MAC–IME–USP

Carlos Eduardo Ferreira
Sala 108C Tel.: 3091 6079
e-mail cef@ime.usp.br
monitor Alexandre Freire
e-mail afreire@ime.usp.br
monitor João Miranda
e-mail joaomm88@gmail.com

MAC 122 – Princípios de Desenvolvimento de Algoritmos

Segundo semestre de 2009

Cadastro Acadêmico – **Entrega: 29 de novembro de 2009**

O objetivo deste exercício-programa é utilização de busca binária e manipulação de listas ligadas.

O cadastro acadêmico a ser construído e manipulado deve representar um conjunto de alunos, um conjunto de disciplinas, e um conjunto de pares `(A, D)`, cada par especificando que o aluno `A` está matriculado na disciplina `D`. O sistema deve permitir que sejam feitas as seguintes operações:

- matrícula de um aluno `A` numa disciplina `D`;
- cancelamento da matrícula de um aluno `A` numa disciplina `D`;
- relação de todos os alunos matriculados numa disciplina `D` (em ordem alfabética de nome);
- relação de todas as disciplinas em que o aluno `A` está matriculado (em ordem alfabética de sigla, por exemplo, FEP0240 antes de MAC0110).

Cada aluno será identificado por um número de 7 dígitos e um nome de no máximo 63 caracteres. Exemplo:

```
2290114  Joao da Silva
```

Cada disciplina terá uma sigla formada por 7 caracteres e um nome de no máximo 63 caracteres. Exemplo:

```
MAC0122  Principios de Desenvolvimento de Algoritmos
```

Representação das informações no cadastro acadêmico

A parte principal da estrutura de dados que representa o cadastro deve consistir de dois conjuntos de listas ligadas:

- para cada aluno `A`, uma lista ligada de todas as disciplinas em que `A` está matriculado, **ordenada** pelas siglas das disciplinas;
- para cada disciplina `D`, uma lista ligada de todos os alunos matriculados em `D`, **ordenada** pelos nomes dos alunos.

Estes dois conjuntos de listas ligadas devem compartilhar um conjunto de células do tipo `Aluno_Disciplina`, cujo formato é definido por:

```
typedef struct aluno             Aluno;
typedef struct disciplina        Disciplina;
typedef struct aluno_disciplina  Aluno_Disciplina;
typedef struct aluno_disciplina* pAluno_Disciplina;


struct aluno_disciplina {
  int ind_aluno;
  int ind_disc;
  pAluno_Disciplina prox_aluno;
  pAluno_Disciplina prox_disc;
};
```

onde

- `ind_aluno` é um índice do vetor de alunos `vet_a` (definido abaixo);
- `ind_disc` é um índice do vetor de disciplinas `vet_d` (definido abaixo);
- `prox_aluno` é um apontador para uma célula do tipo `Aluno_Disciplina` (com o mesmo campo `ind_aluno`);
- `prox_disc` é um apontador para uma célula do tipo `Aluno_Disciplina` (com o mesmo campo `ind_disc`).

Para cada aluno `A`, matriculado numa disciplina `D`, teremos uma célula do tipo `Aluno_Disciplina`.

As **informações de cada aluno** devem ser armazenadas em uma estrutura com o seguinte formato:

```
struct aluno {
  int               nusp;
  char              nome[64];
  pAluno_Disciplina inicio;
};
```

onde

- `nusp` é o número USP do aluno;
- `nome` é o nome do aluno;
- `inicio` é um apontador para a primeira célula da lista ligada das disciplinas em que o aluno está matriculado. (Cada célula dessa lista é do tipo `Aluno_Disciplina`.)

Considere o vetor `vet_a` que deverá ser alocado dinamicamente, definido por:

```
Aluno *vet_a;
```

Para cada índice `j`, `vet_a[j]` é a estrutura do tipo `Aluno` que contém as informações de um aluno. Este vetor está ordenado pelo número USP do aluno (campo `nusp`).

As **informações de cada disciplina** devem ser armazenadas em uma estrutura com o seguinte formato:

```
struct disciplina  {
  char              sigla[8];
  char              nome[64];
  pAluno_Disciplina inicio;
};
```

onde

- `sigla` é a sigla da disciplina;
- `nome` é o nome da disciplina;
- `inicio` é um apontador para a primeira célula da lista ligada dos alunos que estão matriculados na disciplina. (Cada célula dessa lista é do tipo `Aluno_Disciplina`.)

Considere o vetor `vet_d`, que deverá ser alocado dinamicamente, definido por:

```
Disciplina *vet_d;
```

Para cada índice `j`, `vet_d[j]` é a estrutura do tipo `Disciplina` que contém as informações de uma disciplina. Este vetor está ordenado pela sigla da disciplina (campo `sigla`).

