Aula 10 - Alocação de memória

Assuntos

- alocação de memória em C
- `malloc` e `free`

De vetores...

```c
#include <stdio.h>

int main() {
  int *p = {1, 2, 3, 4, 5};
}
```

Por que isso não funciona?

De vetores...

- `int v[] = {1, 2, 3, 4, 5};` aloca o espaço de 5 inteiros na memória;
- `int *p` aloca o espaço de apenas 1 inteiro, então não acessamos as outras posições
- Ou seja, precisamos alocar memória para ter mais do que um ponteiro de inteiro em um espaço contíguo na memória...

Alocação de memória (em C)

- alocação estática: criada e gerenciada em tempo de execução pelo próprio computador (o que fizemos até agora)
- alocação dinâmica: criada e gerenciada pelo programador (o que vamos fazer agora)

Alocação de memória (em C)

Em C temos duas áreas de memória:

- Stack:
  - variáveis estáticas com gerenciamento automático
  - funciona como uma pilha
- Heap:
  - gerenciado pelo programador

Alocação estática

- Quando uma variável é declarada, seu valor é armazenado na stack
- Quando o programa é encerrado, todos os valores na stack são apagados

```c
#include <stdio.h>

int main() {
    int i = 1; // coloca o valor de i na stack
    int j = 2; // coloca o valor de j na stack

    printf("%d %d\n", i, j); // le os valores da stack
} // limpa todos os valores da stack
```

Alocação dinâmica

- Necessário incluir `stdlib.h` e usar ponteiros
- Alocação e liberação de memória por chamada:
  - `malloc` : `void* malloc (sizeof(...))` aloca a memória e retorna um ponteiro para a primeira posição
  - `free` : `void free (void* p)` desaloca a memória do ponteiro `p`

Exemplo

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

int main() {
  int *p;                          // criamos um ponteiro.
  p = (int *) malloc(5*sizeof(int)); // alocamos memória.
  for (int i = 0; i < 5; i++)        // colocamos valores
    *(p + i) = i;                  // em cada posição.
  for (int i = 0; i < 5; i++)        // lemos os valores
    printf("%d ", *(p + i));         // de cada posição.
  free(p);                         // liberamos a memória.
  for (int i = 0; i < 5; i++)        // tentamos ler valores
    printf("%d ", *(p + i));         // de cada posição.
}
```

Exemplo

Neste caso:

```
int main(){

  int *p; // *p está na stack e aponta para o heap

  // stack aponta para heap
  p = (int *) malloc(5*sizeof(int));

  *(p + i) = i; // alteramos o valor no heap

  free(p); // liberamos o conteúdo na stack

} // liberamos os valores do heap
```