
6
7       p=calloc(100, sizeof(float));
8       if(!p){
9            printf("Erro de alocação — abortando.");
10           exit(1);
11           }
12      return p;
13 }
```

malloc()

void *malloc(size_t size);

- A função malloc() devolve um ponteiro para o primeiro byte de uma região de memória de tamanho size que foi alocada do heap;
- Se não houver memória suficiente é devolvido um ponteiro nulo;
- Sempre verificar se o valor devolvido não é um ponteiro nulo antes de utilizá-lo;

malloc - Exemplo de uso do malloc

```
1  #include<stdlib.h>
2  #include<stdio.h>
3
4  struct endereco{
5  char nome[40];
6  char rua[40];
7  char cidade[40];
8  char estado[2];
9  };
10
11 struct endereco *get_struct(void){
12       struct endereco *p;
13
14       if ((p = malloc(sizeof(struct endereco)))==NULL){
15            printf("Erro de alocação");
16            exit(1);
17       }
18       return p;
19 }
```

free()

void free(void *ptr);

- A função free() devolve ao heap a memória apontada por ptr, tornando a memória disponível para alocação futura.
- free() deve ser chamado apenas com um ponteiro que foi previamente alocado com as funções de alocação dinâmica.
- Um ponteiro inválido pode destruir o mecanismo de gerenciamento de memória;

free - Exemplo de uso do free

```
1  #include<stdlib.h>
2  #include<stdio.h>
3  #include<string.h>
4  #define tam 3
5  int main (){
6      char *str[tam];
7      int i;
8
9      for (i=0; i<tam; i++){
10         if ((str[i] = malloc(128))==NULL){
11             printf("Erro de alocação");
12             exit(1);
13         }
14          gets(str[i]);
15          puts(str[i]);
16     }
17     for (i=0; i<tam; i++) free(str[i]);
18     return 0;
19 }
```

realloc()

void *realloc(void *ptr, size_t size);

- A função realloc() modifica o tamanho da memória previamente alocada apontada por ptr para aquele especificado por size;
- O valor de size pode ser maior ou menor que o original;
- Um ponteiro para o bloco de memória é devolvido porque realloc() pode precisar mover o bloco para aumentar o seu tamanho;
- Se precisar mover o bloco, o conteúdo do bloco antigo é copiado no novo bloco, nenhuma informação é perdida.
- Se size é zero, a memória apontada por ptr é liberada.
- Se não há memória livre suficiente no heap é devolvido um ponteiro nulo e o bloco original é deixado inalterado.

realloc - Exemplo de uso do realloc

```
1  #include<stdlib.h>
2  #include<stdio.h>
3  #include<string.h>
4  int main (){
5      char *p;
6      if ((p = malloc(23))==NULL){
7              printf("Erro de alocação");
8              exit(1);
9      }
10     strcpy(p, "isso são 22 caracteres");
11
12     p = realloc(p,24);
13     if(!p){
14             printf("Erro de alocação");
15             exit(1);
16     }
17     strcat(p,".");
18     printf(p);
19     free(p);
20     return 0;
21 }
```

Exercícios

Faça um programa que leia um valor n e crie dinamicamente um vetor de n elementos e passe esse vetor para uma função que vai ler os elementos desse vetor. Depois, no programa principal, o vetor preenchido deve ser impresso. Além disso, antes de finalizar o programa, deve-se liberar a área de memória alocada.

Criar um tipo abstrato de dados que represente uma pessoa, contendo nome, data de nascimento e CPF. Crie uma variável que é um ponteiro para este TAD (no programa principal). Depois crie uma função que receba este ponteiro e preencha os dados da estrutura e também uma uma função que receba este ponteiro e imprima os dados da estrutura. Finalmente, faça a chamada a esta função na função principal.