



École Centrale de Lille

AAP - Algorithmique avancée et programmation S5

BARBOZA DE DEUS FONSECA Fernando

Compte Rendu - TEA Séance 2

Responsable :

MONSIEUR THOMAS BOURDEAUD HUY

Villeneuve-d'Ascq

Novembre 2021

Table des matières

1	Introduction	3
2	Partie 1	3
2.1	Développement	3
2.1.1	Code <i>base_change.c</i>	3
2.1.2	Code <i>main.c</i>	4
2.2	Résultat	5
3	Partie 2	5
3.1	Développement	5
3.1.1	Code <i>list_v2.h</i>	5
3.1.2	Code <i>test.c</i>	6
3.1.3	Code <i>main.c</i>	7
3.2	Résultat	8
4	Partie 3	8
4.1	Développement	8
4.1.1	Code <i>genPNG.c</i>	8
4.1.2	Code <i>main.c</i>	10
4.2	Résultat	11
5	Conclusion	12
6	Références	12

Table des figures

1	Représentation graphique de la conversion de base	3
2	Résultat Partie 1	5
3	Résultat Partie 2	8
4	Résultat Partie 3 - Exécution	11
5	Résultat Partie 3 - PNG	12

1 Introduction

L'objectif de ce TEA est de travailler avec des piles et des listes chaînées afin de s'entraîner avec des pointeurs et structures en langage C. Nous avons les fichiers *elt.h*, *elt.c*, *list.h*, *list.c*, *stack_ckd.h*, *stack_cld.c*, *main.c*, *makefile* et *makefile_sources* utilisés dans la Séance 2.

La activité est divisé en 3 parties :

- Partie 1 : Faire une fonction de conversion de base numérique avec le stockage en pile ;
- Partie 2 : Faire 5 nouvelles méthodes pour la bibliothèque list.h (Itérative ET Récursive) ;
- Partie 3 : Faire une fonction pour générer une représentation graphique d'une liste chaînée à l'aide du logiciel *graphviz* ;

Le dossier S2_TEA est organisé comme suit :

- Fichier **README** avec des informations sur makefile e la structure ;
- Fichier *makefile* pour toutes les parties ;
- Dossier *include* avec les fichiers utilisés dans tous les exercices ;
- Dossiers *partie** avec la *main.c* et les fichiers utilisés dans chaque exercice

2 Partie 1

2.1 Développement

Pour convertir la valeur V en décimal en base B. L'algorithme effectue des divisions successives de V par B et stocke le reste de chaque division dans une pile. Montrant la pile, on a V dans la nouvelle base B.

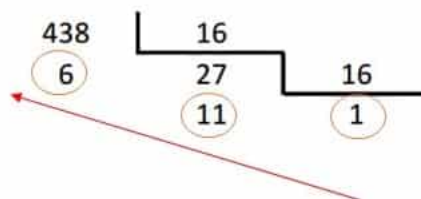


FIGURE 1 – Représentation graphique de la conversion de base

2.1.1 Code *base_change.c*

```
#include "base_change.h"

void base_change_basic(int value, int new_base){
    int Q = value;
```

```

        while(Q!=0){
            printf("%d ", Q % new_base);
            Q /= new_base;
        }

        return;
    }

void base_change_stack(int value, int new_base){
    T_stack s = NULL;
    int Q = value;
    while(Q!=0){
        push(Q % new_base, &s);
        Q /= new_base;
    }

    showStack(&s);

    freeStack(&s);

    return;
}

```

2.1.2 Code *main.c*

```

#include <stdio.h>

#include "../include/traces.h"
#include "../include/check.h"

#include "base_change.h"

int main(int argc, char ** argv) {

    CHECK_IF(argc, 1, "ERROR: Wrong Usage!!\nPlease insert a correct input:\n ./*.exe

    int decimal = atoi(argv[1]);
    int new_base;
    if(argv[2] == NULL) new_base = 2;
    else new_base = atoi(argv[2]);

    printf("Changing %d in base 10 to base %d:\n\n", decimal, new_base);
    printf("Not using stacks (Inverted Order!): ");
    base_change_basic(decimal, new_base);
    NL();
    printf("Using stacks (Correct Order!):          ");
    base_change_stack(decimal, new_base);
    NL();

    return 0;
}

