FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Princípios da Programação Procedimental

Gestor de Exames

2016/2017

Tomás Morgado de Carvalho Conceição
2012138578 | tmdcc@student.dei.uc.pt

Index

Estruturas de Dados

Estruturas

As estruturas em C são um tipo de dados definido pelo programador que lhe permite combinar vários tipos de dados diferentes, incluindo outras estruturas.
São então úteis para guardar registos, pois permite-nos criar um registo de um certo tipo definido pelo programador que por sua vez contém um número de atributos pertencentes a esse registo.

As estruturas criadas para este projecto foram:

Student

A estrutura com o nome *Student* foi criada para representar um aluno inscrito no sistema.
Os seus atributos são:

- id - O id único do estudante, representado por uma *string*.
- degree - O curso frequentado pelo aluno (ie: LEI), representado por uma *string*.
- year - O ano de estudos do aluno (1º, 2º, etc), representado por um inteiro.
- regime - O regime do aluno (normal, Erasmus, etc), representado por uma *string*.

```c
/*********************************************/
/* Data structure for Student type of data */
/*********************************************/
typedef struct student
{
    char* id; /* Student id */
    char* degree; /* Degree */
    int year; /* Year of study */
    int regime; /*  1- Normal
                    2- Student worker
                    3- Athlete
                    4- Associative Leader
                    5- Erasmus
                */
} Student;
```

Course

A estrutura com o nome *Course* foi criada para representar uma disciplina inserida no sistema.
Os seus atributos são:

- name - O nome da disciplina, representado por uma *string*.
- regent - O regente da cadeira, representado por uma *string*.

```c
/*****************************************/
/* Data structure for Course type of data */
/*****************************************/
typedef struct course
{
    char* name; /* Course name */
    char* regent; /* Teacher responsible for the course */

} Course;
```

Date

A estrutura com o nome *Date* foi criada para representar uma data no sistema, com por exemplo a data de um exame.
Os seus atributos são:

- year - O ano da data a representar, do tipo inteiro.
- month - O mês da data a representar, do tipo inteiro.
- day - O dia da data a representar, do tipo inteiro.
- hour - A hora da data a representar, do tipo inteiro.
- minute - O minuto da hora a representar, do tipo inteiro.

```c
/*************************************/
/* Data structure for Date type of data */
/*************************************/
typedef struct date
{
    int year;
    int month;
    int day;
    int hour;
    int minute;

} Date;
```

Exam

Esta é a estrutura mais complexa do projecto. Tal como o nome indica, permite representar um exame no sistema. Tal como descrito em seguida, esta estrutura tem como atributos todas as outras estruturas descritas acima.
Os seus atributos são:

- course - O curso a que o exame pertence, representado por uma estrutura do tipo *Course*.
- start_date - A data de início do exame, representada por uma estrutura do tipo *Data*.
- end_date - A data de fim do exame, representada por uma estrutura do tipo *Data*.
- rooms - Um *array* de salas reservadas para o exame, representado por um *array* de strings.
- n_rooms - O número de salas reservadas para o exame, representado por um inteiro.
- n_students - O número de alunos inscritos no exame, representado por um inteiro.
- period - A época do exame (Normal, Recurso, etc), representada por um inteiro.
- ***students -*** Uma lista ligada que contém os ponteiros que apontam para o endereço de memória de cada aluno inscrito no exame (descrito em pormenor no próximo capítulo).

```c
/*****************************************/
/* Data structure for Exam type of data */
/*****************************************/
typedef struct exam
{
    Course course; /* The exam points to its course */
    Date start_date; /* The exam has a start date associated */
    Date end_date; /* end_date = start_date + the duration of the exam */
    char* rooms[MAX_CHAR]; /* The exam has one or more rooms associated */
    int n_rooms; /* Number of rooms alocated for the exam */
    int period; /*  The exam can be:
                    NORMAL- Normal Period
                    SECOND- Second Period
                    SPECIAL- Special Period */
    struct student_pointers_node *students; /*  List of pointers to students.
                                                The exam has a list of enrolled students
                                            */

} Exam;
```

Listas Ligadas

As listas ligadas permitem-nos ter uma colecção de dados dinâmica.
Uma lista ligada é constituída por nós, cada um ligado entre si por um ponteiro. Na minha implementação de listas ligadas para este projecto, foram implementadas listas ligadas simples sem cabeçalho, em que cada nó aponta para o próximo através de um ponteiro, sendo que o último nó da lista aponta para *Null*.

