Introdução - Alocação dinâmica de memória

Rodolfo Meneguette

Introdução

- Até agora -
  - Todas as variáveis que usamos em C ocupavam um espaço na memória que pertencia à sua função – stack

- Heap
  - Local da memória que utilizada para alocação dinâmica
- Caracteristicas:
  - Desordenada;
  - Reservar e liberar espaço a qualquer momento;
  - Tem menos restrições que a de stack

Como obter memória de forma dinâmica ?

Como fazer

Com ponteiros, podemos obter memória à medida que vamos precisando.

A alocação e a liberação de espaço de memória e feita por duas funções que estão presentes na biblioteca

stdlib.h

A duas funções são :

Malloc() - Alocar espaço de memória
Free() - Liberar espaço de memória

Alocaçãode memória

Função

malloc(int tamanho);

•Alocar um espaço na memória.
•Para determinarmos o tamanho necessário, devemos usar a função sizeof().
•A memória não e infinita.

•**Ponto Importante**
•O parâmetro passado para malloc.
- É o espaço total de memória que tem que ser alocado.

malloc(*<número de elementos> * sizeof(<tipo>)*);

- Exemplo

- **struct pos *vetor = malloc(100 * sizeof(struct pos));**

Exemplo

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

int main() {
  int *p;
  int i;

  p = (int*) malloc(10 * sizeof(int));

  for (i = 0; i < 10; i++)
    p[i] = i;
  for (i = 0; i < 10; i++)
   printf(”%d\n,” p[i]);

  return 0;
}
```

Liberação de memória

Função

free(tipo_ponteiro);

• Desaloca um bloco de memória alocado com malloc.

•Se você não desalocar a memória alocada com malloc, a memória não pode ser utilizada por outros programas do sistema, e eventualmente o computador pode ficar sem memória. Toda memória alocada é liberada quando o programa termina. Dizemos que programas que não desalocam a memória que consomem "vazam memória".

• Você não pode desalocar memória que não foi alocada. Você não pode desalocar o mesmo bloco de memória duas vezes.

Exemplo

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

int main() {
  int i;
  int *p;

    p =  malloc(100 * sizeof(int));
    printf("endereço alocado = 0x%x\n", (unsigned int) p);
    free(p);

  return 0;
}
```

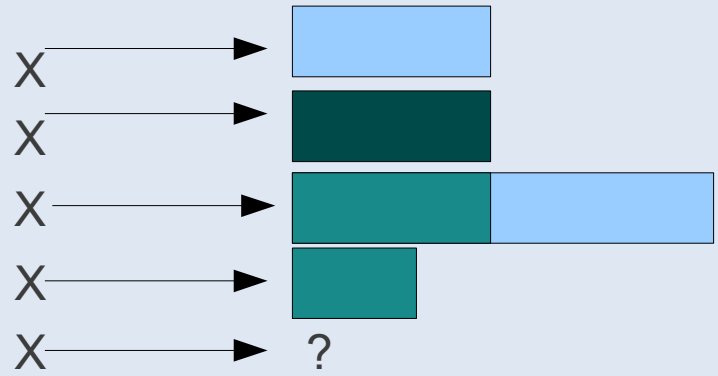
Outra Função

Função

realloc(void *apontador, int novo_tamanho);

- Faz um bloco já alocado crescer ou diminuir.

```
int main() {
 int *x,i; x
  x = (int *) malloc(4000*sizeof(int));
      for(i=0;i<4000;i++) x[i] = rand()%100;
 x = (int *) realloc(x, 8000*sizeof(int));
 x = (int *) realloc(x,2000*sizeof(int));
     free(x);
 return 0;
 }
```