```

2.2 Résultat

```
fernandobdf23@fernandofonseca-VM:~/AAP/S2_TEA$ make partie1
Execution de make partie1 :
make[1]: Entrando no diretório '/home/fernandobdf23/AAP/S2_TEA/partie1'
Le programme partie1.exe a été produit dans le répertoire partie1
make[1]: Saindo do diretório '/home/fernandobdf23/AAP/S2_TEA/partie1'
fernandobdf23@fernandofonseca-VM:~/AAP/S2_TEA$ ./partie1/partie1.exe 438 16
Changing 438 in base 10 to base 16:

Not using stacks (Inverted Order!): 6 11 1
Using stacks (Correct Order!):      1 11 6
fernandobdf23@fernandofonseca-VM:~/AAP/S2_TEA$
```

FIGURE 2 – Résultat Partie 1

Nous avons testé le fonction sans et avec piles. A partir des résultats, l'algorithme sans pile présente le résultat inversé, tandis qu'avec une pile, il présente le résultat correct.

3 Partie 2

3.1 Développement

Pour cette partie, nous avons créé une nouvelle version du fichier *list.c* avec les nouvelles méthodes implémentées qui appelées *list_v2.c*. De plus, une fonction de test appelée *test.c* a été créée, dans laquelle une liste d'exemples est faite et les méthodes itératives et récursives sont testées. Enfin, la liste chaînée est libérée de la mémoire.

3.1.1 Code *list_v2.h*

```
#pragma once
#include "elt.h" // T_elt
// #define CLEAR2CONTINUE
// #define DEBUG
#ifndef _LIST_H_
#define _LIST_H_

typedef struct node {
    T_elt data;
    struct node *pNext;
} T_node, * T_list;

T_node * addNode (T_elt e, T_node * n);
void showList(T_list l);
void freeList(T_list l);
T_elt getFirstElt(T_list l);
T_list removeFirstNode(T_list l);
void showList_rec(T_list l);
void showList_inv_rec(T_list l);
```

```

void freeList_rec(T_list l);

//New Methods
unsigned int getSizeIte(const T_list l);
unsigned int getSizeRec(const T_list l);
T_list tailAddNodeIte(T_elt e, T_list l);
T_list tailAddNodeRec(T_elt e, T_list l);
T_list sortAddNodeIte(T_elt e, T_list l);
T_list sortAddNodeRec(T_elt e, T_list l);
int inListIte(T_elt e, const T_list l);
int inListRec(T_elt e, const T_list l);
T_list removeDuplicatesIte(T_list l);
T_list removeDuplicatesRec(T_list l);

#endif

```

3.1.2 Code *test.c*

```

#include <assert.h> // assert

#include "../include/check.h"

#include "../include/traces.h"

#include "list_v2.h"
#include "test.h"

T_list genTestList(){

    T_list l = NULL;

    for(int i = 5; i > 0; i--){
        l = addNode(i,l);
    }

    return l;
}

T_list genDuplicatedTestList(){

    T_list l = NULL;
    for(int i = 5; i > 0; i--){
        l = addNode(i,l);
        if(i%2) l = addNode(i,l);
    }

    return l;
}

void test(){

    T_list l = genTestList();

    printf("Testing new functions for lists\n");
    printf("Exemple list: ");
    showList(l); NL();
}

```

```

printf("Size of the list (Iterative): %d\n", getSizeIte(l));
printf("Size of the list (Recursive): %d", getSizeRec(l));
NL();

printf("Adding %d to the tail (Iterative): ", 7);
l = tailAddNodeIte(7, l); showList(l); NL();
printf("Adding %d to the tail (Recursive): ", 9);
l = tailAddNodeRec(9, l); showList(l); NL();

printf("Sort adding %d to the list (Iterative): ", 6);
l = sortAddNodeIte(6, l); showList(l); NL();
printf("Sort adding %d to the list (Recursive): ", 8);
l = sortAddNodeRec(8, l); showList(l); NL();

printf("Is %d in the list? (Iterative) %d (True)\n", 9, inListIte(9, l));
printf("Is %d in the list? (Recursive) %d (False)\n", 10, inListIte(10, l));
NL();

T_list l_dup1 = genDuplicatedTestList();
printf("New example list with duplicates: ");
showList(l_dup1); NL();

printf("Removing duplicates (Iterative): ");
l_dup1 = removeDuplicatesIte(l_dup1); showList(l_dup1); NL();

T_list l_dup2 = genDuplicatedTestList();
printf("Removing duplicates (Recursive): ");
l_dup2 = removeDuplicatesRec(l_dup2); showList(l_dup2); NL();

freeList(l); freeList(l_dup1); freeList(l_dup2);

return;
}

