BCC702 - Programação de Computadores II

Registros
Processamento de Arquivos

Profª Valéria de Carvalho Santos

UFOP

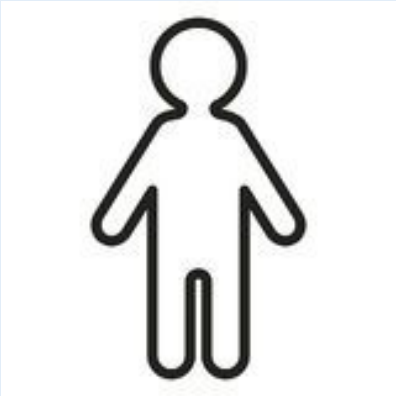
decom
departamento
de computação

Relembrando...

- Variáveis simples
  - int idade;
  - double altura;
- Vetores
  - int idades[10];
  - double alturas[5];
- Matrizes
  - int idades[2][10];

Exemplo

- Como representar as características de uma pessoa?

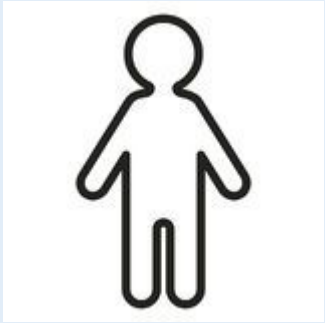


Registros

- São estruturas de dados capazes de agregar várias informações.
- São coleções de dados heterogêneos agrupados em um mesmo elemento de dados.
- Dessa forma, os programadores podem gerar novos tipos de dados, não se limitando à utilização dos tipos de dados primitivos.

Exemplo

- Como representar as características de uma pessoa?



- Nome
- CPF
- Altura
- Peso
- Cor do cabelo

Registro em C++

- Os registros em C++ são definidos pela utilização da palavra reservada *struct*, conforme a seguir:

```
struct nome_do_registro{
        tipo campo1;
        tipo campo2;
        ...
        tipo campoN;
};
```

Registro em C++

- A definição do registro através da palavra reservada *struct* apenas “cria um modelo” do registro
  - Define quais características compõem aquele registro
- Depois da definição do registro, uma ou mais variáveis do tipo *nome_do_registro* podem ser declaradas, sendo que os valores das variáveis estão relacionados aos campos do registro.

Registro em C++

- A partir da *struct* definida, pode-se considerar que existe um novo tipo de dado a ser utilizado.
- Cada informação contida em um registro é chamada de campo
- Os campos podem ser de diferentes tipos primitivos ou, ainda, podem representar outros registros.
- Por isso são chamadas variáveis compostas heterogêneas

Exemplo

- Declaração do registro pessoa:

```
struct pessoa{
        string nome;
        int idade;
        double salario;
};
```

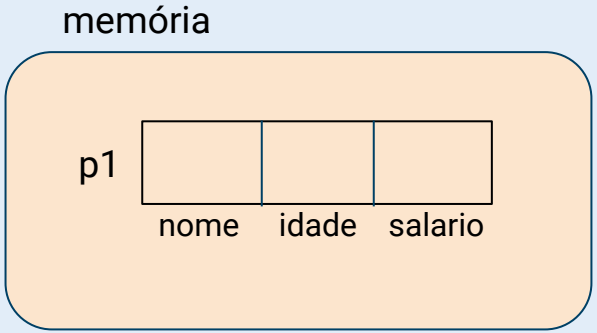
- Declaração de uma variável do tipo pessoa:

```
pessoa p1;
```

Exemplo

- Declaração do registro pessoa:

```
struct pessoa{
        string nome;
        int idade;
        double salario;
};
```



- Declaração de uma variável do tipo pessoa:

```
pessoa p1;
```

Registro

- Para acessar os campos de um registro, é necessário indicar o nome da variável e o nome do campo desejado, separados por ponto.

```
pessoa p1;
cout << “Digite o nome da pessoa: ”;
cin >> p1.nome;
```

Exemplo completo

```
int main() {
	pessoa p1;
	cout << "Digite nome, idade e salario da pessoa:";
	cin >> p1.nome;
	cin >> p1.idade;
	cin >> p1.salario;

	cout << p1.nome << " tem " << p1.idade << " anos" << endl;
	cout << p1.nome << " tem " << p1.salario << " de salario" << endl;
	return 0;
}
```

Registro

- O tipo do campo pode ser outro registro

```
struct data{
        int dia, mes, ano;
};
struct pessoa{
        string nome;
        int idade;
        double salario;
        data data_nasc;
};
```

