Métodos de Programação II

Trabalho Prático 2

Leandro Alves
A82157

J. Eduardo Santos
A82350

David Machado
A82381

Maio de 2017

Conteúdo

Introdução

Este relatório tem como objetivo descrever o trabalho desenvolvido no âmbito da Unidade Curricular Métodos de Programação II, do Mestrado Integrado em Engenharia de Telecomunicações e Informática da Universidade do Minho.

Este trabalho foi desenvolvido através da linguagem de programação C com o auxílio a estruturas, ficheiros sequênciais, listas ligadas, apontadores e memória alocada dinamicamente, visando o desenvolvimento de um gestor de um armazém digital como foi proposto no enunciado.

Problema

O objetivo deste trabalho prático é fazer a gestão de um armazém.

O tema que escolhemos para o nosso trabalho foi um armazém de jogos chamado Steam Powered Storage onde as compras são feitas digitalmente e os jogos são recebidos na hora em vez de serem necessárias encomendas.

Cada jogo tem um ID único, um nome, um preço e um n.º de cópias em *stock*.

Na gestão do armazém, o utilizador deve poder carregar um jogo, atualiza-lo e apaga-lo.

Depois de criado o armazém, o cliente deve poder visualizar os dados dum jogo, ler o catálogo completo e comprar (no nosso trabalho também é possível vender) um jogo se houver *stock* suficiente e se houver, este deve ser atualizado e o valor total da compra/venda deve ser apresentado.

Também é necessário o armazenamento dos dados num ficheiro para que se possa sair do programa sem perder os mesmos.

Resolução

Inicialização do Código

1. Declaração de uma estrutura com os dados de cada jogo (ID, nome, preço e stock) e a recursividade da mesma. Esta estrutura é guardada num apontador.
2. Declaração das funções Upload, Search, Update, Delete, Search, Catalog e Buy.
3. Declaração de uma função *ScanfSub* que substituirá os *scanf* por *fgets* para permitir o uso funcional dos *getchar*.
4. Declaração de uma função *Insert* para armazenar ordenadamente os dados de cada jogo.

Função *main*

1. Ao abrir o programa, os dados do ficheiro serão lidos.
2. Passagem dos dados da função *Insert* para o apontador.
3. Menu principal com as funções numeradas. O utilizador deve inserir o número da função que pretende aceder. Se não houverem jogos, o utilizador será avisado que o armazém está vazio, portanto não poderá aceder à função (à exceção de *Upload*).
4. Ao sair do programa, os dados do apontador serão armazenados no ficheiro.

Função *Upload*

1. O utilizador escreve o ID do jogo que quer carregar.
2. Se não houver nenhum ID em uso igual ao introduzido, o utilizador introduzirá os dados do jogo (nome, preço, e stock), se houver, este será avisado e não poderá carregar o jogo.
3. Os dados do jogo carregado serão carregados na função *Insert*.

Função *Update*

1. O utilizador escreve o ID do jogo que quer atualizar.
2. Se o ID introduzido já existir, os dados do jogo correspondente serão apresentados e poderão ser alterados, se não existir, o utilizador será avisado e não poderá atualizar o jogo.
3. Os dados do jogo serão atualizados no apontador.

Função *Delete*

1. Na função *main*, o utilizador escreve o ID do jogo que quer apagar.
2. Se o ID introduzido já existir, os dados do jogo serão apagados, se não existir, o utilizador será avisado e não poderá apagar o jogo. A função é chamada recursivamente enquanto não encontrar o ID correspondente.
3. Os dados do jogo serão atualizados no apontador.

Função *Search*

1. O utilizador escreve o ID do jogo que quer ver.
2. Se o ID introduzido já existir, os dados do jogo correspondente serão apresentados, se não existir, o utilizador será avisado e não poderá ver os dados do jogo.
3. Os dados do jogo serão atualizados no apontador.

Função *Catalog*

Os dados de cada jogo serão apresentados, cada um via recursividade.

Função *Buy*

1. O utilizador escreve o ID do jogo que quer comprar ou vender.
2. Se o ID introduzido já existir, os dados do jogo correspondente serão apresentados e o utilizador escolherá quantoas cópias quer comprar ou vender, se não existir, o utilizador será avisado e não poderá comprar nem vender cópias do jogo.
3. No *Checkout*, o ID e o nome do jogo, o n.º de cópias e o preço total a pagar ou a receber serão apresentados.