As listas ligadas implementadas neste projecto são:

Student Node (Snode)

A estrutura de dados *Snode* representa um nó de uma lista ligada de alunos, contendo os seguintes campos:

- student - Uma estrutura de dados do tipo *Student*, que é o campo de informação de um nó do tipo *Snode*.
- next - Um ponteiro para outro nó do tipo *Snode*.

```c
/*************************************/
/* Node for a linked list of Students  */
/*************************************/
typedef struct student_node
{
    Student student; /* Student data type */
    struct student_node *next; /* Pointer to next student node */
} Snode;
```

Course Node (Cnode)

A estrutura de dados *Cnode* representa um nó de uma lista ligada de disciplinas (ou *Courses*), contendo os seguintes campos:

- course - Uma estrutura de dados do tipo *Course*, que é o campo de informação de um nó do tipo *Cnode*.
- next - Um ponteiro para outro nó do tipo *Cnode*.

```c
/*************************************/
/* Node for a linked list of Courses   */
/*************************************/
typedef struct course_node
{
    Course course; /* Course data type */
    struct course_node *next; /* Pointer to next course node */
} Cnode;
```

Exam Node (Enode)

A estrutura de dados *Enode* representa um nó de uma lista ligada de exames, contendo os seguintes campos:

- exam - Uma estrutura de dados do tipo *Exam*, que é o campo de informação de um nó do tipo *Enode*.
- next - Um ponteiro para o próximo nó do tipo *Enode*.

```c
/************************************/
/* Node for a linked list of Exams */
/************************************/
typedef struct exam_node
{
    Exam exam;
    struct exam_node *next;

}Enode;
```

Student Pointer Node (Spointer)

A estrutura de dados *Spointer* representa um nó de uma lista ligada de ponteiros para alunos, contendo os seguintes campos:

- student - Um ponteiro para uma estrutura de dados do tipo *Student*.
- next - Um ponteiro para o próximo nó do tipo *Spointer*.

```c
/*****************************************************/
/* Node for a linked list of pointers to students  */
/*****************************************************/
typedef struct student_pointers_node
{
    Student* student; /* Pointer to student structure */
    struct student_pointers_node *next; /* Pointer to next spointer node */
} Spointer;
```

Ficheiros

De modo a manter os dados guardados após terminar o programa, é necessário gravar os dados contidos nas listas em ficheiros.
Para tal, são usados três ficheiros:

courses_list.txt

Este ficheiro guarda os dados contidos na lista ligada de disciplinas. A informação neste ficheiro é guardada do seguinte modo: A primeira linha representa o nome da disciplina e a segunda linha representa o nome do regente da disciplina. Esta informação é depois repetida para cada disciplina guardada no ficheiro.

students_list.txt

Este ficheiro guarda os dados contidos na lista ligada de alunos. A informação neste ficheiro é guardada do seguinte modo: A primeira linha representa o ID único do aluno. A segunda linha representa o curso que o aluno representa. A terceira linha representa o ano em que o aluno está. A quarta linha representa o seu estatuto.
Esta informação é depois repetida para cada aluno guardado no ficheiro.

exams_list.txt

Este ficheiro guarda os dados contidos na lista ligada de exames. A informação é guardada no ficheiro do seguinte modo: A primeira linha representa o nome da disciplina a que o exame pertence, a segunda linha representa o responsável da cadeira e a terceira linha representa a época de exame. Após esta informação é guardada a informação da data e hora em que começa e termina o exame, com cada atributo da data (ver estrutura Data e estrutura Exam) em cada linha diferente. É guardada depois a informação relativa ao número de salas reservadas para o exame, seguida da descrição de cada sala, uma sala por cada linha do ficheiro. Por fim é guardada a informação relativa aos inscritos no exame, com uma linha a representar o número de inscritos no exame e após essa informação o ID de cada aluno inscrito no exame, um por linha.
Esta informação é depois repetida para cada exame guardado no ficheiro.

