Estruturas de Dados II

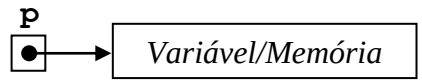
Professor: Francisco Assis da Silva

Ponteiro em C

Ponteiros (*pointers*) ou apontadores são variáveis que armazenam um endereço de memória (endereço de regiões de memória ou variáveis de tipo pré-definido, ex: int, float ou definido pelo programador, ex: TipoInfo).

Quando um ponteiro contém o endereço de uma variável, dizemos que o ponteiro está "apontando" para essa variável.

Como um ponteiro é apenas um endereço de memória, não é preciso muitos bytes para a sua definição, são usados 4 *bytes* para isto. Exemplo:



Neste caso, a variável ponteiro `p` do lado esquerdo está apontando para a variável/memória do lado direito. Internamente, a variável `p` contém o endereço da variável apontada.

Atribuindo um endereço para um ponteiro:

Vamos supor que temos uma variável num e queremos que o ponteiro `p` (ponteiro para int) aponte para esta variável. Para isso, usamos o operador `&` para obter o endereço de num, da seguinte forma:

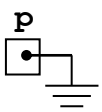
```
p = &num;
```

Exemplo:

```
int main()
{
   int num;
   int *p;
   num = 10;
   p = &num;
   printf("Endereco de p: %u \n", p);
   printf("Conteudo de p: %d", *p);
}
```

Um ponteiro também pode receber o conteúdo de outro ponteiro. Além disso, existe um endereço especial, denominado NULL (nulo ou 0), que serve para dizer que é um endereço nulo e que o ponteiro ainda não tem nenhum endereço válido.

```
p = NULL;
```



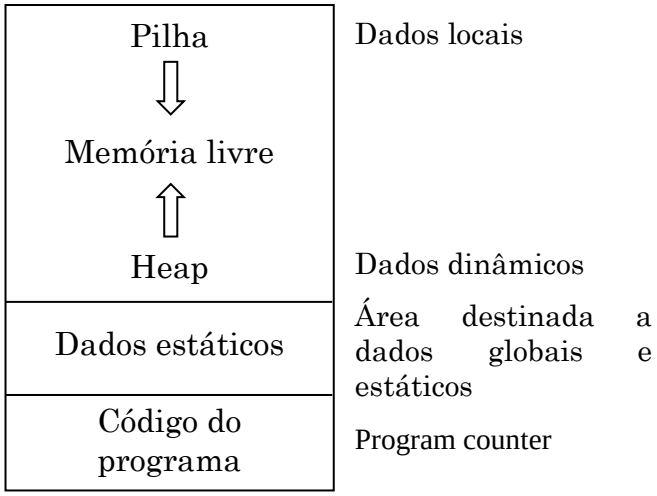
Trabalhando com a memória apontada pelo ponteiro:

Para acessar a memória que o ponteiro `p` está apontado, usamos o operador `*`, com a sintaxe `*p`. Naturalmente `p` deverá ter um endereço válido, isto é, `p` deve estar apontando para uma variável ou conteúdo de memória alocado no *heap*.

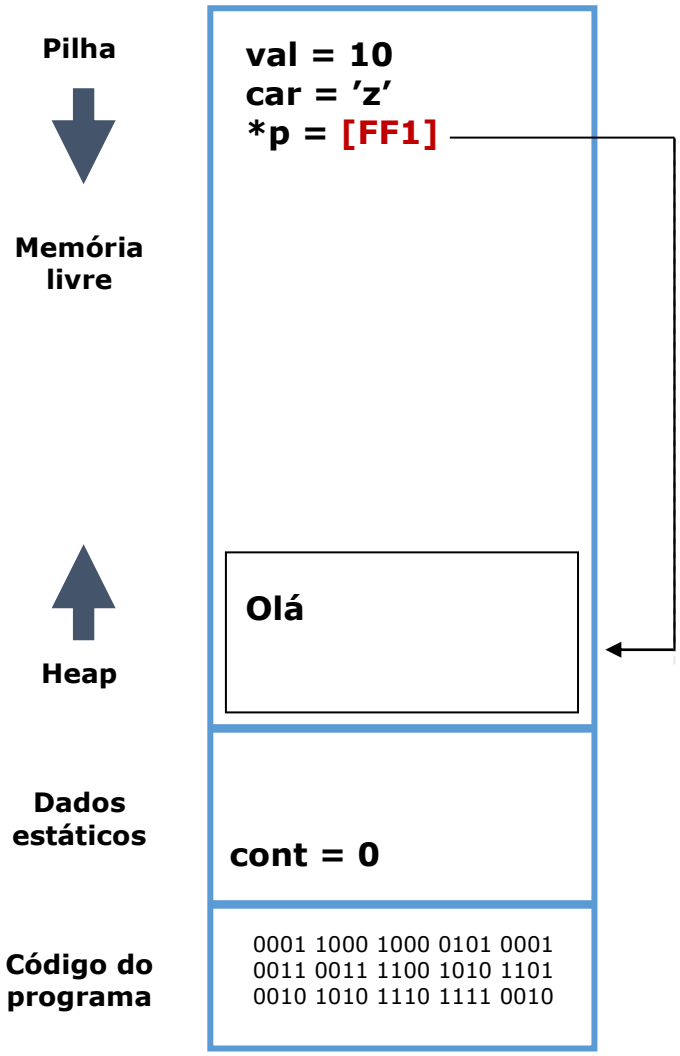
```
#include<stdlib.h>  //malloc
int main()
{
    int *p, *v;
    float *m;
    //v = new int();
    v = (int*)malloc(sizeof(int));
    m = new float();
    *v = 150;
    *m = 5.5;
    printf("%d \n",*v);
    p=v;
    printf("%d",*p);
}
```

Em C++, as áreas de memória dinâmicas são alocadas com o comando `new int()` ou em Ansi C `(int*)malloc(sizeof(int))`. Esse comando verifica se existe uma área de memória disponível e, se existir, aloca essa área e armazena seu endereço na variável `p`.
Para liberar uma área previamente alocada usa-se o comando `free(p)`.

Mapa de memória de um programa:



SEBESTA, R. W.; SANTOS, J. C. B. Conceitos de linguagens de programação. Editora Bookman.




```
char cont=0;

int main()
{
    int val = 10;
    char car;
    char *p;
    car = 'z';
    p = malloc(10);
    strcpy(p,"Olá");
    ...
    ...
}
```