Lidando com esses vetores alocados

- Copiar memória de um vetor pra outro

Funções

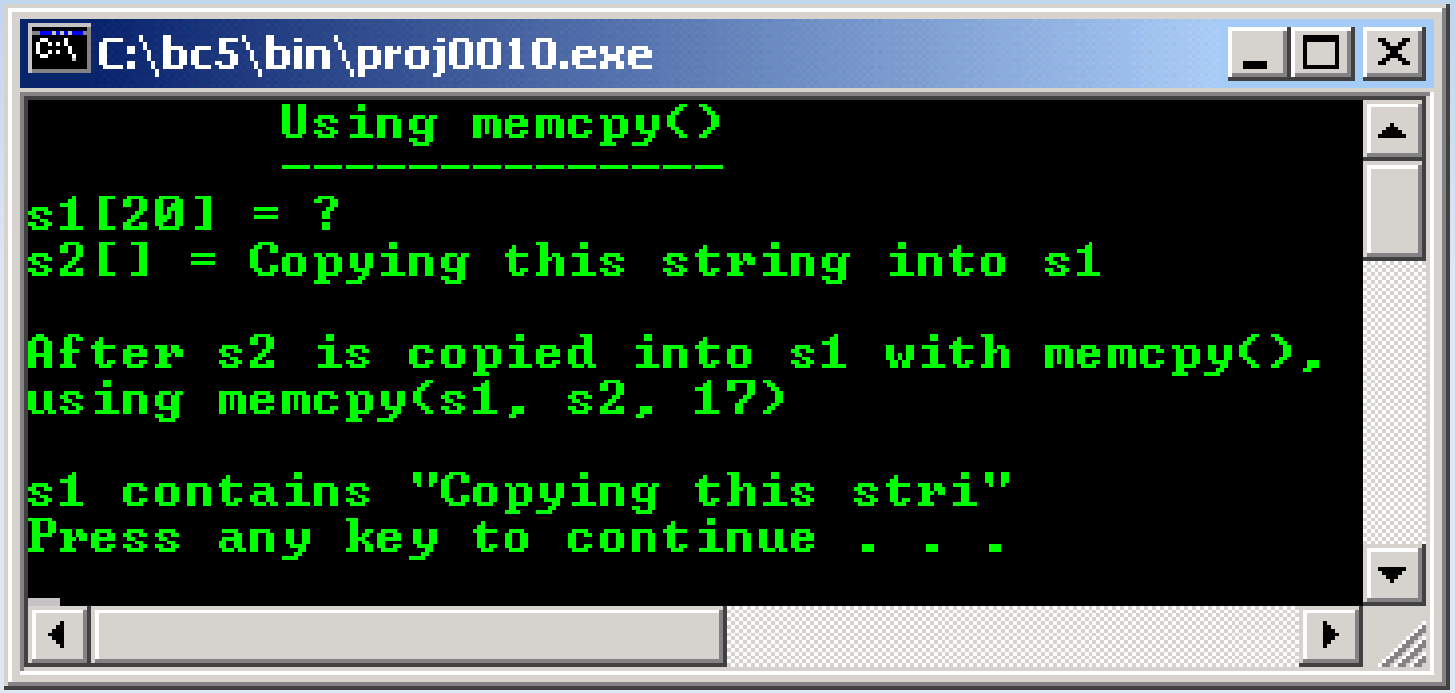
memcpy(a, b, n);

- A função copia os primeiros n bytes de b pra a.
- Se a e b são pedaços diferentes do mesmo vetor existe a função