```

3.1.3 Code *main.c*

```

#include "test.h"

int main(int argc, char ** argv) {

    test();

    return 0;
}

```

3.2 Résultat

```
Fernandobdf23@fernandofonseca-VM:~/AAP/S2_TEA$ make partie2
Execution de make partie2 :
make[1]: Entrando no diretório '/home/fernandobdf23/AAP/S2_TEA/partie2'
Le programme partie2.exe a été produit dans le répertoire partie2
make[1]: Saindo do diretório '/home/fernandobdf23/AAP/S2_TEA/partie2'
Fernandobdf23@fernandofonseca-VM:~/AAP/S2_TEA$ ./partie2/partie2.exe
Testing new functions for lists
Exemple list: 1 2 3 4 5
Size of the list (Iterative): 5
Size of the list (Recursive): 5
Adding 7 to the tail (Iterative): 1 2 3 4 5 7
Adding 9 to the tail (Recursive): 1 2 3 4 5 7 9
Sort adding 6 to the list (Iterative): 1 2 3 4 5 6 7 9
Sort adding 8 to the list (Recursive): 1 2 3 4 5 6 7 8 9
Is 9 in the list? (Iterative) 1 (True)
Is 10 in the list? (Recursive) 0 (False)

New exemple list with duplicates: 1 1 2 3 3 4 5 5
Removing duplicates (Iterative): 1 2 3 4 5
Removing duplicates (Recursive): 1 2 3 4 5
Fernandobdf23@fernandofonseca-VM:~/AAP/S2_TEA$
```

FIGURE 3 – Résultat Partie 2

4 Partie 3

4.1 Développement

Pour faire des PNG des liste chaînée, Nous avons créé :

- Un modèle de base appelé *template.dot* qui contient les informations de formatage du fichier final;
- Une fonction pour créé la liste d'exemple et un autre pour l'utilisateur pour créer la liste;
- Une fonction qui prend le nom du fichier et la liste chaînée, crée le *.dot et exécute la commande pour générer le PNG;

4.1.1 Code *genPNG.c*

```
#include <assert.h> // assert

#include "../include/check.h"

#include "../include/traces.h"

#include "list_v2.h"
#include "genPNG.h"

void generatePNG(const T_list l, const char * filename){
```



```

char fNameOutput[50], cmd[100];

sprintf(fNameOutput, "%s.dot", filename);

FILE *fpTemplate, *fpOutput;

fpTemplate = fopen("template.dot", "r");
if(fpTemplate == NULL){
    fpTemplate = fopen("partie3/template.dot", "r");
    sprintf(fNameOutput, "partie3/%s.dot", filename);
}

fpOutput = fopen(fNameOutput, "w");

char ch = fgetc(fpTemplate);
while (ch != EOF){
    fputc(ch, fpOutput);
    ch = fgetc(fpTemplate);
}

int i = 1;
T_list aux = 1;

while(aux != NULL){
    if(aux->pNext != NULL){
        fprintf(fpOutput, "\\ID_%04i\\" [label = "\\{<elt> %s | <next> }\\"];\\n", i, t
        fprintf(fpOutput, "\\ID_%04i\\" : next -> "\\ID_%04i\\";\\n", i, i+1);
    }
    else{
        fprintf(fpOutput, "\\ID_%04i\\" [label = "\\{<elt> %s | <next> NULL}\\"];\\n");
    }
    i++;
    aux = aux->pNext;
}

fclose(fpTemplate);
fclose(fpOutput);

sprintf(cmd, "dot %s -T png -o %s.png", fNameOutput, filename);

printf("Generating PNG:\\n");
printf(">> %s\\n", cmd);
system(cmd);

return;
}

T_list exempleList(){
    T_list l = NULL;

    l = tailAddNodeIte("AAP", l);
    l = tailAddNodeIte("=", l);
    l = tailAddNodeIte("Algorithmique", l);
    l = tailAddNodeIte("Avanc e", l);

```

```

    l = tailAddNodeIte("et", l);
    l = tailAddNodeIte("Programmation", l);

    return l;
}

T_list genList(){

    T_list l = NULL;
    int i = 0;
    char **inputs;
    inputs = (char**) malloc(50*sizeof(char*));
    char input[50];

    while(1){

        printf("Input the %d element of the list: (write q! to finish the list)", i);
        getchar();
        scanf("%s", input);

        if(strcmp("q!", input) == 0) break;
        else {
            inputs[i] = (char*) malloc(strlen(input)*sizeof(char));
            strcpy(inputs[i], input);

            l = tailAddNodeIte(inputs[i++], l);
        }

    }

    free(inputs);

    return l;
}

