

Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia de Telecomunicações e Informática
Métodos de Programação II

Trabalho Prático 1: Arrays, estruturas e ficheiros

Gestão de Salas num Complexo Pedagógico

Bruno Xavier Brás dos Santos A72122

Ana Catarina Gonçalves da Cunha A85492

Grupo 11

Guimarães
2018

Conteúdo

1	Introdução	2
2	Análise e Especificação	3
2.1	Descrição informal do problema	3
2.2	Especificação dos Requisitos	3
2.2.1	Dados	3
2.2.2	Inputs/Outputs do programa	3
3	Concepção/desenho da Resolução	4
3.1	Estruturas de Dados	5
4	Codificação e Testes	6
4.1	Alternativas, Decisões e Problemas de Implementação	6
4.2	Testes realizados e Resultados	6
5	Conclusão	8
6	Bibliografia	9
A	Código do Programa	10

Capítulo 1

Introdução

O presente relatório foi realizado para um projeto da unidade curricular de métodos de programação II do 1º ano do curso de MIETI da Universidade do Minho.

O projeto consiste em escolher um de entre quatro problemas. O nosso grupo decidiu pela escolha do problema número 4 - Gestão de salas de um complexo pedagógico. Optamos por este problema, porque era o que, à partida, trazia mais dificuldades na sua resolução e demonstrou ser o mais ambicioso. Para o projeto, tivemos que implementar um programa escrito na linguagem de programação C e elaborar um relatório referente ao mesmo.

O objetivo do programa é ajudar um gestor de um complexo pedagógico, responsável por fazer horários de uma escola, armazenando para cada sala do complexo a sua ocupação de hora a hora entre as 8h e as 20h em cada dia de 2af a 6af. Na secção ”Descrição informal do problema”, colocamos em detalhado o que o programa devia fazer.

Para a elaboração do presente relatório, optamos por utilizar o software L^AT_EX.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

O projeto consiste na escolha de um de quatro enunciados possíveis, implementar um programa na linguagem de programação C utilizando estruturas, ficheiros e arrays, e elaborar um relatório. Tal como explicito na "Introdução", foi feita a escolha do enunciado número 4.

No enunciado nº4, pretende-se ajudar o gestor de um complexo pedagógico, responsável por fazer horários de uma escola, armazenando para cada sala do complexo a sua ocupação de hora a hora entre as 8h e as 20h em cada dia de 2af a 6af. Por cada hora deverá ser registado se a sala está L(livre), ou O(ocupada) e nesse caso deve conter a sigla da UC marcada para essa hora e a sigla do curso. O objetivo é que o programa responda ao utilizador qual o estado de uma dada sala, numa dada hora e num dado dia (indicados pelo utilizador), ou que liste todas as salas livres a determinada hora de um dado dia (indicados pelo utilizador).

O programa deve ler o identificador da sala (uma simples sigla) e por cada dia útil da semana, ler a hora e a UC/Curso por cada marcação desse dia. Os dados introduzidos devem ser gravados num ficheiro binário para que possam ser recuperados numa sessão de trabalho posterior. Os resultados produzidos (estado dumha sala ou lista de salas livre) devem ser enviados para o standard output devidamente formatados.

2.2 Especificação dos Requisitos

2.2.1 Dados

Os principais dados a ter em conta da análise da descrição informal do problema são: um complexo que engloba várias salas, os cinco dias da semana úteis (2a feira, 3a feira, 4a feira, 5a feira e 6a feira), as 13 horas(8h - 20h) de cada dia útil da semana passíveis de serem ocupadas e um ficheiro em binário que guardará os dados introduzidos pelo utilizador.

Assim, por ordem decrescente de nível de complexidade temos: um complexo; várias salas; 5 dias da semana; 13 horas em cada dia da semana.

2.2.2 Inputs/Outputs do programa

Os inputs do programa correspondem ao identificador de uma sala, um dia da semana e uma hora. Os outputs correspondem aos resultados produzidos pelos inputs, que é o estado de uma dada sala ou uma lista de salas livres a uma dada hora e um dado dia.

Capítulo 3

Concepção/desenho da Resolução

O nosso objetivo para este projeto foi implementar um programa que, para além de satisfazer os requisitos do enunciado do problema, fosse um programa que se assemelhasse o mais possível de um programa possível de ser utilizado por qualquer utilizador. Para tal, para além de os outputs serem o estado de uma dada hora e uma lista de salas livres a uma dada hora, o nosso programa também exibe um mapa semanal de uma dada sala, permite inserir uma dada unidade curricular numa dada hora e possui uma interface dinâmica, intuitiva e de fácil manipulação dos dados inseridos e guardados num ficheiro binário.