Registro

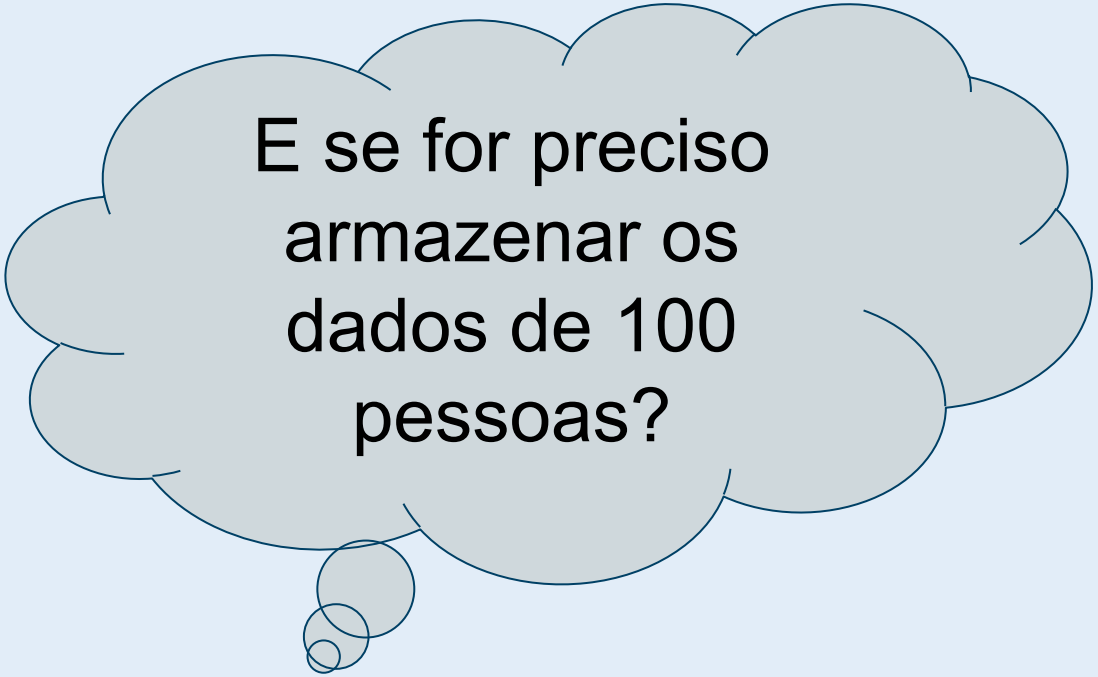
- O tipo do campo pode ser outro registro

```
int main(){
        pessoa p1;
        p1.data_nasc.dia=20;
        p1.data_nasc.mes=11;
        p1.data_nasc.ano=2001;
        cout << p1.data_nasc.dia << "/"<< p1.data_nasc.mes <<"/"
        << p1.data_nasc.ano << endl;
}
```

Exercício

- Defina um registro carro que contém os campos modelo, cor e ano de fabricação.

  Leia os dados de três carros e imprima o modelo do carro mais antigo.


E se for preciso armazenar os dados de 100 pessoas?

Vetores de Registros

- As estruturas vetores e registros podem ser combinadas para armazenar vários valores do tipo *registro* de uma vez
- Cada elemento do vetor será do tipo *registro*
- Como o vetor é uma estrutura homogênea, todos os elementos devem ser do mesmo tipo *registro*

Exemplo - vetores de registros

```
int main() {
	pessoa vet_pessoas[10];
	for(int i=0; i<10; i++){
		cin >> vet_pessoas[i].nome;
		cin >> vet_pessoas[i].idade;
		cin >> vet_pessoas[i].salario;
	}

	return 0;
}
```

Exercício

Elabore um programa que leia uma quantidade de carros e crie um vetor com os dados dos carros (cada carro tem um nome, ano e preço).

Leia um preço de referência e mostre as informações de todos os carros com preço menor (na ordem em que foram lidos).

Arquivos

- Até agora, todos os programas que fizemos usavam apenas a memória RAM do computador durante a execução.
- Portanto, ao finalizar o programa, nada permanecia guardado no computador.
- Mas a grande maioria dos programas de uso comum são úteis justamente por salvarem informações para uso posterior.

