Curso de C++ para Resolução de Problemas

Projeto de Ensino e Aprendizagem de Programação para Olimpíadas

Struct e Vector

- Struct
- Vector
  - Funções Membro
  - Vetores e Matrizes

Struct

Struct é um conjunto de elementos de dados agrupados sob o mesmo nome. Estes elementos de dados, conhecidos como membros, podem ter diferentes tipos e tamanhos. Uma **`struct`** pode ser declarada em C++ usando a seguinte sintaxe:

```
struct tipo_nome {
tipo_membro1 nome_membro1;
tipo_membro2 nome_membro2;
tipo_membro3 nome_membro3;
...
} nome_objeto;
```

Onde **`tipo_nome`** é um nome para o tipo de estrutura, **`nome_objeto`** pode ser um conjunto de identificadores válidos para objetos que tenham o tipo desta estrutura. Não é adequado utilizar a nomenclatura de variável para **`nome_objeto`** porque a estrutura, diferente da variável, é capaz de reter e armazenar mais de um valor ou expressão, isto é, refere-se a um conjunto de dados agrupados, ao invés de apenas um único elemento de informação. Também não é obrigatória a utilização do **`nome_objeto`** na etapa de definição da estrutura. Por fim, dentro de chaves **`{}`**, existe uma lista com os membros de dados, sendo que cada um é especificado com um tipo e um identificador válido como o seu nome.

Para exemplificar, definimos a seguir uma estrutura para representar um produto, que por sua vez contém os dados de nome, peso e preço de um produto qualquer.

```cpp
#include <iostream>
#include <iomanip>
#include <string>

using namespace std;

struct produto{
    string nome;
    int peso;
    double preco;
};

int main(){
    produto p;
    p.nome = "Ford Ka";
    p.peso = 905;
    p.preco = 30499.99;
    cout << "Nome: " << p.nome << endl;
    cout << "Peso: " << p.peso << " kg" << endl;
    cout << fixed << setprecision(2) << "Preço: R$ " << p.preco << endl;
    return 0;
}
```

Assim como declaramos uma variável do tipo `int` (inteiro), `char` (char), `float` (ponto flutuante) e etc., podemos usar a sintaxe aplicada em `produto p` para definir um objeto do tipo da estrutura produto. Para acessar cada um dos membros dessa estrutura, apenas usamos o **.** entre o nome do objeto criado e nome do membro desejado, qualquer que seja o tipo do membro definido. Isto quer dizer que as variáveis **nome**, **peso** e **preco** da `struct` **produto** são acessadas de forma semelhante, independente de uma ser do tipo `string` (cadeia de caracteres), outra do tipo `int` (inteiro) e a terceira do tipo `double` (ponto flutuante de precisão dupla).

● Vector

Vectors são contêineres sequenciais que representam matrizes que podem mudar de tamanho. Então, assim como matrizes (de uma ou mais dimensões), **vectors** usam locais de armazenamento contíguo para os seus elementos, o que significa que os seus elementos também podem ser acessados usando a mesma sintaxe de vetores e matrizes da linguagem C. Porém, são mais eficientes, já que o seu tamanho pode mudar dinamicamente e toda a questão de armazenamento e espaço de memória necessário é tratado de maneira automática e interna pelo contêiner.

- **Funções Membro**

```
#include <iostream>
#include <vector>

using namespace std;

int main(){
    vector<int> v1;
    vector<int> v2(5,-10);
    v2[4] = 24;
    return 0;
}
```

Para declarar um vector vazio aplicamos o padrão `vector<int> v1`. Em `vector<int> v2(5,-10)` declaramos um vector de tamanho `5` inicializado com valor `-10` em todas as suas posições, de `0` a `4`, pois os índices funcionam da mesma forma que os índices de vetores e matrizes utilizados na linguagem C. Assim, também é possível atribuir conteúdo para qualquer uma das posições do vector diretamente por meio do operador de atribuição `=` combinado ao operador `[]` para acessar o endereço de memória da posição especificada.

```
#include <iostream>
#include <vector>

using namespace std;

int main(){
    vector<int> v3(5,0);
    v3[4] = 24;
    cout << "O vector v3 tem tamanho " << v3.size() << endl;
    cout << "Seus elementos são: ";
    for(int i=0; i<(int)v3.size(); ++i)
        cout << v3[i] << " ";
    cout << endl;
    return 0;
}
```

Para saber o tamanho de um vector utilizamos a função membro **`size`**, acessada a partir do próprio objeto desejado. Esta função retorna o tamanho do vector como um número inteiro sem sinal, por isso a conversão `(int)v3.size()` foi utilizada dentro da estrutura de repetição `for`. Novamente, através do operador `[]` é possível acessar o conteúdo de cada posição válida do vector declarado.

- **Vetores e Matrizes**

A declaração de vetores por meio de vector segue o mesmo padrão visto no exemplos anteriores, sendo válida para qualquer tipo de dados ou estrutura definida usando C++. Já para criar matrizes, isto é, vetores multidimensionais, através de vector podemos utilizar algumas estratégias. No exemplo a seguir temos duas alternativas igualmente possíveis e corretas, uma que cria uma matriz de dimensões `M` por `N` do tipo `int` (inteiro) e outra que cria uma matriz de dimensões `2` por `2` de `struct ponto`, que contém as coordenadas `x` e `y` de pontos no plano cartesiano.

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

struct ponto{
    int x, y;
};

int main(){
    int M, N;
    cin >> M >> N;
    vector<vector<int> > m1(M);
    for(int i=0; i<M; i++)
        m1[i].resize(N);
    vector<vector<ponto> > m2(2, vector<ponto>(2));
    for(int i=0; i<2; i++)
        for(int j=0; j<2; j++)
            cin >> m2[i][j].x >> m2[i][j].y;
    for(int i=0; i<2; i++){
        for(int j=0; j<2; j++){
            cout << m2[i][j].x << "," << m2[i][j].y << " ";
        }
        cout << endl;
    }
    return 0;
}
```

PROGRAMATHON

Projeto de Ensino e Aprendizagem de Programação para Olimpíadas

EQUIPE

Reginaldo M. Kuroshu (Coordenador Geral do Projeto)
Alexandre Hild Aono
Bruno Bernardo de Moura
Danilo Gustavo Hansen Laboissiere
Diogo Augusto Hansen Laboissiere
Lucas de Alencar Barbosa
Marcos Castro de Souza
Nathan de Melo Cruz
Rodrigo de Farias Ramires
Thauany Moedano
Victor de Sá Nunes
Willian da Silva Zocolau

APOIO

Pró-Reitoria de Extensão (PROEX-UNIFESP)

REALIZAÇÃO

Instituto de Ciência e Tecnologia (ICT-UNIFESP)

CONTATO

programathon.unifesp@gmail.com

PARA MAIS INFORMAÇÕES ACESSE

programathon-unifesp.github.io