O "esqueleto" ou a "base" do nosso programa corresponde às estruturas de dados que implementamos para guardar os dados lidos do utilizador e do ficheiro. Começamos pela unidade de nível de complexidade mais baixa - uma hora. No contexto do nosso programa, uma hora ou pode estar LIVRE ou OCUPADA. Para tal definimos um tipo denominado "estadoHora" que só pode tomar dois valores: OCUPADA ou LIVRE. Se uma hora estiver Ocupada, o programa tem que guardar a UC-Curso referente a essa hora. Para tal, definimos um novo tipo "HORA" que corresponde a uma estrutura que possui duas variáveis: uma do tipo "estadoHora" definido anteriormente e outra do tipo "char", que vai receber uma string com o nome da UC-Curso. De seguida, para cada dia útil, existe 13 horas disponíveis de serem ocupadas. Logo, decidimos declarar um vetor de inteiros de duas dimensões. A primeira com 5 elementos e a segunda com $20-8=13$ elementos. Ou seja, definimos um tipo "COMPLEXO", que corresponde ao vetor declarado anteriormente. Na função main, declaramos um vetor de dimensão 10 (corresponde às 10 salas, que pode ser alterado conforme o número de salas existentes no complexo) do tipo "COMPLEXO". Sendo assim, temos o nosso "esqueleto" definido: "COMPLEXO c[SALA][DIA SEMANA][HORA].(estadoHora ou UC)".

Depois de definir como armazenar os dados, elaboramos um Menu Inicial, onde o utilizador pode "navegar" e interagir com o programa. Depois de cada interação, os dados são guardados no ficheiro binário "salas".

Na figura 4.4 podemos ver as várias opções que o programa permite de serem realizadas. Na opção 5 - Defenicoes, existem duas opções, como pode ser visto na figura 4.1. Se o utilizador escolher a opção 1, é lido um ficheiro de texto com as instruções que o utilizador deve seguir para possuir uma interação máxima com o programa. A opção 2 permite restaurar o programa, ou seja, volta a inicializar o vetor de salas do tipo "COMPLEXO" com o valor LIVRE em todas as horas. Sendo de seguida guardado no ficheiro binário os respetivos valores.

3.1 Estruturas de Dados

O programa implementado, como explicito anteriormente, guarda os dados inseridos num ficheiro binário, lê a partir do ficheiro binário e mostra os dados no standard output devidamente formatados. A estrutura de dados descrita anteriormente está apresenta de forma codificada em C na figura 3.1.

```
13     typedef enum {LIVRE, OCUPADA}  estadoHora;
14
15     typedef struct hora{
16         estadoHora estado;
17         char UC[NOME_UC];
18     }HORA;
19
20     typedef HORA OcupacaoSalaDia[5][NUM_HORAS];
21
22     typedef OcupacaoSalaDia complexo ;
23
```

Figura 3.1: estruturas de dados usado no projeto

Declaração na função main(), do vetor do tipo ”COMPLEXO”.

```
41     int main(){
42         complexo c[NUM_SALAS]; // c é o complexo com NUM_SALAS. c[sala][dia][hora].LIVRE/OCUPADA
43         char *nomeFicheiro="salas.txt";
```

Figura 3.2: declaração do vetor c

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

No decorrer da implementação em linguagem C do problema, foi necessário criar algumas alternativas e tomar algumas decisões.

Foi necessário supor que no nosso complexo existe 10 salas. Para tal definimos uma constante simbólica que pode ser alterada sempre que decidir mudar o numero de salas. Em relação aos dias da semana úteis, normalmente eles são identificados por segunda feira, terça feira e assim sucessivamente. No nosso programa eles são identificados pelos números 2, 3, 4, 5 e 6, respetivamente. O que é algo de intuitivo. As salas são identificadas pelo identificados S_n , em que n é o número da sala. Por exemplo a sala 3 é identificada por S_3 .

Todas estas decisões, estão na opção 5 do Menu Iniciar e opção 1 do Menu Defenicoes como mostra a imagem em baixo.

