IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

ATAD 2019/20

Algoritmos e Tipos Abstratos de Dados

Enunciado de Laboratório

AVALIADO

7

Leitura / Importação de Dados de Ficheiros
Utilização e manipulação do ADT List

Objetivos:

- Implementar código para importar dados de ficheiro CSV para coleção ADT List.
- Saber utilizar o ADT List num programa envolvendo ordenação e pesquisa em listas. Saber calcular uma estatística.
- Identificar o tempo de execução do código tendo em consideração a Estrutura de Dados usada na implementação do ADT List (ArrayList vs. LinkedList).

Pré-requisitos:

- Ficheiro de dados disponibilizado no Moodle – ***kahootReports.csv***.
- Leitura do capítulo correspondente do livro "Tipos Abstratos de Dados – Linguagem C. Bruno Silva".

Notas importantes:

- Deverá garantir que possui o **valgrind** instalado e incluir o *bash script* na pasta do projeto com o nome `mem_check.sh` (ver laboratórios anteriores).
- Deverá manter a possibilidade de compilar os diferentes níveis no ficheiro *Makefile*.
- No template de submissão do relatório deverá também colocar o **output da execução do *valgrind*** em cada nível.
- Deverá gerar a documentação *doxygen* para todos os níveis.
- **No final deste enunciado encontra os resultados esperados para cada um dos níveis.**

Nível 1

1. Copie o ficheiro "*kahootReports.csv*" para a sua pasta de trabalho.
2. Visualize o seu conteúdo, notando que cada linha representa a informação relativa à prestação de cada participante nos questionários kahoot ao longo de várias semanas.

```
<week>;<rank>;<nickname>;<total_score>;<correct_answers>;<incorrect_answers>
```

Onde:

- *week*, representa a semana em que foi efetuado o questionário;
- *rank*, o lugar na tabela classificativa onde o participante ficou;
- *nickname*, a "alcunha" do participante;
- *total_score*, a pontuação total do participante nessa semana;
- *correct_answers*, o número de respostas corretas;
- *incorrect_answers*, o número de respostas incorretas.

3. Crie o módulo **kahoot**, onde define o tipo concreto KahootReport que permite guardar a informação relativa a uma participação num questionário kahoot. Adicione ao módulo as funções kahootReportCreate e kahootReportPrint. Implemente-as e teste-as no ficheiro main1.c.

Nível 2

4. Descarregue a pasta **ADTList** do GitHub, e copie os ficheiros *list.h, listArrayList.c listLinkedList.c e listElem.h e listElem.c* para a sua pasta de trabalho.
5. Altere o tipo ListElem de forma a podermos manipular elementos do tipo KahootReport.
6. Crie um módulo **utils,** adicione e implemente a função split (figura 1 em anexo), que recebe um array de carateres e devolve um array de arrays de carateres.
   Necessita incluir a biblioteca <string.h>.
7. Defina e implemente a função da Figura 2 (em anexo), que recebe o nome de um ficheiro, no módulo **utils**. Esta função abre o ficheiro em modo de leitura, adiciona as linhas do ficheiro a uma lista e liberta o acesso ao ficheiro; para cada linha invoca a função *split* para obter todos diferentes campos da resposta de um aluno.
8. Crie um programa (no ficheiro main2.c) que importe os dados do ficheiro "*kahootReports.csv*" para ADTList e depois os imprima.
9. Compile e execute o programa, testando o resultado esperado. Nota: Tenha atenção que o *Makefile* terá de contemplar todo o código fonte necessário para uma compilação com sucesso.
10. Teste o programa com o *valgrind* e garanta que não existem *memory leaks*.

Nível 3

11. Implemente uma função, no módulo **utils,** que ordena a lista de participações no kahoot por ordem alfabética do nickname. Utilize um dos algoritmos de ordenação dados nas aulas TP, adaptando o algoritmo para usar as operações do ADT List (e.g., listGet, listGet, etc.).
12. Implemente agora uma função que ordene a lista de participações por ordem crescente de semana. Utilize o algoritmo de ordenação que não usou na alínea anterior. Isto é se na alínea anterior usou o selection sort agora terá que usar o bubblesort e vice-versa.
13. Teste a função anterior no ficheiro main3.c e imprima o conteúdo da lista ordenada.
14. Teste o programa com o *valgrind*.