Especificações do programa

Faça um programa em C que tenha as seguintes partes:

1. `Constrói o vetor vet_a`, a partir de um dado arquivo texto que contém informações de todos os alunos, ordenadas pelo número USP. A primeira linha desse arquivo contém o número total de alunos, e cada uma das linhas seguintes tem o seguinte formato:

   ```
   número nome
   ```

   correspondentes ao número USP e nome de um aluno, começando na coluna 1 e com exatamente um espaço entre os dois.

   Exemplo:

   ```
   2290114 Joao da Silva
   ```

2. `Constrói o vetor vet_d`, a partir de um dado arquivo texto que contém informações de todas as disciplinas, ordenadas pela sigla. A primeira linha desse arquivo contém o número total de disciplinas, e cada uma das linhas seguintes tem o formato:

`sigla nome`

correspondentes à sigla e ao nome de uma disciplina, começando na coluna 1 e com exatamente um espaço entre os dois.

Exemplo:

```
MAC0122 Principios de Desenvolvimento de Algoritmos
```

3. `Executa` cada uma das `instruções` contidas num dado arquivo texto. A primeira linha desse arquivo contém o número total de instruções. Cada uma das demais linhas desse arquivo contém uma instrução, que pode ser de um dos quatro tipos descritos a seguir, começando na coluna 1 e com exatamente um espaço separando cada informação.

   (a) `M num sigla`
   requer a matrícula do aluno com número USP `num` na disciplina `sigla`.

   (b) `C num sigla`
   requer o cancelamento da matrícula do aluno com número USP `num` na disciplina `sigla`.

   (c) `A sigla`
   requer a impressão da relação de alunos (números USP e nomes) da disciplina `sigla`, em ordem alfabética.

   (d) `D num`
   requer a impressão da relação de disciplinas (siglas e nomes) do aluno com número USP `num`, em ordem alfabética de sigla.

   Exemplo de arquivo:

```
4
M 1234567 MAC0122
C 1234567 MAC0122
A MAC0122
D 1234567
```

   Imprima mensagens adequadas quando `num` ou `sigla` ou o tipo da instrução ("M", "C", "A" ou "D") não forem válidos e, também, quando se deseja efetuar a matrícula numa disciplina em que o aluno já esteja matriculado, ou cancelar a matrícula numa disciplina em que o aluno não esteja matriculado.

4. Para buscar o número USP de um aluno no vetor `vet_a`, ou buscar a sigla de uma disciplina no vetor `vet_d`, utilize obrigatoriamente a `busca binária`.

Algumas sugestões

- Os nomes dos três arquivos de entrada e o de saída podem ser fornecidos pelo usuário através da linha de comando. Para isso considere o trecho de programa a seguir.

```
  FILE *fp_a; /* arquivo de alunos */
  FILE *fp_d; /* arquivo de disciplinas */
  FILE *fp_i; /* arquivo de instrucoes */
  FILE *fp_s; /* arquivo de saida */

  [...]

int
main(int argc, char *argv[])
     /* argc = numero de argumentos na linha de comando */
     /* argv = vetor de apontadores para strings contendo esses argumentos */
{

  [...]

  if (argc < 5)
    {
      fprintf(stdout,
      "Uso: %s <arq_alunos> <arq_disc> <arq_instr> <arq_saida>\n",
      argv[0]);
      return -1;
    }

  if ((fp_a = fopen(argv[1],"r")) == NULL)
    {
      fprintf(stderr,
      "%s: arquivo de alunos %s nao pode ser aberto.\n",
      argv[0], nome_arq_a);
      return -1;
    }

  if ((fp_d = fopen(argv[2],"r")) == NULL)
    {
      fprintf(stderr,
      "%s: arquivo de disciplinas %s nao pode ser aberto.\n",
              argv[0], nome_arq_d);
      return -1;
    }

  if ((fp_i = fopen(argv[3],"r")) == NULL)
    {
```

```
      fprintf(stderr,
      "%s: arquivo de instrucoes %s nao pode ser aberto.\n",
      argv[0], nome_arq_i);
      return -1;
    }

  if((fp_s = fopen(argv[4],"w")) == NULL)
    {
      fprintf(stderr,
      "%s: arquivo de saida %s nao pode ser criado.\n",
      argv[0], nome_arq_s);
      return -1;
    }

  [...]

  fclose(fp_a);
  fclose(fp_d);
  fclose(fp_i);
  fclose(fp_s);
}
```

- Na leitura de qualquer um dos três arquivos de entrada, você pode fazer a leitura de cada linha de um arquivo num vetor de caracteres, utilizando a função `fgets`. Para extrair as informações desse vetor, você pode utilizar as funções `strncpy` e `atoi`. Essas funções estão disponíveis na biblioteca de funções do C.
- Para comparar dois strings (siglas de disciplinas ou nomes de alunos), pode utilizar a função `strcmp` ou a função `strncmp`, também disponíveis na biblioteca de funções do C.
- Alguns exemplos de arquivos estão disponíveis em

  `http://www.ime.usp.br/~cef/mac122-09/ep4-dados/`