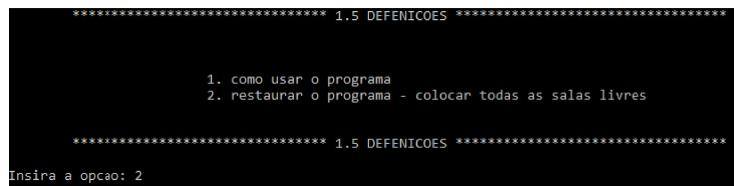


Figura 4.1: sub menu Defenicoes

4.2 Testes realizados e Resultados

No início, quando executo o programa, o programa pede ao utilizar se pretende instalar o programa. Instalar o programa consiste em iniciar o vetor do tipo COMPLEXO com o estado LIVRE em todas as horas e a criar o ficheiro -salas.txt-, em binário, onde será armazenado o estado de cada hora/dia/sala.

```
Se for a primeira vez que utiliza este programa, deve instalal-o.  
Prima <1> para instalar o programa. Prima <2> para ir para o Menu Inicial:
```

Figura 4.2: mensagem inicial

A interface do Menu Inicial encontra se na seguinte figura 4.4. Se inserirmos a opção 4 e de seguida colocarmos a sala 3, podemos ver o respetivo mapa da sala, como visualizamos na figura 4.3. Se voltarmos ao Menu Inicial, se escolhemos a opcao 2 podemos ver o estado de uma dada sala, como mostra a figura 4.5.

salas existentes no complexo: S0 S1 S2 S3 S4 S5 S6 S7 S8 S9						
1 MENU INICIAL --> 1.4 VER HORARIO SEMANAL POR SALA						
Insira a sala: S3						
SALA S3						
horas	SEGUNDA	TERCA	QUARTA	QUINTA	SEXTA	SABADO
8	LIVRE	SD-MIETI	LIVRE	LIVRE	LIVRE	
9	LIVRE	LIVRE	LIVRE	LIVRE	PP-MIEPOL	
10	LIVRE	LIVRE	LIVRE	LIVRE	LIVRE	
11	LIVRE	LIVRE	C-MIEBIOM	LIVRE	LIVRE	
12	LIVRE	LIVRE	LIVRE	LIVRE	LIVRE	
13	LIVRE	SC-MIEGSI	LIVRE	LIVRE	REOLOGIA-M	
14	LIVRE	LIVRE	LIVRE	LIVRE	LIVRE	
15	LIVRE	LIVRE	LIVRE	LIVRE	LIVRE	
16	LIVRE	LIVRE	C-MIEMAT	LIVRE	LIVRE	
17	LIVRE	LIVRE	LIVRE	LIVRE	LIVRE	
18	LIVRE	LIVRE	LIVRE	LIVRE	LIVRE	
19	SC-MIEGSI	LIVRE	C-MIEBIOM	LIVRE	LIVRE	
20	LIVRE	LIVRE	LIVRE	MP-MIETI	LIVRE	

Prima <M> para o Menu Inicial. Prima <0> para sair do programa: -

Figura 4.3: mapa semanal sala S3

```
***** 1 MENU INICIAL *****
salas existentes no complexo: S0 S1 S2 S3 S4 S5 S6 S7 S8 S9

1. Ver estado de uma sala
2. Ver salas livres a uma dada hora
3. Inserir Unidade Curricular
4. Ver mapa de horas por sala
5. Definicoes

0. Terminar o Programa
***** 1 MENU INICIAL *****

Insira a opcao:
```

Figura 4.4: Menu Inicial

```
salas existentes no complexo: S0 S1 S2 S3 S4 S5 S6 S7 S8 S9
1 MENU INICIAL --> 1.1 VER ESTADO DE UM [SALA] [DIA] [HORA]
Insira: (sala) (Dia da semana) (hora 8h-20h): s6 2 14
OCUPADA: MPII-MIETI

<1> para ver estado de outra sala. <2> para continuar: -
```

Figura 4.5: estado de uma sala

Capítulo 5

Conclusão

Através da realização deste projeto proposto no âmbito da unidade curricular de métodos de programação II do curso de MIETI, conseguimos compreender e aplicar os conhecimentos adquiridos nas aulas. Cumprimos os objetivos propostos, implementando um programa eficaz e que cumpria os requisitos propostos.

Decidimos, para além de cumprir os requisitos pedidos no enunciado, elaborar um programa que possuísse uma boa interface com um utilizador comum e fosse mais amplo nas opções que podia oferecer ao utilizador.

Um dos aspectos negativos deste projeto é a sua relação com a memória. Como a memória dinâmica não era um dos aspectos a serem considerados neste projeto, e como ainda não sentimos preparados em relação a este tema, decidimos não utilizar os seus princípios.

Os principais conceitos da linguagem C abordados neste projeto passaram pela utilização de estruturas, arrays e ficheiros binários.

Para a elaboração deste relatório, utilizamos o software L^AT_EX. No inicio foi trabalhoso aprender como o software funciona, mas com o estudo e a prática deste software, concluímos que este é uma excelente forma de produzir trabalhos científicos, relatórios e textos académicos com excelente qualidade.

