Linguagem C

Alocação dinâmica de memória

g.p. telles

Alocação dinâmica de memória

- A alocação de memória é feita através de funções da biblioteca padrão.
- As funções são genéricas e retornam `void*`, que deve ser convertido implicita ou explicitamente.
- É responsabilidade do programador liberar a memória alocada dinamicamente.

stdlib.h

- Declara funções para alocar, realocar e liberar blocos de memória em tempo de execução.
- `void* malloc(size_t n);`
  Aloca n bytes consecutivamente na memória. Retorna um apontador void para a base da região alocada ou NULL se não for possível alocá-la.
- `void* calloc(size_t n, size_t t);`
  Aloca n elementos de t bytes consecutivamente na memória e inicializa todos os bits da região com zero. Retorna um apontador void para a base da região alocada ou NULL se não for possível alocá-la. Mais lenta que malloc.

stdlib.h

- `void free(void *ptr);`
  Recebe um apontador para uma região alocada anteriormente por calloc ou malloc.

Exemplos

- Alocação de memória para um inteiro:

```
int *p = malloc(sizeof(int));
p = calloc(1,sizeof(int));
free(p);
```

- Alocação de memória para um array de n inteiros:

```
int *p = malloc(n * sizeof(int));
p = calloc(n,sizeof(int));
free(p);
```

Exemplos

- Alocação de um array de apontadores para array, resultando em uma matriz n por n:

```
int i,n=10;
double **a;

a = malloc(n*sizeof(double*));
if (!a)
  return 0;

for (i=0; i<n; i++) {
  a[i] = malloc(n*sizeof(double));
  if (!a[i]) {
    while (i)
      free(a[--i]);
    free(a);
    return 0;
  }
}
```

stdlib.h

- `void* realloc(void *ptr, size_t n);`
  Modifica o tamanho do bloco apontado para n bytes. Retorna um apontador void para a base da região contígua alocada na memória. Ou retorna NULL se não for possível alocar, e o bloco permanece intacto. A nova região pode estar em uma posição diferente da memória. Preserva o conteúdo da memória até o mínimo entre os tamanhos da velha e da nova regiões.

Exemplos

- Duplicação de uma região:

```
int n;
double *A, *RA;

A = calloc(n,sizeof(double));
...
RA = realloc(A,2*n);
if (RA != NULL)
  ; /* foi, temos um array 2x maior */
else
  ; /* não foi, ainda temos A */
```