```

4.1.2 Code *main.c*

```

#include <stdio.h>
#include <assert.h>

// #define CLEAR2CONTINUE
#include "../include/traces.h"
#include "../include/check.h"
// C'est dans le fichier elt.h qu'on doit choisir l'implémentation des T_elt
#include "elt.h"
#include "list_v2.h"
#include "genPNG.h"

int main(int argc, char ** argv) {

    char filename[50];
    T_list l;

```

```

printf("Input the filename: ");
scanf("%s", filename);
printf("Generating PNG in %s.dot\n", filename);

printf("1-Use Exemple List\n2-Input a List\n0-Exit\n");
int c;
scanf("%d", &c);

switch (c)
{
case 1:
    l = exempleList();
    break;
case 2:
    l = genList();
    break;
}

CHECK_IF(l, NULL, "Wrong Usage: PNG will not be generated!!\n");

printf("Generated list: "); showList(l); NL();
generatePNG(l, filename);

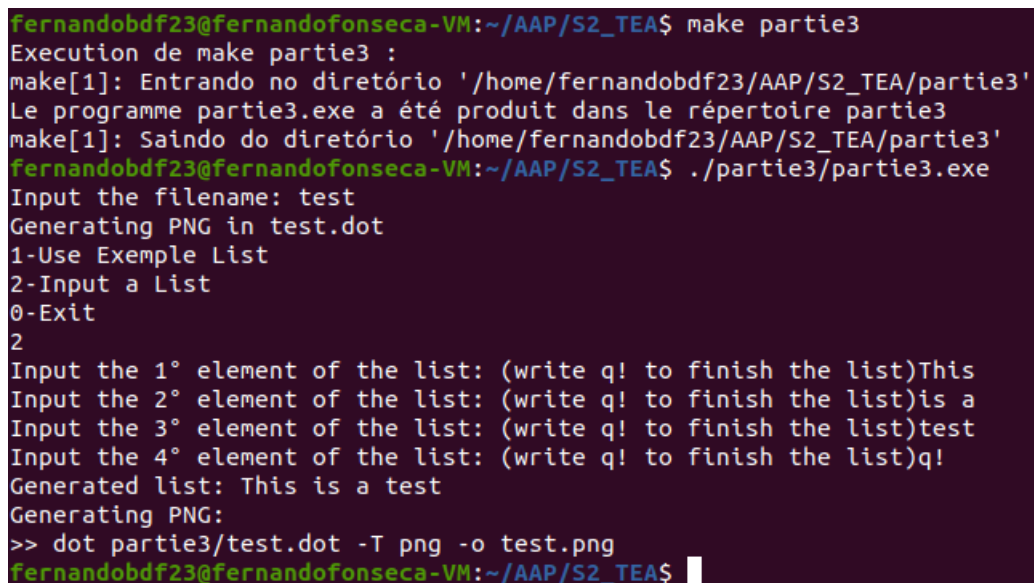
freeList(l);

return 0;
}

```

4.2 Résultat

Au fin, nous avons le PNG comme prévu :



```

fernandobdf23@fernandofonseca-VM:~/AAP/S2_TEA$ make partie3
Execution de make partie3 :
make[1]: Entrando no diretório '/home/fernandobdf23/AAP/S2_TEA/partie3'
Le programme partie3.exe a été produit dans le répertoire partie3
make[1]: Saindo do diretório '/home/fernandobdf23/AAP/S2_TEA/partie3'
fernandobdf23@fernandofonseca-VM:~/AAP/S2_TEA$ ./partie3/partie3.exe
Input the filename: test
Generating PNG in test.dot
1-Use Exemple List
2-Input a List
0-Exit
2
Input the 1° element of the list: (write q! to finish the list)This
Input the 2° element of the list: (write q! to finish the list)is a
Input the 3° element of the list: (write q! to finish the list)test
Input the 4° element of the list: (write q! to finish the list)q!
Generated list: This is a test
Generating PNG:
>> dot partie3/test.dot -T png -o test.png
fernandobdf23@fernandofonseca-VM:~/AAP/S2_TEA$

```

FIGURE 4 – Résultat Partie 3 - Exécution



FIGURE 5 – Résultat Partie 3 - PNG

5 Conclusion

Dans ce TEA, nous avons bien travailler avec des pointeurs et des structure de mémoire comment des piles et des liste chaînées.

6 Références

- AAP - Séance 2
- AAP - TEA S2
- Linked Lists