Capítulo 6

Bibliografia

1. Damas, Luís (2001) Linguagem C, Lisboa: FCA
2. Slides colocados na BlackBoard Learn da unidade curricular
3. Sites sobre L^AT_EXdisponíveis em: <https://www.latex-project.org/>

Apêndice A

Código do Programa

Lista-se a seguir o código C do programa que foi desenvolvido.

```

1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<string.h>
4 #include<ctype.h>
5
6 #define NUM_SALAS 10
7 #define NUM_DIAS 5
8 #define NUM_HORAS 20-8+1
9 #define NOME_UC 10
10
11
12 // DEFINICAO DE TIPOS
13 typedef enum {LIVRE, OCUPADA} estadoHora;
14
15 typedef struct hora{
16     estadoHora estado;
17     char UC[NOME_UC];
18 }HORA;
19
20 typedef HORA OcupacaoSalaDia[5][NUM_HORAS];
21
22 typedef OcupacaoSalaDia complexo ;
23
24
25 // DECLARACAO DE FUNCOES
26 void imprimeSalas();
27 void iniciarComplexoLivre(complexo *c);
28 void criarFicheiro(char* nomeFich);
29 void colocarDadosFicheiro(complexo *c, char *nomeFicheiro);
30 void lerDadosFicheiro(complexo *c, char *nomeFicheiro);
31 void creditos();
32 int menu();
33 int esperar();
34 void verHorarioSemanalSala(complexo *c, int nSala);
35 void verSalasLivresHora(complexo *c);
36 void verEstadoHora(complexo *c);
37 void inserirUC(complexo *c);
38 int subMenu5defenicoes();
39
40 //FUNCAO MAIN
41 int main(){
42     complexo c[NUM_SALAS]; // c é o complexo com NUM_SALAS. c[sala][dia][hora].LIVRE/OCUPADA
43     char *nomeFicheiro="salas.txt";
44     int nOpcao, nSala, hora,n;
45     char sala[2+1];
46
47     printf("Se for a primeira vez que utiliza este programa, deve instala lo.\n\n");
48     printf("Prima <1> para instalar o programa. Prima <2> para ir para o Menu Inicial: ");
49     scanf(" %d", &n);
50     if(n==1){
51         iniciarComplexoLivre(c); // Inicia o complexo com o valor LIVRE. c[SALA] [DIA] [HORA]=LIVRE
52         criarFicheiro(nomeFicheiro); // cria ficheiro
53         colocarDadosFicheiro(c,nomeFicheiro);
54         system("cls");
55         printf("\nPROGRAMA INSTALADO COM SUCESSO!\n\n");
56         esperar();
57     }
58     else{
59         lerDadosFicheiro(c,nomeFicheiro);
60     }
61
62     while(1){
63         system("cls"); // Limpa tela do monitor
64         nOpcao=menu();
65         system("cls");
66

```

```

67     if(nOpcao==0) {
68         puts("Programa terminado com sucesso!");
69         puts("Obrigado.");
70         printf("\n\n");
71         creditos();
72         exit(10);
73     }
74
75     if(nOpcao==1) {
76         do{
77             imprimeSalas();
78             putchar('\n');
79             printf("1 MENU INICIAL --> 1.1 VER ESTADO DE UM [SALA] [DIA] [HORA]\n\n");
80             verEstadoHora(c);
81             printf("<1> para ver estado de outra sala. <2> para continuar: ");
82             scanf(" %d", &n);
83             system("cls");
84         }
85         while(n==1);
86     }
87
88     if(nOpcao==2) {
89         imprimeSalas();
90         putchar('\n');
91         printf("1 MENU INICIAL --> 1.2 VER SALAS LIVRES A UMA DADA HORA\n\n");
92         verSalasLivresHora(c);
93     }
94
95     if(nOpcao==3) {
96         do{
97             imprimeSalas();
98             printf("1 MENU INICIAL --> 1.3 INSERIR UNIDADE CURRICULAR\n\n");
99             putchar('\n');
100            inserirUC(c);
101            printf("<1> para adicionar nova unidade curricular. <2> para continuar: ");
102            scanf(" %d", &n);
103            system("cls");
104        }
105        while(n==1);
106    }
107
108    if(nOpcao==4) {
109        imprimeSalas();
110        putchar('\n');
111        printf("1 MENU INICIAL --> 1.4 VER HORARIO SEMANAL POR SALA\n\n");
112        printf("Insira a sala: "); scanf(" %s",sala);
113        nSala=(int)sala[1]-'0';
114        verHorarioSemanalSala(c,nSala);
115    }
116
117    if(nOpcao==5) {
118        n=subMenu5defenicoes();
119        if(n==1) {
120            system("cls");
121            FILE *fp;
122            char ch;
123            if((fp=fopen("defenicoes.txt","r"))==NULL) {
124                puts("Erro ao abrir ficheiro 'defenicoes'.\n");
125                continue;
126            }
127            while((ch=fgetc(fp))!=EOF)
128                putchar(ch);
129            fclose(fp);
130            putchar('\n');
131        }
132        if(n==2) {