Nível 4

14. Implemente uma função que recebe a lista de participações no kahoot e uma dada semana e que calcula o número de questões que foram efetuadas nessa semana.
15. Implemente uma função que recebe a lista de participações no kahoot e que imprime numa matriz a pontuação total média dos participantes por semana. Se achar necessário pode usar uma das funções implementadas anteriormente. Exemplo:

Week	Total_Score Average
1	2457.78
2	1247.34
…	…

16. Compile e teste as funções no ficheiro main4.c. Teste o programa com o *valgrind*.

Nível 5

17. De forma a ter noção dos tempos de execução do nosso código usando a implementação do ADT List com ArrayList ou com LinkedList, considere o seguinte esqueleto de código, que calcula o tempo de execução de um segmento de código em microssegundos (Nota: Dado o volume de dados em questão ser reduzido e de forma a conseguir visualizar um tempo, a escala tem de ser da ordem dos micro segundos. Em situações em que o volume de dados é significativo usar-se-ia a escala dos segundos):

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
//...

typedef unsigned long long timestamp_t;

static timestamp_t get_timestamp();

int main(){
    // Calculate time taken by a code segment
    printf("Timer starts \n");

    timestamp_t t0 = get_timestamp();


    // CODE TO TEST <--- HERE --->


    printf("Timer ends \n");

    // calculate the elapsed time
    timestamp_t t1 = get_timestamp();

    double secs = (t1 - t0) / 1000000.0L;

    printf("The program took %f microseconds to execute\n", secs);

    return EXIT_SUCCESS;
}

static timestamp_t get_timestamp (){
    struct timeval now;
    gettimeofday (&now, NULL);
    return  now.tv_usec + (timestamp_t)now.tv_sec * 1000000;
}
```

18. Copie o código para o ficheiro main5.c, e acrescente uma funcionalidade desenvolvida anteriormente à sua escolha (com excepção do nível 1).
19. Compile com a implementação listArrayList.c e teste. Tire um screenshot do resultado.
20. Compile agora com a implementação listLinkedList.c e teste. Tire um screenshot do resultado.
21. Registe o que observa. Qual das duas implementações é mais rápida?

Anexos

Funções

```c
#include <stdlib.h>
#include <string.h>


char** split(char *string, int nFields, const char *delim) {

    char **tokens = (char**) malloc(sizeof(char*) * nFields);

    int index = 0;

    char *token = (char*)strtok(string, delim);

    while (token) {
        tokens[index++] = token;
        token = strtok(NULL, delim);
    }

    return tokens;
}
```

Figura 1 – Função split.

```c
void importKahootReportsFromFile(char *filename, PtList *listKR) {
    FILE *f = NULL;

    f = fopen(filename, "r");

    if (f == NULL) {
        printf("An error ocurred... It was not possible to open the file %s ...\n", filename);
        return;
    }

    char nextline[1024];

    int countKR = 0; //kahoot report count
    bool firstLine = true;

    *listKR = listCreate(10);

    while (fgets(nextline, sizeof(nextline), f)) {
        if (strlen(nextline) < 1)
            continue;

        /*As the first line of the file contains the names of the fields it should be ignored*/
        if (firstLine){
            firstLine = false;
            continue;
        }

        char **tokens = split(nextline, 6, ";");

        //At this moment the tokens array is composed with the following "strings"
        //tokens[0] - week
        //tokens[1] - rank
        //tokens[2] - nickname
        //tokens[3] - total_score
        //tokens[4] - correct_answers
        //tokens[5] - incorrect_answers

        KahootReport kr = kahootReportCreate(atoi(tokens[0]), atoi(tokens[1]),
                                             tokens[2], atoi(tokens[3]),
                                        atoi(tokens[4]), atoi(tokens[5]));

        free(tokens); // release of the memory alocated in function split

        int error_code = listAdd(*listKR, countKR, kr);

        if (error_code == LIST_FULL ||error_code == LIST_INVALID_RANK ||
            error_code == LIST_NO_MEMORY || error_code == LIST_NULL){
            printf("An error ocurred... Please try again... \n");
            return;
        }
        countKR++;
    }

    printf("\n\n%d kahoot reports were read!\n\n", countKR);
    fclose(f);
}
```

Figura 2 – importFile Function

Resultados Esperados

Nível 1:

(exemplo aleatório)

```
1   | 3   | Tony       | 3956  | 5   | 1
```

Nível 2:

(Trecho com os primeiros 22 elementos)

```
95 kahoot reports were read!

List contents (by rank):
Rank  0:  1   | 1   | Frank      | 4259  | 5   | 1
Rank  1:  1   | 2   | Mick       | 4211  | 5   | 1
Rank  2:  1   | 3   | Tony       | 3956  | 5   | 1
Rank  3:  1   | 4   | Peppe      | 3740  | 4   | 2
Rank  4:  1   | 5   | Gabriel    | 3722  | 5   | 1
Rank  5:  1   | 6   | Rick       | 3586  | 4   | 2
Rank  6:  1   | 7   | Cory       | 3571  | 4   | 2
Rank  7:  1   | 8   | Sofi       | 3509  | 4   | 2
Rank  8:  1   | 9   | Theo       | 3367  | 4   | 2
Rank  9:  1   | 10  | Vi         | 3200  | 4   | 2
Rank 10:  1   | 11  | Alf        | 3105  | 4   | 2
Rank 11:  1   | 12  | Russ       | 3068  | 4   | 2
Rank 12:  1   | 13  | Johnny     | 2951  | 4   | 2
Rank 13:  1   | 14  | Dick       | 2721  | 3   | 3
Rank 14:  1   | 15  | Mike       | 2518  | 3   | 3
Rank 15:  1   | 16  | Michl      | 2414  | 3   | 3
Rank 16:  1   | 17  | Bernie     | 1635  | 2   | 4
Rank 17:  1   | 18  | Kahoot     | 1576  | 2   | 4
Rank 18:  1   | 19  | Andy       | 1283  | 2   | 4
Rank 19:  1   | 20  | Fred       | 1196  | 2   | 4
Rank 20:  1   | 21  | Peter      | 743   | 1   | 5
Rank 21:  1   | 22  | Someone    | 0     | 0   | 6
Rank 22:  1   | 23  | Walt       | 0     | 0   | 6
...
```

Nível 3:

(Trecho com os primeiros 22 elementos)