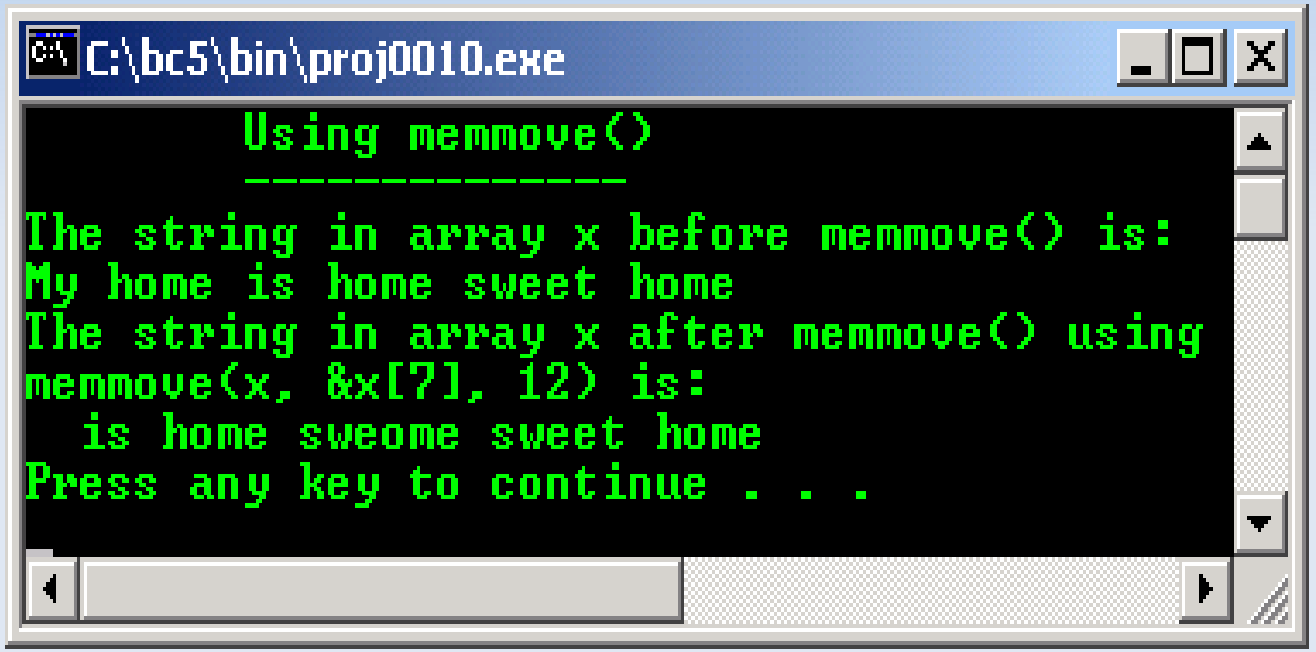
memmove(a, b, n);

Exemplo

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

int main()
{
    char s1[20], s2[] = "Copying this string into s1";
    memcpy(s1, s2, 17);
    printf("      Using memcpy()\n");
    printf("      ------------\n");
    printf("s1[20] = ?\n", s1);
    printf("s2[ ] = %s\n", s2);
    printf("\nAfter s2 is copied into s1 with memcpy(),\n");
    printf("using memcpy(s1, s2, 17)\n");
    printf("\ns1 contains \"%s\"\n", s1);
    return 0;
}
```


C:\bc5\bin\proj0010.exe
Using memcpy()
s1[20] = ?
s2[] = Copying this string into s1
After s2 is copied into s1 with memcpy(),
using memcpy(s1, s2, 17)
s1 contains "Copying this stri"
Press any key to continue . . .

```
#include <stdio.h>
#include <string.h>

int main()
{
   char x[ ] = "My home is home sweet home";

   printf("      Using memmove()\n");
   printf("      ----------\n");
   printf("The string in array x before memmove() is: \n%s", x);
   printf("\nThe string in array x after memmove() using \n");
   // memmove_s(void *dest, size_t sizeInBytes, const void *src, size_t
      //count); - a secure version
   printf("memmove(x, &x[7], 27) is:\n %s\n", memmove(x, &x[7], 27));
   return  0;
}
```


C:\bc5\bin\proj0010.exe
Using memmove()
The string in array x before memmove() is:
My home is home sweet home
The string in array x after memmove() using
memmove(x, &x[7], 12) is:
is home sweome sweet home
Press any key to continue . . .

Obrigado