UnB Departamento de Ciência da Computação

2015/2

Estruturas de Dados

Prof. Guilherme N. Ramos

1 Registros

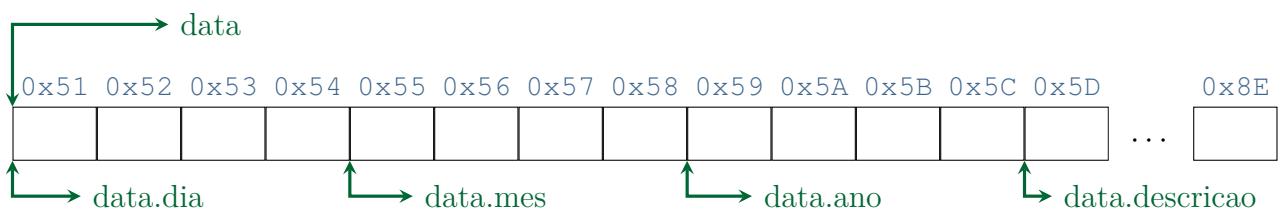
Vetores são conjuntos de dados homogêneos, mas muitas vezes deseja-se lidar com dados heterogêneos. Um registro é uma estrutura que pode armazenar diferentes tipos de dados em um bloco de memória.

```
1  Algoritmo LeFuncionários
2  Definições
3      funcionario : registro (nome, endereço : string;
4                                        sexo : caractere;
5                                      código : inteiro;
6                                     salário : real)
7  Variáveis
8      funcionários : vetor[1000] de funcionario
9      total, p100 : real
10 Início
11     total ← 0
12     Para i de 0 a 999 Faça
13         Leia(funcionários[i])
14         total ← total + funcionários[i].salário
15     FimPara
16     Para i de 0 a 999 Faça
17         p100 ← 100 * funcionários[i].salário / total
18         Escreva(funcionários[i].nome, "recebe ", p100, "%")
19     FimPara
20 Fim
```

Na linguagem `C`, o registro é uma coleção de uma ou mais variáveis, possivelmente de tipos diferentes, colocadas juntas sob um único nome para manipulação conveniente [1]. É definido pela palavra-chave `struct`, e o acesso a seus componentes pelo identificador e o caractere '`.`'.

0-data.c

```
1      struct {
2          int dia, mes, ano;
3          char descricao[50];
4      } data;
5
6      leia_string("Digite a descrição: ", data.descricao);
7      data.ano = leia_int("Digite o ano: ");
8      data.mes = leia_int("Digite o mes: ");
9      data.dia = leia_int("Digite o dia: ");
10
11     printf("%s:\n%02d/%02d/%04d\n", data.descricao,
12                                     data.dia,
13                                     data.mes,
14                                     data.ano);
15
```



A disposição dos elementos na memória é sequencial, *na mesma ordem em que são declarados* na definição do registro. A codificação pode ser facilitada pela definição um tipo (que deve ser adequadamente identificado). Um registro pode ser composto por outros registros, utilizado como argumento de função ou elemento de um vetor, ser manipulado por ponteiros, etc. (leia com atenção o arquivo `1-multiplos.c`).

Um aspecto interessante é que, por ser uma coleção arbitrária, um registro não pode ser comparado a outro diretamente [1], mas é bem simples definir uma ordem sobre eles:

`2-ordem.c`

```c
/* Indica ordem alfabética, utilizando a cronológica em caso
 * de igualdade. */
int alfabetica_cronologica(aniversario_t a, aniversario_t b){
    int em_ordem = alfabetica(a,b);
    return (em_ordem ? em_ordem : cronologica(a, b));
}
```

Em linguagem `C`, para evitar cópia de muitos bytes de memória, geralmente é mais eficiente se passar um ponteiro de um registro para uma função [1]. Muita atenção ao conceito de registro e de funções associadas a sua manipulação, são o primeiro passo rumo a Orientação a Objetos (na verdade, o conceito de registro em `C` é simplesmente uma variante do conceito de classe em `C++` [2])...