Exemplo

```
1  (!) Hi and welcome to the Steam Powered Storage, after 1 month in development, it
       was worth the wait.
2  (!) Thanks and have fun!
3
4  [STEAM POWERED STORAGE]
5
6  [Storage management]
7  1. Upload a game;
8  2. Update a game's data;
9  3. Delete a game;
10
11 [Clients area]
12 4. Search for a game;
13 5. Catalog;
14 6. Buy / sell a game;
15
16 7. Exit.
17
18 > Please, choose a number: 5
19
20 [Catalog]
21
22 (!) The storage is empty! Please, upload a game if you want to see the catalog.
23
24 (!) Press 'Enter' to continue.
25
26 [STEAM POWERED STORAGE]
27
28 < menu principal >
29
30 > Please, choose a number: 1
31
32 [Uploading a game]
33
34 > ID: 10
35 > Name: Counter-Strike 1.6
36 > Price: $7.99
37 > Stock: 5000
38
39 (!) Counter-Strike 1.6 (10) was successfully uploaded to the Storage!
40
41 (!) Press 'Enter' to continue.
42
43 [STEAM POWERED STORAGE]
```

```
44
45 < menu principal >
46
47 > Please, choose a number: 1
48
49 [Uploading a game]
50
51 > ID: 10
52
53 (!) This ID is already being used by Counter-Strike 1.6! Please, choose another one.
54
55 (!) Press 'Enter' to continue.
56
57 [STEAM POWERED STORAGE]
58
59 < menu principal >
60
61 > Please, choose a number: 2
62
63 [Updating a game's data]
64
65 > ID: 10
66 > Name (Counter-Strike 1.6): Counter-Strike
67 > Price ($7.99): $9.99
68 > Stock: (5000): 10000
69
70 (!) Counter-Strike (10) was successfully updated!
71
72 (!) Press 'Enter' to continue.
73
74 [STEAM POWERED STORAGE]
75
76 < menu principal >
77
78 > Please, choose a number: 6
79
80 [Buying / selling a game]
81
82 > ID: 10
83 > Name: Counter-Strike
84 > Price: $9.99
85 > Copies (10000 in stock) (> 0 to buy, < 0 to sell, = 0 to cancel): 10001
86
87 (!) There are only 10000 copies of Counter-Strike in stock!
88 (!) If you need more copies, please update the game's data and refill the stock.
       Otherwise, please choose a lower number of copies.
89
90 (!) Press 'Enter' to continue.
91
92 [STEAM POWERED STORAGE]
93
94 < menu principal >
95
96 > Please, choose a number: 1
97
98 [Uploading a game]
99
100 > ID: 30000
```

```
> Name: Half-Life 3
> Price: $99.99
> Stock: 1000

(!) Half-Life 3 (30000) was successfully uploaded to the Storage!

(!) Press 'Enter' to continue.

[STEAM POWERED STORAGE]

< menu principal >

> Please, choose a number: 4

[Searching for a game]

> ID: 30000
> Name: Half-Life 3
> Price: $99.99
> Stock: 1000

(!) Press 'Enter' to continue.

[STEAM POWERED STORAGE]

< menu principal >

> Please, choose a number: 6

[Buying / selling a game]

> ID: 30000
> Name: Half-Life 3
> Price: $99.99
> Copies (1000 in stock) (> 0 to buy, < 0 to sell, = 0 to cancel): 1000

[Checkout]

> ID: 30000
> Name: Half-Life 3
> Copies to buy: 1000
> Total amount to pay: $99990.00

(!) Press 'Enter' to confirm.

(!) 1000 copies of Half-Life 3 were added to your inventory, enjoy them and have fun
    !

(!) Press 'Enter' to continue.

[STEAM POWERED STORAGE]

< menu principal >

> Please, choose a number: 6

[Catalog]

