

Material de apoio para disciplina GCC224 - Introdução aos Algoritmos

Período: ERE 2020

Neste material você encontrará dicas e códigos discutidos passo-a-passo. Não pense que as sugestões de códigos a seguir são as melhores soluções. Sempre se pergunte como você faria. Se quiser, compartilhe conosco sua solução.

Link do vídeo do atendimento:

01/07/2020 (structs): https://youtu.be/6qvMmY8x_FU

Registros (struct)

Uma struct pode ser vista como um conjunto de variáveis sob um mesmo nome, sendo que cada uma delas pode ter qualquer tipo (ou o mesmo tipo). A ideia básica é criar apenas um tipo de dado que contenha vários membros, que nada mais são que outras variáveis (campos). Esse conceito de struct pode receber outros nomes em outras linguagens de programação, mas a ideia básica é a mesma.

Em um problema real, você deverá identificar o que pode vir a ser uma struct e quais seus campos, e o tipo de cada campo.

Exemplo de uma struct para representar um carro qualquer.
cor, ano e marca são *campos* de um carro.

```
struct carro {  
    string cor;  
    int ano;  
    string marca;  
}; //este ponto-vírgula é obrigatório
```

Para criar uma variável do tipo carro: `carro meuCarro;`

Para inserir dados para a variável meuCarro, precisamos acessar os campos do registro. Usamos o operador ponto (.) para referenciar os campos de um registro. No exemplo a seguir, estamos atribuindo valores para cada campo da variável meuCarro.

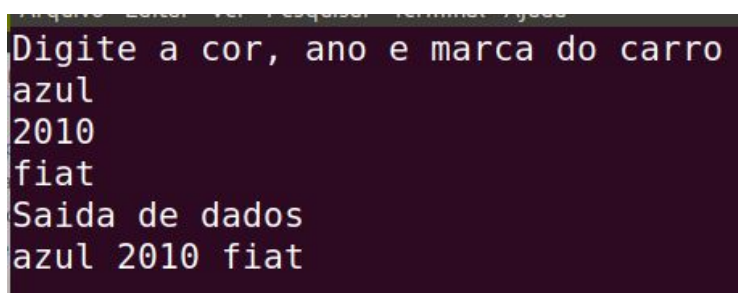
```
int main () {  
    //criando uma variável do tipo carro  
    carro meuCarro;
```

```

//lendo os dados para carro
cout<<"Digite a cor, ano e marca do carro "<<endl;
//identificador_variavel.identificador_campo
//ATENTE PARA O PONTO
cin>>meuCarro.cor;
cin>>meuCarro.ano;
cin>>meuCarro.marca;

cout<<"Saida de dados"<<endl;
cout<<meuCarro.cor<<" "<<meuCarro.ano<<" "<<meuCarro.marca<<endl;
return 0;
}

```



```

Digite a cor, ano e marca do carro
azul
2010
fiat
Saida de dados
azul 2010 fiat

```

Uma struct pode ter campos que são outras estruturas.

Um pneu tem aro e marca. Suponha que o pneu é um campo na struct carro. Veja no código a seguir, na definição da struct carro tem um campo *pneus* que é do tipo pneu. A struct pneu foi definida antes de ser utilizada.

```

#include <iostream>
using namespace std;

struct pneu {
    int aro;
    string marcaPneu;
};

struct carro {
    string cor;
    int ano;
    string marca;
    pneu pneus; //aqui temos um campo que é uma struct
};

int main () {
    //criando uma variável do tipo carro
    carro meuCarro;
}

```

```

//lendo os dados para carro
cout<<"Digite a cor, ano e marca do carro "<<endl;
cin>>meuCarro.cor;
cin>>meuCarro.ano;
cin>>meuCarro.marca;

//inserindo os dados do pneu
cout<<"Digite o tamanho do aro e marca do pneu "<<endl;
cin>>meuCarro.pneus.aro;
cin>>meuCarro.pneus.marcaPneu;





cout<<"Saida de dados"<<endl;
cout<<meuCarro.cor<<" "<<meuCarro.ano<<" "<<meuCarro.marca<<endl;
cout<<"Pneus "<<meuCarro.pneus.aro<<" "<<meuCarro.pneus.marcaPneu
<<endl;
return 0;
}

```

Arrays de estruturas

Podemos ter um array de registros. O tipo do array será do tipo da estrutura. Exemplo: carro meuCarro[4]; onde carro é a struct e meuCarro é variável que armazena até 4 registros do tipo carro.

Em cada posição do array haverá um registro do tipo carro. Ou seja, em cada posição, teremos os dados da cor, ano e marca de um carro. Observe que os dados são de diferentes tipos em cada posição, por isso dizemos que structs são estruturas heterogêneas; já int vetor[5] é uma estrutura homogênea, pois nesse caso, guarda somente inteiros.