2 Arquivos

Um arquivo de computador é um espaço de memória cujas características dependem do tipo de dispositivo em que está armazenado. De forma abstrata, um arquivo é uma sequência de blocos [de bytes] de dados. Cada arquivo está associado a um nome, que, junto ao seu *caminho*, o identifica unicamente no sistema operacional.

As operações mais comuns são *leitura/escrita* de dados. A manipulação de memória é gerenciada pelo sistema, mas *é responsabilidade do programador iniciar e finalizar o processo.* Nem sempre as operações são bem sucedidas, é preciso ficar atento a erros.

Na linguagem `C`, os arquivos são manipulados por um ponteiro para o tipo `FILE`, que identifica a sequência de bytes do arquivo. O conjunto de dados é finalizado com o byte `EOF` (*end of file*). As principais funções para manipulação de arquivos estão na biblioteca `stdio.h`, mas lembre-se que antes de ser lido/gravado, um arquivo deve ser *aberto* [1].

Acesso: (fopen/fclose e fseek) A ação de abrir um arquivo permite ao usuário localizar e abrir um canal de comunicação com o meio de armazenamento [do arquivo] e pode inviabilizar alterações por outros processos; e fechar o arquivo interrompe este canal de comunicação.

E/S: (fread/fwrite e fscanf/fprintf) Uma vez aberto o canal, é relativamente simples ler/escrever uma quantidade determinada de bytes nele.

`0-binario.c`

```c
  FILE *fp = fopen(arquivo, "wb+"); /* "b" de "binário"... */
  if((fp == NULL) && (!fp) && (fp == 0)) {
      /* Os testes são equivalentes. */
```

```
4       printf("Não foi possível abrir \"%s\".\n", arquivo);
5       return EXIT_FAILURE;
6   }
7
8   /* Escreve no arquivo na ordem: string -> real -> inteiro. */
9   fwrite(str, sizeof(str), 1, fp);
10  fwrite(&d, sizeof(d), 1, fp);
11  fwrite(&i, sizeof(i), 1, fp);
12
13  /* O programador é responsável pelo arquivo. */
14  fclose(fp);
```

Como de costume, bastam o endereço inicial e a quantidade de bytes a ser manipulada para que tudo funcione como esperado. Mas nem tudo são flores. O arquivo existe? O usuário tem permissão para abrir o arquivo? O arquivo está disponível para acesso? Estas (entre outras) situações implicam que a instrução (linha 1) pode não ser bem sucedida...

Não se preocupe (muito), é fácil lidar com isto já que só há duas possibilidades a serem tratadas (linhas 2 a 6). O que fazer em caso de não ser possível abrir o arquivo depende da aplicação e do desenvolvedor; no entanto:

> *Uma vez aberto o arquivo, é responsabilidade do programador garantir que ele seja fechado corretamente.*

Portanto a linha 14 é essencial...

Os arquivos são armazenados na memória na forma binária, claro, mas para facilitar a interação com usuários humanos, é possível utilizar formas mais amigáveis (veja também os outros exemplos):

3-texto.c

```
1   const string str = "Algoritmos e Programação de Computadores";
2   const double d = 12.23;
3   const int i = 101;
4
5   fprintf(fp, "%s\n%lf\n\t%d", str, d, i);
```

Referências

[1] Brian W. Kernighan and Dennis M. Ritchie. *C: a linguagem de programação padrão ANSI*. Campus, Rio de Janeiro, 1989.

[2] Bjarne Stroustrup. *The C++ programming language*. Addison-Wesley, Reading, Mass., 3. ed., 20. print edition, 2004.

[3] Aaron M Tenenbaum, Yedidyah Langsam, and Moshe Augenstein. *Estruturas de dados usando C*. Pearson Makron Books, São Paulo (SP), 1995.