Fiche DS

Char

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

int vowels(char *word);
void arrayOfChar(unsigned int n);
void capitalLetters(char* word);
bool palindrome(char* word);
bool anagram();


int main(int argc, char** argv){
    printf("The number of vowels of %s is: %d \n", argv[1], vowels(argv[1]));
    printf("\n");
    arrayOfChar(atoi(argv[1]));
    printf("\n");
    capitalLetters(argv[1]);
    printf("\n");
    palindrome(argv[1]);
    anagram();
    return 0;
}

int vowels(char* word){
    int i = 0;
    int cpt = 0;
    int len = strlen(word);
    for(i = 0; i < len; i++){
            if(word[i] == 'a' || word[i] == 'e' || word[i] == 'i' || word[i] == 'o' || word[i] == 'u' || word[i] == 'y'){
                    cpt ++;
            }
        }
        return cpt;
}



void arrayOfChar(unsigned int n){
    char str[10][32];
        for(int i=0; i<n; i++){
            printf("%d Enter a string : ", i+1);
            scanf("%s", str[i]);
        }
        printf("The sentence is : ");
        for(int j=0; j<n; j++){
            printf("%s ", str[n-j-1]);
        }
        printf("\n");
}



void capitalLetters(char* word){
    int n = strlen(word);
    for(int i = 0; i < n; i++){
        if(word[i] >= 'a' && word[i] <= 'z'){
            word[i] = word[i] - 32;
        }
    }
    printf("%s\n", word);
}

bool palindrome(char* word){
    int len = strlen(word);
    for(int i = 0; i < len; i++){
        if(word[i] != word[len - i - 1]){
            printf("%s is not a palindrome\n", word);
            return false;
        }
    }
    printf("%s is a palindrome\n", word);
    return true;
}

bool anagram(){
    char str1[32];
    char str2[32];
    printf("Enter first string : ");
    scanf("%s", str1);
    int len1 = strlen(str1);
    printf("Enter second string : ");
    scanf("%s", str2);
    int len2 = strlen(str2);
    if(len1 != len2){
        printf("The strings are not anagrams\n");
        return false;
    }
    for(int i = 0; i < len1; i++){
        for(int j = 0; j < len2; j++){
            if(str1[i] == str2[j]){
                str2[j] = ' ';
                break;
            }
```

```
        }
    }
    for(int i = 0; i < len2; i++){
        if(str2[i] != ' '){
            printf("The strings are not anagrams\n");
            return false;
        }
    }
    printf("The strings are anagrams\n");
    return true;
}

unsigned int letterSum(char** words, unsigned int n){
    unsigned int sum = 0;
    for (int i = 0; i < n; i++){
        sum += strlen(words[i]);
    }
    return sum;
}



unsigned int wordCount(char** words, unsigned int n, char* word){
    unsigned int count = 0;
    for (int i = 0; i < n; i++){
        if (strcmp(words[i], word) == 0){
            count++;
        }
    }
    return count;
}
```

Structure

Person.h

```
#ifndef _PERSON_H
#define _PERSON_H

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

// Structure à utiliser
struct Person{
    char* name;
    unsigned int age;
    float height;
};

