TP Noté Entraînement

Yohann Trivino

19 avril 2022

1 Tableau Dynamique

Ecrire une fonction *CreerTableauFibonacci* qui prend en entrée *nb* un entier qui décrit la taille voulue du tableau t_1. Puis écrire une fonction *extraitMultiple* qui prend en entrée $t1$, *nb* et un entier *multiple* et qui retournera un tableau contenant les multiples l'entier *multiple*. Evidemment, il faudra aussi l'afficher

```c
int main() {
    int *t1;
    int nb=0;
    int nbItemsLus;

    printf("Combien de termes de la suite de Fibonacci voulez-vous saisir ? "
    );
    nbItemsLus = scanf("%d",&nb);
    if(nbItemsLus != 1) {
        printf("Erreur de saisie\n");
        return 1;
    }
    t1 = CreerTableauFibonacci(nb);
    if(t1==NULL) {
        printf("Impossible de creer le tableau\n");
        return 1;
    }


    int *t2;
    int multiple = scanf("%d",&multiple);
    t2 = extraitMultiple(t1, nb, multiple);

    free(t1);

    //A VOUS DE COMPOSER

    return 0;
}
```

Listing 1 – main de tab.c

cf Correction du TP Noté 1 pour le remplissage des valeurs

```c

int *extraitPairs(int *td, int taille, int multiple) {
    int *tp;
    int nbPairs = 0;
    for (int i=0;i<taille;i++) {
```

```
        if (td[i]%multiple==0) nbPairs = nbPairs + 1 ;
    }
    tp = (int *)malloc((nbPairs) * sizeof(int));
    if(tp==NULL) return NULL;
    else {
        int j = 0;
        for(int i=0; i<taille; i++) {
            if (td[i]%multiple==0) {
                tp[j]=td[i];
                j++;
            }
        }
    }
    return tp;
}
```

Listing 2 – fonction ExtraitMultiple

2 Lecture en Terminal

```

#include <stdio.h>

int main(int argc, char * argv[]) {
    //AJOUTEZ UN AFFICHAGE D'ERREUR SI LE NOMBRE DE PARAMETRES N'EST PAS BON

    int i=0, res=0;
    char found=0;
    while(argv[2][i]!='\0') {
        if(argv[2][i]==argv[1][0]) {
            found = 1;
            break;
        }
        i++;
    }

    //AJOUTEZ UN AFFICHAGE D'ERREUR SI FOUND = 0

    i=0;
    while(argv[2][i]!='\0') {
        if(argv[2][i]==argv[1][0]) res++;
        i++;
    }
    printf("Found %d occurence(s) of %c in %s\n", res, argv[1][0], argv[2]);
    return 0;
}
```

Listing 3 – abc.c

Que fait ce programme ? Complétez ce code en ajoutant les messages d'erreur.

```

#include <stdio.h>

int main(int argc, char * argv[]) {
    if(argc!=3) {
        printf("Wrong usage: expected ./abc char word\n");
        return -1;
    }
```

```
    int i=0, res=0;
    char found=0;
    while(argv[2][i]!='\0') {
        if(argv[2][i]==argv[1][0]) {
            found = 1;
            break;
        }
        i++;
    }
    if(!found) {
        printf("No occurence of char %c in %s\n",argv[1][0],argv[2]);
        return 0;
    }
    i=0;
    while(argv[2][i]!='\0') {
        if(argv[2][i]==argv[1][0]) res++;
        i++;
    }
    printf("Found %d occurence(s) of %c in %s\n", res, argv[1][0], argv[2]);
    return 0;
}
```

Listing 4 – Correction abc.c

3 Liste chaînée

On déclare une structure chaînée de cette forme :

```
typedef string ELEMENT;
typedef struct domino *LISTE;
typedef struct domino{
      ELEMENT val;
      LISTE suiv;
} DOMINO;
```

Listing 5 – Structure

Créer une fonction *cree_domino* qui crée UN SEUL ET UNIQUE domino à la structure chaînée de façon dynamique. Ensuite, vous creerez une fonction *inserer_queue* qui insère un élément en bout de la struture chaînée (étant donné en entrée un ELEMENT)

```
LISTE cree_domino(ELEMENT e){
  LISTE l;
  l=malloc(sizeof(DOMINO));
  l->val=e;
  l->suiv=NULL;
return l; }
```

Listing 6 – fonction cree$_d$*omino*