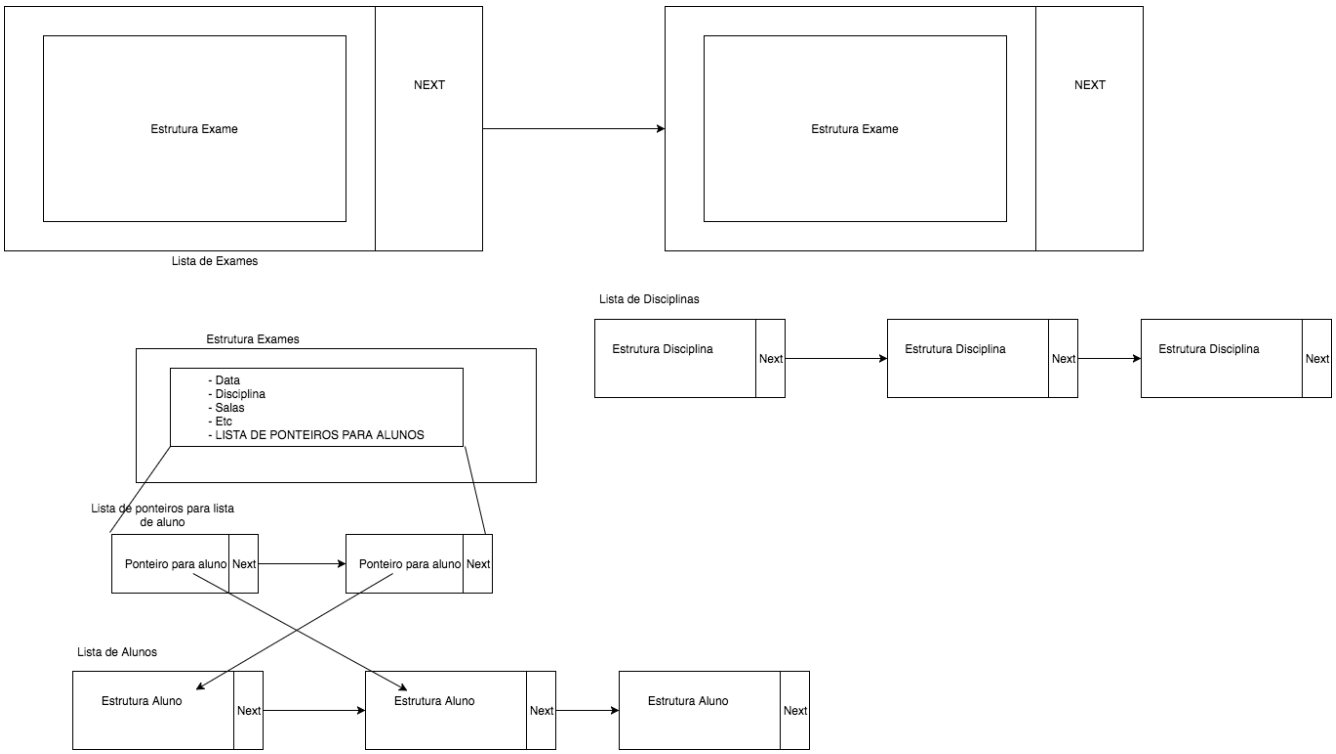
Como a lista ligada de exames contém também uma lista de ponteiros para alunos, de maneira a conseguir guardar os alunos inscritos num exame é necessário guardar alguma informação sobre essa lista em ficheiros. Para tal, é guardado o ID de cada aluno inscrito em cada exame no ficheiro de exames, sendo que ao iniciar o programa a lista ligada de exames é carregada

após a lista de alunos para que se possa voltar a criar a lista de ponteiros guardada na estrutura do exame.

Para além destes ficheiros, o utilizador pode criar outros ficheiros ao escolher guardar as listas de inscritos em exames quando consulta essa lista no programa. A lista é depois guardada num ficheiro com o seguinte formato: INSCRITOS_[DISCIPLINA]_[ÉPOCA].txt .

Esquema Estruturas de Dados

Um esquema simplificado de como as estruturas de dados se relacionam entre si pode ser visto em seguida:



Estrutura Geral do Programa

Após estarem definidos os tipos e estruturas de dados usados no projecto, a estrutura geral do Programa é a seguinte:

O programa ao iniciar tenta carregar os dados existentes nos ficheiros pré-definidos para as listas ligadas. Caso não haja dados num dos ficheiros, a lista ligada correspondente é iniciada a *NULL*.
É depois apresentado um Menu ao utilizador, permitindo-lhe escolher se quer aceder ao menu dos Alunos, das Disciplinas, dos Exames ou das Pesquisas.

No menu dos Alunos, o utilizador pode criar um novo aluno, listar os alunos existentes no sistema, editar um aluno ou apagar um aluno.

No menu das Disciplinas, o utilizador pode criar uma nova disciplina, listar as disciplinas existentes, editar uma disciplina e apagá-la.

No menu dos Exames, o utilizador pode criar um novo exame, listar os exames existentes no sistema, editar um exame ou apagá-lo.
Após listar os exames existentes o utilizador pode escolher adicionar um aluno a um exame, listar os alunos inscritos num determinado exame e verificar se algum exame não tem salas suficientes para o número de alunos inscritos.

No menu das Pesquisas, o utilizador pode pesquisar um aluno em particular, uma disciplina em particular, um exame em particular ou um aluno inscrito num determinado exame.

Execução do Programa

O ficheiro de entrega contém um *makefile* que automatiza o processo de compilação do código. Assim é simplesmente necessário escrever *make* no terminal com o terminal aberto no nível da pasta do projecto e um executável chamado *Project* será criado automaticamente.
Para iniciar o programa basta escrever *./Project* no terminal e o programa é iniciado.