Linguagem C: ponteiros para estruturas, alocação dinânica

Prof. Críston

Algoritmos e Programação

Ponteiro para estrutura

```
typedef struct
{
    char nome[100];
    int idade;
} pessoa;

main()
{
    pessoa joao;
    pessoa *p = &joao;

    strcpy(joao.nome, "joao da silva");
    joao.idade = 20;
    printf("%s, %d\n", (*p).nome, (*p).idade);

    (*p).idade = 18;
    printf("%s, %d\n", joao.nome, joao.idade);
}
```

Operador -> substitui (*p).

```
typedef struct
{
    char nome[100];
    int idade;
} pessoa;

main()
{
    pessoa joao;
    pessoa *p = &joao;

    strcpy(joao.nome, "joao da silva");
    joao.idade = 20;
    printf("%s, %d\n", p->nome, p->idade);

    p->idade = 18;
    printf("%s, %d\n", joao.nome, joao.idade);
}
```

Estruturas auto-referenciadas

```
typedef struct pessoa pessoa;
struct pessoa
{   char nome[100];
    int idade;
    pessoa *pai;
};
main()
{   pessoa joao, pedro;
    pessoa *p = &joao;

    strcpy(joao.nome, "joao da silva"); joao.idade = 20;
    strcpy(pedro.nome, "pedro da silva"); pedro.idade = 45;

    joao.pai = &pedro;
    printf("%s, %d\n", joao.pai->nome, joao.pai->idade);
    printf("%s, %d\n", p->pai->nome, p->pai->idade);
}
```

Se não fosse o operador -> ...

```
printf("%s, %d\n", (*(*p).pai).nome, (*(*p).pai).idade);
// no lugar de
printf("%s, %d\n", p->pai->nome, p->pai->idade);
```

Alocação dinâmica

- Permite solicitar memória em tempo de execução
- Função para alocar memória (stdlib.h):

```
malloc(num_bytes)
```

  - Retorna o endereço de memória da região alocada
  - Retorna zero se não for possível alocar
- Função para liberar a memória (stdlib.h):

```
free(endereco_regiao_alocada)
```

  - A região fica disponível para outras variáveis/alocações

Alocação dinâmica

```
main()
{
    // alocando um inteiro
    int *p = (int*) malloc(sizeof(int));
    if (p)
    {
        *p = 3;
        printf("%d\n", *p);
        free(p);
    }
}
```

Alocação dinâmica

```
typedef struct
{
    char nome[100];
    int idade;
} pessoa;

main()
{
    // alocando uma estrutura
    pessoa *p = (pessoa*) malloc(sizeof(pessoa));
    if (p)
    {
        p->idade = 3;
        printf("%d\n", p->idade);
        free(p);
    }
}
```

Alocação dinâmica

```
main()
{
    // alocando um vetor com 3 inteiros
    int *v = (int*) malloc( 3 * sizeof(int) );
    if (v)
    {
        v[0] = 10;
        v[1] = 20;
        v[2] = 30;
        printf("%d %d %d\n", v[0], v[1], v[2]);
        free(v);
    }
}
```

Modificando o tamanho da região alocada

- A função realloc (stdlib.h) permite modificar o tamanho de uma região alocada, conservando os dados previamente armazenados
- No caso de uma aumento de tamanho, realloc tenta utilizar bytes adjacentes à região já alocada
- Caso não seja possível, uma nova região é alocada, e os dados armazenados na antiga região são copiados

Modificando o tamanho da região alocada

```
main()
{
    int *v = (int*) malloc( 3 * sizeof(int) );
    if (v)
    {
        v[0] = 10; v[1] = 20; v[2] = 30;
        printf("%d %d %d\n", v[0], v[1], v[2]);

        v = (int*) realloc(v, 4 * sizeof(int) );
        if (v)
        {
            v[3] = 40;
            printf("%d %d %d %d\n", v[0], v[1], v[2], v[3]);
            free(v);
        }
    }
}
```

Layout da memória