```

```
158 * ID: 10 | Name: Counter-Strike | Price: $9.99 | Stock: 10000
159 * ID: 30000 | Name: Half-Life 3 | Price: $99.99 | Stock: 0
160
161 (!) Press 'Enter' to continue.
162
163 [STEAM POWERED STORAGE]
164
165 < menu principal >
166
167 > Please, choose a number: 3
168
169 [Deleting a game]
170
171 > ID: 10
172
173 (!) Counter-Strike (10) was deleted!
174
175 (!) Press 'Enter' to continue.
176
177 [STEAM POWERED STORAGE]
178
179 < menu principal >
180
181 > Please, choose a number: 3
182
183 [Searching for a game]
184
185 > ID: 10
186
187 (!) ID not found!
188
189 (!) Press 'Enter' to continue.
190
191 [STEAM POWERED STORAGE]
192
193 < menu principal >
194
195 > Please, choose a number: 7
196
197 (!) Thank you for using the Steam Powered Storage!
```

Implementação

Código em *C*

```c
//  mp216TP2Gr06.c
//  TP2 | Armazem

//  Grupo 6 - PL1
//  Leandro Alves    | A82157
//  J. Eduardo Santos | A82350
//  David Machado    | A82381

//  Versao final

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX 100         //  Indice maximo das 'strings'

typedef struct storage     //  Estrutura com os dados de cada jogo
{
  int id;           //  ID do jogo
  char name[MAX];     //  Nome
  float price;        //  Preco
  int stock;          //  Quantidade
  struct storage *next; //  Estrutura recursiva
} info, *pointer;       //  Apontador da estrutura

//  Funcoes
pointer Upload();
void Update();
pointer Delete();
void Search();
void Catalog();
void Buy();

//  Funcao que substitui o 'scanf' por 'fgets'
float ScanfSub(char temp[MAX], float temp_temp)
{
  temp[strlen(temp) - 1] = '\0';  //  Remove a linha ('\n') da variavel
    temp_temp = atof(temp);       //  Converte para 'float' o valor de 'temp' e
        copia-o para uma nova variavel

  return temp_temp;
}
```

```
41 //  Funcao que armazena ordenadamente os dados do armazem
42 pointer Insert (int temp_id, char temp_name [], float temp_price, int temp_stock,
       pointer ptr)
43 {
44   pointer new;  //  Armazenamento dum novo jogo
45   new = (pointer) malloc (sizeof (info));
46
47   if (ptr)    //  Se o fim do armazem nao for encontrado
48   {
49     if (temp_id < ptr -> id)  //  Se o ID for menor que o ID anteriormente
           armazenado, aquele ficara por cima deste
50     {
51       new -> id = temp_id;
52       strcpy (new -> name, temp_name);
53       new -> price = temp_price;
54       new -> stock = temp_stock;
55       new -> next = ptr;
56       ptr = new;
57     }
58     else
59     {
60       //  Chamada recursiva para verificar o proximo jogo
61       ptr -> next = Insert (temp_id, temp_name, temp_price, temp_stock, ptr -> next)
             ;
62     }
63   }
64   else  //  Se o ID for maior do que todos os IDs, este ficara no fundo da lista
65   {
66     new -> id = temp_id;
67     strcpy (new -> name, temp_name);
68     new -> price = temp_price;
69     new -> stock = temp_stock;
70     new -> next = ptr;
71     ptr = new;
72   }
73   return ptr;
74 }
75 //  Funcao principal
76 int main ()
77 {
78   char temp [MAX];
79   int temp_id;
80   char temp_name [MAX];
81   float temp_price;
82   int temp_stock;
83   int number = 0;
84
85   pointer ptr = NULL;
86
87   FILE *storage;                    //  Apontador do ficheiro
88
89   storage = fopen ("steam_storage.txt", "r");   //  Leitura do ficheiro
90
91     if (storage)                    //  Se o armazem nao estiver vazio
92   {
93       fgets (temp, MAX, storage);
94       temp_id = atoi (temp);
95
```

```
96      while (!feof (storage))           //  Enquanto o armazenamento do ficheiro nao
            chegar ao fim
97      {
98        fgets (temp_name, MAX, storage);
99        temp_name [strlen (temp_name) - 1] = '\0';
100       fgets (temp, MAX, storage);
101         temp_price = atof (temp);
102         fgets (temp, MAX, storage);
103         temp_stock = atoi (temp);
104
105       //  Os dados da funcao sao inseridos na estrutura
106       ptr = Insert (temp_id, temp_name, temp_price, temp_stock, ptr);
107
108       fgets (temp, MAX, storage);
109         temp_id = atoi (temp);
110     }
111     fclose (storage);             //  Fim da leitura do ficheiro
112     }
113     system ("clear");               //  Limpeza do terminal
114     printf ("\n(!) Hi and welcome to the Steam Powered Storage, after 1 month in
            development, it was worth the wait.\n");
115     printf ("(!) Thanks and have fun!\n");
116
117   while (number != 7)               //  Enquanto o utilizador nao quiser sair do
        programa