```

```

133         iniciarComplexoLivre(c);
134         printf("PROGRAMA RESTAURADO COM SUCESSO.\n\n");
135     }
136 }
137
138 colocarDadosFicheiro(c,nomeFicheiro); // guarda dados no ficheiro
139
140 nOpcao=esperar();
141 if(nOpcao==0){
142     system("cls");           // limpa monitor
143     puts("Programa terminado com sucesso!");
144     puts("Obrigado\n");
145     creditos();
146     exit(10);
147 }
148 }
149 }
150
151
152 // DEFINICAO DAS FUNCOES UTILIZADAS
153 void iniciarComplexoLivre(complexo *c){      // Inicia o complexo com o valor LIVRE em todas as
horas/dias/salas
154     int i,j,k;
155     for(i=0;i<NUM_SALAS;i++) {
156         for(j=0;j<NUM_DIAS;j++) {
157             for(k=0;k<NUM_HORAS;k++) {
158                 c[i][j][k].estado=LIVRE;
159                 strcpy(c[i][j][k].UC,"LIVRE");
160             }
161         }
162     }
163 }
164
165 void criarFicheiro(char* nomeFich){          // Criar ficheiro se ainda nao existir
166     FILE *fp;
167     if((fp=fopen(nomeFich,"wb"))==NULL){        // Se nao criar, dā erro numero 1 - exit(1)
168         puts("Erro ao criar ficheiro.\n");
169         exit(1);
170     }
171     fclose(fp);
172 }
173
174 void colocarDadosFicheiro(complexo *c, char *nomeFicheiro){      // guarda os dados em modo binario no
ficheiro salas.txt
175     FILE *fp;
176     int i,j,k;
177     char ch;
178     if((fp=fopen(nomeFicheiro,"wb"))==NULL){
179         puts("Erro ao copiar dados para o ficheiro.\n");
180         exit(2);
181     }
182     for(i=0;i<NUM_SALAS;i++){
183         for(j=0;j<NUM_DIAS;j++){
184             for(k=0;k<NUM_HORAS;k++){
185                 if(c[i][j][k].estado==LIVRE){
186                     ch='L';
187                     fwrite(&ch,1,1,fp);
188                     fwrite(c[i][j][k].UC,NOME_UC,1,fp);
189                 }
190                 else{
191                     ch='O';
192                     fwrite(&ch,1,1,fp);
193                     fwrite(c[i][j][k].UC,NOME_UC,1,fp);
194                 }
195             }
196         }

```



```

326 }
327
328 void verEstadoHora(complexo *c) {
329     int hora,nSala,diaSemana;
330     char sala[2+1];
331     printf("Insira: (sala) (Dia da semana) (hora 8h-20h): ");
332     scanf("%s %d %d",sala,&diaSemana,&hora);
333     nSala=sala[1]-'0';
334     hora-=8;                                // indices variam entre 0 e 12
335     diaSemana-=2;
336     if(c[nSala][diaSemana][hora].estado==OCUPADA)
337         printf("OCUPADA: %s\n",c[nSala][diaSemana][hora].UC);
338     else
339         printf("LIVRE\n");
340     putchar('\n');
341 }
342
343 void inserirUC(complexo *c) {
344     char sala[2+1];
345     int nSala, diaSemana, hora;
346     printf("Insira: (sala) (Dia da semana) (hora 8h-20h): ");
347     scanf("%s %d %d",sala,&diaSemana,&hora);
348     nSala=sala[1]-'0';
349     hora-=8;                                // indices variam entre 0 e 12
350     diaSemana-=2;
351     if(c[nSala][diaSemana][hora].estado==OCUPADA){
352         printf("Hora ja ocupada: %s\n",c[nSala][diaSemana][hora].UC);
353     }
354     else{
355         printf("Insira a UC (UC-Curso): ");
356         scanf("%s",c[nSala][diaSemana][hora].UC);
357         c[nSala][diaSemana][hora].estado=OCUPADA;
358         printf("UC inserida com sucesso.\n");
359     }
360 }
361 putchar('\n');
362 }

```