NOTAS DE AULA 10 – Estruturas

1. Estruturas de Dados

A linguagem C/C++ fornece uma porção de tipos de variáveis: int, float, long, double, boolean, char... dentre outros. Mas todos esses tipos são chamados de "tipos simples" ou "tipos elementares" porque armazenam apenas uma informação: um número.

Algumas vezes precisamos representar tipos de dados mais complexos como um cliente ou um produto. Um produto, por exemplo, pode ser representado como um **id**, um **nome** e um **preço**. Como fazemos para criar uma lista de produtos?

Uma das alternativas seria criar uma lista "maluca", com vários vetores, cada um armazenando um tipo de informação diferente, mas o gerenciamento da mesma seria muito complicado!

Uma outra alternativa, mais simples, é **criar nosso próprio tipo de dado**: o tipo **produto**. Para fazer isso, usamos a instrução **struct**:

```
struct produto {
        int id;
        char nome[60];
        float preco;
};
```

Só isso? Só! Mas como usamos isso? Do mesmo jeito que qualquer outro tipo de dado. Por exemplo, se criamos uma pilha para inteiros assim:

```
int pilha[50];
int tpilha = -1;
```

Para criarmos uma pilha de produtos fazemos assim:

```
produto pilha[50];
int tpilha;
```

Observe como é a mesma coisa... só que o tipo de dado do vetor não é mais "int" e sim "produto". Ok, mas como é que eu acesso os dados de um produto? Por exemplo, como eu defino o "**id**" de um produto? Assim:

```
produto x;      // Declara o produto "x"
x.id = 10;      // Altera o id do produto "x"
```

O "." Indica que eu quero alterar o "id" que está dentro do produto "x". Mas... Como definir o nome do produto?

Bem, como o nome é um vetor, podemos fazer assim:

```
produto x;      // Declara o produto "x"
x.nome[0] = 'S';        // Guada primeira letra do nome
x.nome[1] = 'a';        // Guada segunda letra do nome
x.nome[2] = 'b';        // Guada terceira letra do nome
x.nome[3] = 'ã';        // Guada quarta letra do nome
x.nome[4] = 'o';        // Guada quinta letra do nome
x.nome[5] = '\0';       // Indica que o nome acabou
```

Como isso é muito chato e desagradável, os criadores do C/C++ elaboraram algumas funções para trabalhar com textos. Essas funções foram colocadas dentro da biblioteca **string.h**. Uma dessas funções é a **strcpy**, que copia um texto para dentro de uma variável:

```
produto x;
strcpy(x.nome,"Sabão");
```

Essa função funciona bem, mas para isso é preciso, no topo do código, acrescentar essa linha:

```
#include <string.h>
```

Experimente o programa abaixo:

```
#include <iostream>
#include <string.h>
using namespace std;
struct produto {
        int id;
        char nome[60];
        float preco;
};
int main(void) {
        produto p;
        p.id = 1;
        strcpy(p.nome, "Mouse");
        p.preco = 35.0;
        cout << p.id << ":";
        cout << p.nome << "-";
        cout << p.preco << endl;
}
```

Podemos ainda copiar o conteúdo de um dado complexo para outro, normalmente:

```
#include <iostream>
#include <string.h>
using namespace std;
struct produto {
        int id;
        char nome[60];
        float preco;
};
int main(void) {
        produto p, q;
        p.id = 1;
        strcpy(p.nome, “Mouse”);
        p.preco = 35.0;
        q = p;
        cout << q.id << “:”;
        cout << q.nome << “-“;
        cout << q.preco << endl;
}
```