118   {
119     printf ("\n[STEAM POWERED STORAGE]\n\n");
120
121     printf ("[Storage management]\n");
122     printf ("1. Upload a game;\n");
123     printf ("2. Update a game's data;\n");
124     printf ("3. Delete a game;\n\n");
125
126     printf ("[Clients area]\n");
127     printf ("4. Search for a game;\n");
128     printf ("5. Catalog;\n");
129     printf ("6. Buy / sell a game;\n\n");
130
131     printf ("7. Exit.\n\n");
132
133     printf ("> Please, choose a number: ");
134     fgets (temp, sizeof (temp), stdin);
135     number = ScanfSub (temp, number);
136     system ("clear");
137
138     //  O utilizador ao escolher um numero de 1 a 6, chamara a respetiva funcao
139     switch (number)
140     {
141       case 1: printf ("\n[Uploading a game]\n\n");
142                 ptr = Upload (ptr);
143       break;
144
145       case 2: printf ("\n[Updating a game's data]\n\n");
146           if (ptr)  //  Se o armazem nao estiver vazio
147           {
148                     Update (ptr, temp_id);
149                   }
150                   else
```

```
151             {
152               printf ("(!) The storage is empty! Please, upload a game if you want to
                      update its data.\n");
153             } break;
154
155         case 3: printf ("\n[Deleting a game]\n\n");
156             if (ptr)
157             {
158               printf ("> ID: ");
159                     fgets (temp, sizeof (temp), stdin);
160                     temp_id = ScanfSub (temp, temp_id);
161               ptr = Delete (ptr, temp_id);
162             }
163             else
164             {
165               printf ("(!) The storage is empty! Please, upload a game if you want to
                      delete it.\n");
166             } break;
167
168         case 4: printf ("\n[Searching for a game]\n\n");
169             if (ptr)
170             {
171               Search (ptr);
172             }
173                     else
174             {
175               printf ("(!) The storage is empty! Please, upload a game if you want to
                      search for it.\n");
176             } break;
177
178         case 5: printf ("\n[Catalog]\n\n");
179             if (ptr)
180             {
181               Catalog (ptr);
182             }
183             else
184             {
185               printf ("(!) The storage is empty! Please, upload a game if you want to
                      see the catalog.\n");
186             } break;
187
188         case 6: printf ("\n[Buying / selling a game]\n\n");
189             if (ptr)
190             {
191               Buy (ptr);
192             }
193             else
194             {
195               printf ("(!) The storage is empty! Please, upload a game if you want to
                      buy or sell it.\n");
196             } break;
197
198         default: if (number != 7)
199              {
200               printf ("\n[STEAM POWERED STORAGE]\n\n");
201               printf ("> Please, choose a number: %d\n\n", number);
202               printf ("(!) Invalid number!\n");
203              }  break;
```

```
204     }
205     if (number != 7)
206     {
207       printf ("\n(!) Press 'Enter' to continue.\n");
208           getchar ();
209           system ("clear");
210         }
211   }
212   printf ("\n[STEAM POWERED STORAGE]\n\n");
213   printf ("(!) Thank you for using the Steam Powered Storage!\n\n");
214
215   storage = fopen ("steam_storage.txt", "w"); //  Armazenamento dos dados no
          ficheiro
216   while (ptr)                  //  Enquanto o fim do armazem nao for encontrado
217   {
218     fprintf (storage, "%d\n%s\n%f\n%d\n", ptr -> id, ptr -> name, ptr -> price, ptr
          -> stock);
219     ptr = ptr -> next;             //  Armazenamento do proximo jogo
220   }
221   fclose (storage);              //  Fim do armazenamento dos dados no ficheiro
222
223   return 0; //  Fim do programa
224 }
225 //  Funcao para carregar jogos no armazem
226 pointer Upload (pointer ptr)
227 {
228   char temp [MAX];
229   int temp_id;
230   char temp_name [MAX];
231   float temp_price;
232   int temp_stock;
233   int check_id = 1;
234   pointer temp_ptr;
235   temp_ptr = ptr;
236
237   printf ("> ID: ");
238   fgets (temp, sizeof (temp), stdin);
239   temp_id = ScanfSub (temp, temp_id);
240
241   while (temp_ptr)           //  Enquanto o fim do armazem nao for encontrado
242   {
243     if (temp_ptr -> id == temp_id)  //  Se ja existir um ID igual ao introduzido
244     {
