MC102 - Algoritmos e programação de computadores

Aula 24: Apontadores e Alocação Dinâmica de Memória

Variável

A uma variável `x` como a declarada abaixo:

```
int x = (100);
```

tem a ela associados:

- um nome (`x`);
- um endereço de memória ou referência (0xbfd267c4);
- um valor (100).

O operador address-of (&)

Retorna o endereço de uma variável:

```
int x = 100;
printf("valor de x = %d\n", x);
printf("endereço de x = 0x%x\n",
       (unsigned int) &x);
```

Veja o código `x.c`

Apontador

Um apontador é uma variável que pode armazenar endereços de outras variáveis.

```
int x;
int *ap_x; /* apontador para inteiros */

ap_x = &x; /* ap_x aponta para x */
```

Veja o código `ap_x.c`

Declaração de apontadores em C

Utilizamos o operador unário `*`

```
int    *ap_int;
char   *ap_char;
float  *ap_float;
double *ap_double;
```

Veja os códigos `apontadores.c` e `ap_tabela.c`

Declaração de apontadores em C

Para declarar vários apontadores em uma única linha:

```
int *ap1, *ap_2, *ap_3;
```

A declaração abaixo declara quantos inteiros e quantos apontadores para inteiro?

```
int *ap1, ap_ou_int1, ap_ou_int2;
```

Fazendo acesso aos valores das variáveis referenciadas

```
int x;
int ap_x = &x;

*ap_x = 3;
```

`*ap_x` pode ser usado em qualquer contexto que `x` seria.

Veja o código `valores.c`

Apontadores para estruturas

```
struct ponto {
  double x; double y;
};
typedef struct ponto Ponto;

\* ... *\
Ponto *ap_p;
ap_p->x = 4.0;
ap_p->y = 5.0;
```

Veja o código `ap_struct.c`

Passagem de parâmetros por valor

```
void nao_troca(int x, int y) {
  int aux;

  aux = x;
  x = y;
  y = aux;
}
```

Apenas uma cópia de x e y é passada para a função.

Passagem de apontadores

```
void troca(int *ap_x, int *ap_y) {
  int aux;

  aux = *ap_x;
  *ap_x = *ap_y;
  *ap_y = aux;
}
```

Veja o código: troca.c

Passagem de estruturas como parâmetros

- Seguem a mesma regra de variáveis simples;
- Veja o código `troca_struct.c`

Apontadores e Vetores

C permite manipulação de endereços via

- Indexação (v[4]) ou
- Aritmética de endereços (*(ap+4))

Veja os códigos `ap_e_vetores.c` e `cadeias.c`

Vetores de apontadores

```
int *vet_ap[5];
char *vet_cadeias[5];
```

- São vetores semelhantes aos vetores de tipos simples
- Veja o código `vet_cadeias.c`

Alocação dinâmica de memória

função malloc()

- Aloca um bloco consecutivo de bytes na memória e retorna o endereço deste bloco;
- Permite escrever programas mais flexíveis.
- Veja os códigos `malloc.c` e `nao_eh_infinita.c`

Liberação de memória

função free()

- Libera o uso de um bloco de memória;
- Futuras chamadas à função malloc poderão usar estes bytes.
- Veja o código `free.c`