Structure de données TP5

Table des matières

[Exercice 1 2](#_Toc90222090)

[Code C 2](#_Toc90222091)

[Fichier File.c utilisé pour cet exercice : 4](#_Toc90222092)

[Exercice 2 et 3 réflexions 7](#_Toc90222093)

## Exercice 1

### Code C

#include "file.c"

#include "stdlib.h"

#include "stdio.h"

#include "string.h"

#define NB\_PATIENTS 5

/\* \*\*\*\*\*\*\*\*\*AJOUT IMPLEMENTATIONS FILES \*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\*                                                    \*/

/\*                                                    \*/

/\*                                                    \*/

/\*                                                    \*/

/\*                                                    \*/

/\*--------------------------------------------------- \*/

/\*--------------------------------------------------- \*/

/\* ---------------------------------------------------------- \*/

/\* ---------------------------------------------------------- \*/

void simulation(client tab [], int n) {

//Le tableau est déjà trié par ordre croissant de temps d'arrivée

    file f;

    initialiser (f);

    enfiler (&f, &tab[0]) ;

    enfiler (&f, &tab[1]) ;

    enfiler (&f, &tab[2]) ;

    enfiler (&f, &tab[3]) ;

    enfiler (&f, &tab[4]) ;

    client current ;

    int nb\_elem = 5 ;

    int t\_total = 0, i=0 ;

    for(int i=0 ; i<n ; i++){

        t\_total += tab[i].t\_traitement ;

    }

    printf("temps total = %d\n", t\_total) ;

    i=0 ;

    int temps=0 , examination\_time =0 ;

    while ((f.tete != NULL) && (i < n)){

        printf("il est t=%d, CLient en consultation : n° %d\n", temps, i) ;

        if( (i == 0) && (temps == tab[i].t\_traitement) ){

            printf("Traitement du client numero %d (%s) est terminé \n", i, tab[i].nom) ;

            defiler(&f, &current) ;

            nb\_elem-- ;

            //printf("Client %s parti \n", current.nom) ;

            printf("%s parti, -----PROCHAIN----- %s, %d elements restant  \n",current.nom, f.tete->client.nom, nb\_elem) ;

            examination\_time += tab[i].t\_traitement ;

            printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") ;

            i++ ;

        }

        else if(temps == examination\_time+tab[i].t\_traitement && i!=0){

            printf("Traitement du client numero %d (%s) est terminé \n", i, tab[i].nom) ;

            defiler(&f, &current) ;

            nb\_elem-- ;

            printf("%s parti, %d elements restant  \n",current.nom, nb\_elem) ;

            i++ ;

            examination\_time = temps ;

            printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n") ;

        }

        temps++;

    }

    if(i == n)

        printf("Fin de la journee!\n");

}

/\* ---------------------------------------------------------- \*/

/\* ---------------------------------------------------------- \*/

/\* ---------------------------------------------------------- \*/

int main(int argc, char const \*argv[])

{

    /\*int n = 0 ;

    printf("Combien it a t il de patients?\n");

    scanf("%d", &n);\*/

    client tab [NB\_PATIENTS] ;

    // init Patient

    client p1, p2, p3, p4, p5;

    strcpy(p1.nom, "Edouard");

    p1.t\_arrivee = 3 ;

    p1.t\_traitement = 10 ;

   strcpy(p2.nom, "LePen");

    p2.t\_arrivee = 0 ;

    p2.t\_traitement = 6;

   strcpy(p3.nom, "Zemmour");

    p3.t\_arrivee = 2 ;

    p3.t\_traitement = 8 ;

   strcpy(p4.nom, "Melenchon");

    p4.t\_arrivee = 4 ;

    p4.t\_traitement = 9 ;

   strcpy(p5.nom, "Macron");

    p5.t\_arrivee = 6 ;

    p5.t\_traitement = 4 ;

    // init valeurs temps

//---

    //remplir tableau de patients

    tab[0] = p2 ;

    tab[1] = p3 ;

    tab[2] = p1 ;

    tab[3] = p4 ;

    tab[4] = p5 ;

    //simulation

    simulation(tab, NB\_PATIENTS);

    return 0;

}

## Fichier File.c utilisé pour cet exercice :

#include<stdio.h>

#include<stdlib.h>

typedef struct client{

    char nom [10] ;

    int t\_arrivee ;

    int t\_traitement ;

}client ;

typedef struct maillon

{

    client client;

    struct maillon \* suivant;

}cellule;

typedef struct file{

    cellule \* tete;

    cellule \* queue;

    client top; //tete de file: premier element à supprimer

}file;

void initialiser (file f){

    f.tete = NULL ;

    f.queue = NULL ;

}

int estPleine (file f){

    return 0; //Une file n'est jamais pleine dans cette implémentation

}

int estVide  (file f){

    return (f.tete==NULL) ;

}

int top\_file (file f, client  \* next){

    if (estVide (f) == 1){

        printf("Pile vide \n");

        return 0;

    }

    else{

        \*next = f.tete->client ;

        return 1 ;

    }

}

int enfiler (file \* pfile, client \* new){  //j'enfile à la fin de la liste

    if(estPleine (\*pfile) == 1){

        printf("Pile pleine : Error \n") ;

        return 0 ;

    }

    else if (estVide(\*pfile)){

        //crée une nouveau maillon à relier à liste chaînée

        cellule \*N = malloc(sizeof(cellule)) ;

        //remplit le nouveau maillon

        N->client = \*new ;

        N->suivant = NULL;

        pfile->tete = N ;

        pfile->queue= N ;

    }

    else{

        //crée une nouveau maillon à relier à liste chaînée

        cellule \*N = (cellule \*) malloc(sizeof(cellule)) ;

        //remplit le nouveau maillon

        N->client = \*new ;

        N->suivant = NULL;

        pfile->queue->suivant = N ;

        pfile->queue = N ;

    }

    pfile->top = pfile->tete->client ;

    return 1 ;

}

int defiler (file \* pfile, client \* pClient){  //je défile au début de la liste

    if (estVide(\*pfile)==1){

        printf("Erreur! File vide\n");

        return 0;

    }

    else if(pfile->tete->suivant == NULL){ // Si la file ne contient qu'un seul élément

        \*pClient = pfile->tete->client ;

        pfile->tete = NULL;

        pfile = NULL ;

        return 1 ;

    }

    else {

        //renseigne le client que je veux supprimer (top)

        \*pClient = pfile->top ;

        //création cellule temporaire pour rupture de la chaîne

        cellule \* temp = pfile->tete;

        pfile->tete = temp->suivant;

        //Retrait de L'ancienne tete de file de la chaîne

        temp->suivant = NULL ;

        //Le second maillon devient La tete de file

        pfile->top = pfile->tete->client ;

        free(temp) ;

        return 1;

    }

}

## Exercice 2 et 3 réflexions

2/ Le 2ème exercice ne diffère que très peu du premier, ainsi une solution pourrait être d’ajouter un champ blood\_test\_time (int) dans la structure client.

3/ Une solution serait d’utiliser une pile qui, à chaque nouveau client, se recréerait en se triant par temps d’occultation avec le médecin.