245       check_id = 0;
246       printf ("\n(!) This ID is already being used by %s! Please, choose another one
              .\n", temp_ptr -> name);
247     }
248     temp_ptr = temp_ptr -> next;  //  Verificacao do jogo seguinte
249   }
250   if (check_id == 1)
251   {
252     printf ("> Name: ");
253     fgets (temp_name, sizeof (temp_name), stdin);
254     temp_name [strlen (temp_name) - 1] = '\0';
255
256     printf ("> Price: $");
257     fgets (temp, sizeof (temp), stdin);
258     temp_price = ScanfSub (temp, temp_price);
```

```
259     if (temp_price < 0) //  Se o preco for negativo, o programa assumira como 0
260     {
261       temp_price = 0;
262     }
263     printf ("> Stock: ");
264     fgets (temp, sizeof (temp), stdin);
265     temp_stock = ScanfSub (temp, temp_stock);
266     if (temp_stock < 0) //  Se o stock for negativo, o programa assumira como 0
267     {
268       temp_stock = 0;
269     }
270     ptr = Insert (temp_id, temp_name, temp_price, temp_stock, ptr); //
            Armazenamento dos dados
271
272     system ("clear");
273     printf ("\n[Uploading a game]\n\n");
274     printf ("(!) %s (%d) was successfully uploaded to the Storage!\n", temp_name,
            temp_id);
275   }
276   return ptr;
277 }
278 //  Funcao para atualizar os dados dum jogo
279 void Update (pointer ptr)
280 {
281   char temp [MAX];
282   int temp_id;
283   char temp_name [MAX];
284   float temp_price;
285   int temp_stock;
286   int check_id = 0;
287
288   printf ("> ID: ");
289   fgets (temp, sizeof (temp), stdin);
290   temp_id = ScanfSub (temp, temp_id);
291
292   while (ptr)
293   {
294     if (temp_id == ptr -> id)
295     {
296       check_id = 1;
297
298       printf ("> Name (%s): ", ptr -> name);
299       fgets (temp_name, sizeof (temp), stdin);
300       temp_name [strlen (temp_name) - 1] = '\0';
301       strcpy (ptr -> name, temp_name);
302
303       printf ("> Price ($%.2f): $", ptr -> price);
304       fgets (temp, sizeof (temp), stdin);
305       temp_price = ScanfSub (temp, temp_price);
306       ptr -> price = temp_price;
307       if (temp_price < 0)
308       {
309         temp_price = 0;
310       }
311       printf ("> Stock: (%d): ", ptr -> stock);
312       fgets (temp, sizeof (temp), stdin);
313       temp_stock = ScanfSub (temp, temp_stock);
314       if (temp_stock < 0)
```

```
315         {
316           temp_stock = 0;
317         }
318         ptr -> stock = temp_stock;
319 
320         system ("clear");
321         printf ("\n[Updating a game's data]\n\n");
322         printf ("(!) %s (%d) was successfully updated!\n", ptr -> name, ptr -> id);
323       }
324       ptr = ptr -> next;  //  Verificacao do jogo seguinte
325     }
326     if (check_id == 0)    //  Se nao existir um ID igual ao introduzido
327     {
328       printf ("\n(!) ID not found!\n");
329     }
330 }
331 //  Funcao para apagar um jogo
332 pointer Delete (pointer ptr, int temp_id)
333 {
334   char temp [MAX];
335   int check_id = 0;
336 
337     if (ptr)  //  Se o fim do armazem nao tiver sido encontrado
338     {
339       check_id = 1;
340 
341     if (ptr -> id == temp_id)
342       {
343         system ("clear");
344         printf ("\n[Deleting a game]\n\n");
345       printf ("(!) %s (%d) has been deleted!\n", ptr -> name, ptr -> id);
346       ptr = ptr -> next;
347       }
348     else
349       {
350       ptr -> next = Delete (ptr -> next, temp_id);  //  Chamada recursiva para
              testar o jogo seguinte
351     }
352     }
353   if (check_id == 0)
354   {
355     printf ("\n(!) ID not found!\n");
356   }
357   return ptr;
358 }
359 //  Funcao para ler os dados dum jogo
360 void Search (pointer ptr)
361 {
362   char temp [MAX];
363   int temp_id;
364   int check_id = 0;
365 
366   printf ("> ID: ");
367   fgets (temp, sizeof (temp), stdin);
368   temp_id = ScanfSub (temp, temp_id);
369 
370   while (ptr)
371   {
```

```
372     if (ptr -> id == temp_id)
373     {
374       check_id = 1;
375
376       printf ("> Name: %s\n", ptr -> name);
377       if (ptr -> price != 0)
378       {
379         printf ("> Price: $%.2f\n", ptr -> price);
380       }
381       else
382       {
383         printf ("> Free to Play\n");
384       }
385       printf ("> Stock: %d\n", ptr -> stock);
386     }
387     ptr = ptr -> next;
388   }
389   if (check_id == 0)
390   {
391     printf ("\n(!) ID not found!\n");
392   }
393 }
394 //  Funcao para ler o catalogo completo dos jogos
395 void Catalog (pointer ptr)
396 {
397   if (ptr)