```
List contents (by rank):
Rank  0:  3   | 7   | ????       | 5725  | 7   | 4
Rank  1:  1   | 11  | Alf        | 3105  | 4   | 2
Rank  2:  2   | 13  | Alf        | 5735  | 6   | 4
Rank  3:  4   | 2   | Alf        | 4798  | 5   | 1
Rank  4:  5   | 4   | Alf        | 4684  | 5   | 2
Rank  5:  1   | 19  | Andy       | 1283  | 2   | 4
Rank  6:  2   | 19  | Andy       | 3855  | 4   | 6
Rank  7:  3   | 18  | Andy       | 2173  | 3   | 8
Rank  8:  4   | 17  | Andy       | 2635  | 3   | 3
Rank  9:  5   | 6   | Andy       | 4102  | 4   | 3
Rank 10:  2   | 14  | Appy       | 5618  | 6   | 4
Rank 11:  3   | 19  | Appy       | 1977  | 3   | 8
Rank 12:  1   | 17  | Bernie     | 1635  | 2   | 4
Rank 13:  2   | 10  | Bernie     | 6141  | 6   | 4
Rank 14:  3   | 8   | Bernie     | 5562  | 7   | 4
Rank 15:  4   | 19  | Brunch     | 0     | 0   | 6
Rank 16:  4   | 10  | Cathy      | 4003  | 4   | 2
Rank 17:  1   | 7   | Cory       | 3571  | 4   | 2
Rank 18:  2   | 18  | Cory       | 4169  | 5   | 5
Rank 19:  2   | 16  | Daniel     | 4477  | 6   | 4
Rank 20:  4   | 9   | Davy       | 4348  | 5   | 1
Rank 21:  5   | 9   | Davy       | 3671  | 4   | 3
Rank 22:  1   | 14  | Dick       | 2721  | 3   | 3
```

Sort by nickname

```
List contents (by rank):
Rank  0:  1   | 11  | Alf        | 3105  | 4   | 2
Rank  1:  1   | 19  | Andy       | 1283  | 2   | 4
Rank  2:  1   | 17  | Bernie     | 1635  | 2   | 4
Rank  3:  1   | 7   | Cory       | 3571  | 4   | 2
Rank  4:  1   | 14  | Dick       | 2721  | 3   | 3
Rank  5:  1   | 1   | Frank      | 4259  | 5   | 1
Rank  6:  1   | 20  | Fred       | 1196  | 2   | 4
Rank  7:  1   | 5   | Gabriel    | 3722  | 5   | 1
Rank  8:  1   | 13  | Johnny     | 2951  | 4   | 2
Rank  9:  1   | 18  | Kahoot     | 1576  | 2   | 4
Rank 10:  1   | 16  | Michl      | 2414  | 3   | 3
Rank 11:  1   | 2   | Mick       | 4211  | 5   | 1
Rank 12:  1   | 15  | Mike       | 2518  | 3   | 3
Rank 13:  1   | 4   | Peppe      | 3740  | 4   | 2
Rank 14:  1   | 21  | Peter      | 743   | 1   | 5
Rank 15:  1   | 6   | Rick       | 3586  | 4   | 2
Rank 16:  1   | 12  | Russ       | 3068  | 4   | 2
Rank 17:  1   | 8   | Sofi       | 3509  | 4   | 2
Rank 18:  1   | 22  | Someone    | 0     | 0   | 6
Rank 19:  1   | 9   | Theo       | 3367  | 4   | 2
Rank 20:  1   | 3   | Tony       | 3956  | 5   | 1
Rank 21:  1   | 10  | Vi         | 3200  | 4   | 2
Rank 22:  1   | 23  | Walt       | 0     | 0   | 6
```

Sort by week

Nível 4:

```
Insert week number: 2
The week 2 had 10 questions


| Week   | Total_Score Average    |
| 1      | 2623.09                |
| 2      | 6047.76                |
| 3      | 4897.30                |
| 4      | 3787.26                |
| 5      | 3898.58                |
```

Nível 5:

(*A cargo do aluno*).

(fim de enunciado)