0	1	2	3
			
preto 2018 fiat	azul 2010 volkswagem	branco 1990 ford	amarelo 1985 chevrolet

Para manipular um vetor que contém registros, utilizamos a notação já conhecida de vetores e após a posição, coloca-se um ponto (.) e o nome do campo que se deseja acessar: nomeDoVetor[posicao].nomeDoCampo

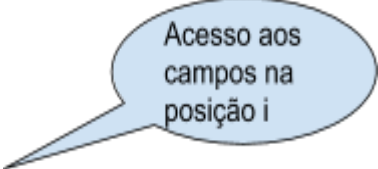
Código exemplo.

```
#include <iostream>
using namespace std;

struct carro {
    string cor;
    int ano;
    string marca;
};

int main () {
    //criando um vetor de carros
    carro meuCarro[4];
    int i;

    //entrada de dados
    for (i=0; i<4; i++) {
        cin>>meuCarro[i].cor;
        cin>>meuCarro[i].marca;
        cin>>meuCarro[i].ano;
    }
    return 0;
}
```



Acesso aos
campos na
posição i

Desafio: acrescente código no exemplo acima para imprimir os carros armazenados.

Estruturas que contêm arrays

Um registro pode conter campos que são arrays.

No exemplo do carro, suponha que para cada elemento armazenado, vamos guardar o preço do carro em 2 lojas. Para isso, o carro terá um vetor de float com tamanho 2.

Código exemplo:

```
#include <iostream>
using namespace std;

struct pneu {
    int aro;
    string marca;
};

struct carro {
```

```

    string cor;
    int ano;
    string marca;
    float preco[2]; //campo do tipo array
};

int main () {
    //criando um vetor de carros
    carro meuCarro[4];
    int i,j;

    //entrada de dados
    for (i=0; i<4; i++) {
        cin>>meuCarro[i].cor;
        cin>>meuCarro[i].marca;
        cin>>meuCarro[i].ano;

        //inserindo 2 precos para o carro i
        for (j=0;j<2;j++) {
            cin>>meuCarro[i].preco[j];
        }

    }

    return 0;
}

```

Busca por um registro

No roteiro de estudos anterior, tratamos de métodos de busca. São eles: busca sequencial e busca binária. Para a busca binária, é necessário que os dados estejam ordenados. Ordenação será o assunto abordado após registros. Quando os dados não estão ordenados, podemos fazer busca sequencial.

No exemplo a seguir, vamos pesquisar quantas ocorrências da marca “fiat” estão armazenadas no vetor de carros. Para isso, vamos utilizar um subprograma que retorna o total de ocorrências ou 0, caso não encontre nenhum carro da marca “fiat”. Observe que o método de busca recebe como parâmetros o vetor de carro, o tamanho do vetor e a chave de busca.

Código exemplo:

```

#include <iostream>
using namespace std;

struct carro {

```

```

        string cor;
        int ano;
        string marca;
    };

    int pesquisar (carro v[], int tamanho, string chave) {
        int i, total=0;
        for (i=0; i<tamanho; i++) {
            if (v[i].marca == chave ) {
                total++;
            }
        }

        return total;
    }

    int main () {
        carro meuCarro[4];
        int i;
        string busca;

        for (i=0; i<4; i++) {
            cin>>meuCarro[i].cor;
            cin>>meuCarro[i].marca;
            cin>>meuCarro[i].ano;
        }

        cout<<"Digite uma marca para buscar ";
        cin>>busca;

        cout<<"Total de ocorrencias "<<pesquisar(meuCarro,4,busca)<<endl;
        return 0;
    }

```

No exemplo a seguir, o subprograma **pesquisarCarroAno** retorna a posição de um registro no vetor ou -1 se não for encontrado. A busca é pelo ano do carro.

```

#include <iostream>
using namespace std;

struct carro {
    string cor;
    int ano;
    string marca;
};

int pesquisarCarroAno(carro v[], int tamanho, int chaveDeBusca) {
    int i=0, posicao=-1;

```

```

while (i<tamanho and v[i].ano != chaveDeBusca)
    i++;

if (i<tamanho)
    return i;

return posicao;
}

int main () {
    carro meuCarro[4];
    int i;
    int busca;

    //entrada de dados
    for (i=0; i<4; i++) {
        cin>>meuCarro[i].cor;
        cin>>meuCarro[i].marca;
        cin>>meuCarro[i].ano;
    }

    cout<<"Digite o ano para buscar ";
    cin>>busca;

    //i recebe o retorno da funcao e eh a posicao
    i=pesquisarCarroAno(meuCarro, 4, busca);
    if (i<4) {
        cout<<"Carro encontrado "<<meuCarro[i].marca<<endl;
    } else {
        cout<<"nao existem carros com esse ano";
    }

    return 0;
}

```



Ei você!
Não deixe de executar
os códigos deste
material!