398   {
399     if (ptr -> price != 0)
400     {
401       printf ("* ID: %d | Name: %s | Price: $%.2f | Stock: %d\n", ptr -> id, ptr ->
              name, ptr -> price, ptr -> stock);
402     }
403     else
404     {
405       printf ("* ID: %d | Name: %s | Free to Play | Stock: %d\n", ptr -> id, ptr ->
              name, ptr -> stock);
406     }
407     Catalog (ptr -> next);  //  Chamada recursiva para ler o jogo seguinte
408   }
409 }
410 //  Funcao para comprar ou vender um jogo
411 void Buy (pointer ptr)
412 {
413   char temp [MAX];
414   int temp_id;
415   float temp_price;
416   int copies;
417   int temp_stock;
418   int check_id = 0;
419
420   printf ("> ID: ");
421   fgets (temp, sizeof (temp), stdin);
422   temp_id = ScanfSub (temp, temp_id);
423
424   while (ptr)
425   {
426     if (temp_id == ptr -> id)
427     {
```

```
428         check_id = 1;
429
430         printf ("> Name: %s\n", ptr -> name);
431
432         if (ptr -> price != 0)
433         {
434           printf ("> Price: $%.2f\n", ptr -> price);
435         }
436         else
437         {
438           printf ("> Price: Free to Play\n");
439         }
440         if (ptr -> stock > 0)
441         {
442           printf ("> Copies (%d in stock) (> 0 to buy, < 0 to sell, = 0 to cancel): ",
                  ptr -> stock);
443         }
444         else
445         {
446           printf ("> Copies (empty stock) (< 0 to sell, = 0 to cancel): ");
447         }
448         fgets (temp, sizeof (temp), stdin);
449         copies = ScanfSub (temp, copies);
450         system ("clear");
451
452         if (copies <= ptr -> stock && copies != 0)  // Se as copias pedidas forem
              menores do que o stock e forem diferentes de 0
453         {
454           printf ("\n[Checkout]\n\n");
455
456           printf ("> ID: %d\n", ptr -> id);
457           printf ("> Name: %s\n", ptr -> name);
458
459           if (copies > 0)
460           {
461             printf ("> Copies to buy: %d\n", copies);
462             printf ("> Total amount to pay: $%.2f\n", ptr -> price * copies);
463           }
464           else
465           {
466             printf ("> Copies to sell: %d\n", -copies);
467             printf ("> Total amount to receive: $%.2f\n", ptr -> price * (-copies));
468           }
469           printf ("\n(!) Press 'Enter' to confirm.");
470           getchar ();
471           system ("clear");
472           printf ("\n[Checkout]\n\n");
473
474           if (copies == 1)
475           {
476             printf ("(!) 1 copy of %s was added to your inventory, enjoy it and have
                    fun!\n", ptr -> name);
477           }
478           else if (copies == -1)
479           {
480             printf ("(!) 1 copy of %s was sold and taken out from your inventory!\n",
                    ptr -> name);
481           }
```

```
482         else if (copies > 1)
483         {
484           printf ("(!) %d copies of %s were added to your inventory, enjoy them and
                have fun!\n", copies, ptr -> name);
485         }
486         else
487         {
488           printf ("(!) %d copies of %s were sold and taken out from your inventory!\
                n", -copies, ptr -> name);
489         }
490         temp_stock = ptr -> stock;
491         temp_stock = temp_stock - copies;
492         ptr -> stock = temp_stock;
493       }
494       else if (copies != 0)
495       {
496         printf ("\n[Buying / selling a game]\n\n");
497         printf ("> Copies: %d\n\n", copies);
498         if (ptr -> stock <= 0)
499         {
500           printf ("(!) The stock for %s is empty, likewise there are no copies to be
                sold!\n", ptr -> name);
501         }
502         else if (ptr -> stock == 1)
503         {
504           printf ("(!) There is only 1 copy of %s in stock!\n", ptr -> name);
505         }
506         else
507         {
508           printf ("(!) There are only %d copies of %s in stock!\n", ptr -> stock,
                ptr -> name);
509         }
510         printf ("(!) If you need more copies, please update the game's data and
              refill the stock. ");
511         printf ("Otherwise, please choose a lower number of copies.\n");
512       }
513       else
514       {
515         printf ("\n[Buying / selling a game]\n\n");
516         printf ("(!) Purchase cancelled!\n");
517       }
518     }
519     ptr = ptr -> next;
520   }
521   if (check_id == 0)
522   {
523     printf ("\n(!) ID not found!\n");
524   }
525 }
```