Arquivos

- O computador possui dois tipos de memória:
  - primária (ex: Memória RAM);
  - secundária (ex: Disco Rígido, pen drive).
- Tudo que fica na memória RAM é perdido quando o computador é desligado.
- A memória RAM é utilizada pelos programas durante sua execução.

Arquivos

- O computador possui dois tipos de memória:
  - primária (ex: Memória RAM);
  - secundária (ex: Disco Rígido, pen drive).

- Já o que é salvo no disco rígido (HD) permanece lá mesmo depois que o computador é desligado.

Manipulação de arquivos

- A manipulação de arquivos é feita por fluxos (streams)
- As funções de manipulação de arquivos estão na biblioteca fstream
- C++ possui três streams de manipulação de arquivos:
  - fstream
  - ifstream
  - ofstream

Manipulação de arquivos

- fstream (file stream)
  - usado tanto para ler quanto para escrever dados em um arquivo
- ifstream (input file stream)
  - usado para ler dados de um arquivo
- ofstream (output file stream)
  - usado para escrever dados em um arquivo

Escrita em arquivos

- Se o arquivo não existir, ele é criado
- Se o arquivo já existia, seu conteúdo é apagado
- É possível adicionar conteúdo ao final do arquivo, sem apagar, adicionando o parâmetro *ios:app* como parâmetro na abertura do arquivo

Exemplo - escrita em arquivo

```
#include <fstream>
using namespace std;
int main() {
	ofstream arquivo;
	arquivo.open(“meu_arquivo.txt”);
	arquivo << “Programação” << endl;
	arquivo.close();

	return 0;
}
```

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

int main() {
	ofstream arquivo(“meu_arquivo.txt”);
	string nome;
	int idade;
	cin >> nome >> idade;
	arquivo << nome << “ ” << idade << endl;
	arquivo.close();
	return 0;
}
```

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

int main() {
      ofstream arquivo(“meu_arquivo.txt”);
      string nome;
      int idade;
      cin >> nome >> idade;
      arquivo << nome << “ ” << idade << endl;
      arquivo.close();
      return 0;
}
```

Se o arquivo não existir, ele é criado.
Se o arquivo já existir, seu conteúdo é apagado.

Escrita de valor de variável

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

int main() {
      ofstream arquivo("meu_arquivo.txt", ios:app);
      string nome;
      int idade;
      cin >> nome >> idade;
      arquivo << nome << " " << idade << endl;
      arquivo.close();
      return 0;
}
```

Se o arquivo já existir, o que for escrito será adicionado ao final do arquivo.

Exemplo - leitura de arquivo

- Usa-se o *ifstream*
- Se o arquivo não existir, ele é criado
- Os dados são lidos de forma similar ao uso do *cin*
- Assim como no cin, é preciso dizer para qual variável os dados serão lidos
- É preciso saber a ordem e o tipo dos dados

Exemplo - leitura de arquivo

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
	ifstream arquivo(“meu_arquivo.txt”);
	string nome;
	int idade;
	if(arquivo){
		arquivo >> nome >> idade;
		cout << “Valores lidos do arquivo:” << endl;
		cout << nome << idade << endl;
		arquivo.close();
	}
	return 0;
}
```

Exemplo - leitura de arquivo

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
        ifstream arquivo(“meu_arquivo.txt”);
        string nome;
        int idade;
        if(arquivo){
                arquivo >> nome >> idade;
                cout << “Valores lidos do arquivo:” << endl;
                cout << nome << idade << endl;
                arquivo.close();
        }
        return 0;
}
```

Verifica se o arquivo foi aberto. Poderia ser *arquivo.is_open()*

Exemplo - leitura de arquivo

- Faça um programa que leia um arquivo contendo nomes e idades de várias pessoas. O nome do arquivo deve ser pedido ao usuário. Cada linha tem um nome e uma idade (o nome e a idade são separados por espaços). Ao final exiba o nome e a idade da pessoa mais velha.

```
int main() {
	string nome, nomeArquivo, nomeDoMaisVelho;
	int idade, maiorIdade = -1;
	cout << "Digite o nome do arquivo: ";
	cin >> nomeArquivo;
	ifstream arquivo(nomeArquivo);

	while (arquivo >> nome >> idade) {
		if (idade > maiorIdade) {
			maiorIdade = idade;
			nomeDoMaisVelho = nome;
		}
	}
	arquivo.close();
	cout << "O mais velho eh: " << nomeDoMaisVelho << ", ele tem " <<
	maiorIdade << " anos." << endl;
	return 0;
}
```