Exemplo de *stock*

Copie e cole o seguinte texto para um ficheiro *steam(underscore)storage.txt* na diretoria do ficheiro em *C*.

```
10
Counter-Strike
9.990000
9999
20
Team Fortress Classic
4.990000
9999
30
Day of Defeat
4.990000
9999
40
Deathmatch Classic
4.990000
9999
50
Half-Life: Opposing Force
4.990000
9999
60
Ricochet
4.990000
9999
70
Half-Life
9.990000
9999
80
Counter-Strike: Condition Zero
9.990000
9999
92
Codename Gordon
0.000000
0
100
Counter-Strike: Condition Zero Deleted Scenes
0.000000
9999
130
Half-Life: Blue Shift
4.990000
9999
220
Half-Life 2
9.990000
9999
240
Counter-Strike: Source
19.990000
9999
280
Half-Life: Source
9.990000
9999
300
Day of Defeat: Source
```

```
9.990000
9999
320
Half-Life 2: Deathmatch
4.990000
9999
340
Half-Life 2: Lost Coast
0.000000
9999
360
Half-Life Deathmatch: Source
9.990000
9999
380
Half-Life 2: Episode One
7.990000
9999
400
Portal
9.990000
9999
420
Half-Life 2: Episode Two
7.990000
9999
440
Team Fortress 2
0.000000
9999
500
Left 4 Dead
19.990000
9999
550
Left 4 Dead 2
19.990000
9999
570
Dota 2
0.000000
9999
620
Portal 2
19.990000
9999
630
Alien Swarm
0.000000
9999
730
Counter-Strike: Global Offensive
14.990000
9999
4000
Garry's Mod
9.990000
9999
```

```
30000
Half-Life 3
99.989998
0
245550
Free to Play
0.000000
9999
450390
The Lab
0.000000
9999
453170
Destions
0.000000
9999
```

Conclusão

Em suma, as tarefas foram cumpridas e conjugadas com êxito, sendo que o resultado final se revela um quanto fruitivo.

Foi, no geral, um trabalho extremamente enriquecedor e altamente pedagógico uma vez que conjugou o aprefeiçoamento em *C* (principalmente no manuseamento em ficheiros, apontadores e listas ligadas), em *L*inux e em LaTeX.