// Opérations de création
struct Person* createPerson(char name[32],unsigned int age, float height);
void editName(struct Person* p, char newName[32]);
void editAge(struct Person* p, unsigned int newAge);
void editHeight(struct Person* p, float newHeight);
// Opérations d'accès
char* getName(struct Person* p);
unsigned int getAge(struct Person* p);
float getHeight(struct Person* p);
// Suppression
void deletePerson(struct Person* p);
// Autres opérations
void printPerson(struct Person* p);
struct Person* oldest(struct Person* ps[],unsigned int n);
struct Person* tallest(struct Person* ps[],unsigned int n);
bool sameName(struct Person* ps[],unsigned int n);
bool allAdults(struct Person* ps[],unsigned int n);
unsigned int numberUnderAge(struct Person* ps[],unsigned int n);
void birthdayForAll(struct Person* ps[],unsigned int n);
#endif // _PERSON_H
```

Person.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include "person.h"



// Opérations de création
struct Person* createPerson(char newName[32],unsigned int newAge, float newHeight){
    struct Person* p = malloc(sizeof(struct Person));
    p -> name = malloc(sizeof(char)*32);
    strcpy(p->name, newName);
    p -> age = newAge;
    p -> height = newHeight;
    return p;
}
void editName(struct Person* p, char newName[32]){
    strcpy(p -> name, newName);
}
void editAge(struct Person* p, unsigned int newAge){
    p -> age = newAge;
}
void editHeight(struct Person* p, float newHeight){
    p -> height = newHeight;
```

```c
}
// Opérations d'accès
char* getName(struct Person* p){
    return p -> name;
}
unsigned int getAge(struct Person* p){
    return p -> age;
}
float getHeight(struct Person* p){
    return p -> height;
}
// Suppression
void deletePerson(struct Person* p){
    free(p -> name);
    free(p);
}

// Autres opérations
void printPerson(struct Person* p){
    printf("Name: %s\n", getName(p));
    printf("Age: %d\n", getAge(p));
    printf("Height: %.2f\n", getHeight(p));
}
struct Person* oldest(struct Person* ps[],unsigned int n){
    struct Person* oldestp = ps[0];
    for (int i = 1; i < n; i++){
        if (getAge(ps[i]) > getAge(oldestp)){
            oldestp = ps[i];
        }
    }
    return oldestp;
}
struct Person* tallest(struct Person* ps[],unsigned int n){
    struct Person* tallestp = ps[0];
    for(int i = 1; i < n; i++){
        if(getHeight(ps[i]) > getHeight(tallestp)){
            tallestp = ps[i];
        }
    }
    return tallestp;
}
bool sameName(struct Person* ps[],unsigned int n){
    for(int i = 0; i < n; i++){
        for(int j = i+1; j < n; j++){
            if(strcmp(getName(ps[i]), getName(ps[j])) == 0){
                return true;
            }
        }
    }
    return false;
}
bool allAdults(struct Person* ps[],unsigned int n){
    for(int i = 0; i < n; i++){
        if(getAge(ps[i]) < 18){
            return false;
        }
    }
    return true;
}
unsigned int numberUnderAge(struct Person* ps[],unsigned int n){
    int minors = 0;
    for(int i = 0; i < n; i++){
        if(getAge(ps[i]) < 18){
            minors++;
        }
    }
}
void birthdayForAll(struct Person* ps[],unsigned int n){
    for(int i = 0; i < n; i++){
        editAge(ps[i], getAge(ps[i])+1);
    }
}
```

main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include "person.h"

int main(int argc, char** argv){
    // Test person
    struct Person* p = createPerson("John", 25, 1.99);
    struct Person* p2 = createPerson("Nirina", 18, 1.10);
    struct Person* p3 = createPerson("Yanis", 18, 1.93);
    printPerson(p);
    printf("\n");
    struct Person* ps[3] = {p, p2, p3};
    struct Person* p4 = createPerson("Nirina", 18, 1.80);
    struct Person* ps2[2] = {p2, p4};
    printf("Tallest: %s\n", getName(tallest(ps, 3)));
    printf("\n");
    printf("Oldest: %s\n", getName(oldest(ps, 3)));
    printf("\n");
    printf("Same name: %d\n", sameName(ps, 3));
    printf("\n");
    printf("Same name: %d\n", sameName(ps2, 2));
    printf("\n");
    printf("All adults: %d\n", allAdults(ps, 3));
    printf("\n");
```

```c
    struct Person* p5 = createPerson("Amaury", 17, 1.85);
    struct Person* ps3[4] = {p, p2, p3, p5};
    printf("All adults: %d\n", allAdults(ps3, 4));
    printf("\n");
    printf("Number under age: %d\n", numberUnderAge(ps3, 4));
    printf("\n");
    printPerson(p2);
    printf("\n");
    editName(p2, "Félix");
    editHeight(p2, 1.80);
    printPerson(p2);
    deletePerson(p);
    deletePerson(p2);
    deletePerson(p3);
    deletePerson(p4);
    deletePerson(p5);
}
```

Pointer

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Fonction pour démontrer les bases des pointeurs
void basicPointers() {
        int a = 10;
        int *p = &a; // Pointeur qui stocke l'adresse de a
        printf("Valeur de a: %d\n", a);
        printf("Adresse de a: %p\n", &a);
        printf("Adresse stockée dans p: %p\n", p);
        printf("Valeur pointée par p: %d\n", *p);
        *p = 20; // Modification de a via le pointeur
        printf("Nouvelle valeur de a après modification via p: %d\n", a);
}
// Exemple avec tableau dynamique
void dynamicArray() {
        int n;
        printf("Entrez la taille du tableau : ");
        scanf("%d", &n);
        int *arr = malloc(n * sizeof(int)); // Allocation dynamique d'un tableau
        for (int i = 0; i < n; i++) {
                arr[i] = i + 1; // Initialisation
        }
        printf("Contenu du tableau : ");
        for (int i = 0; i < n; i++) {
                printf("%d ", *(arr + i)); // Utilisation des pointeurs
        } printf("\n");
        free(arr); // Libération de la mémoire
}
// Exemple avec pointeur sur une chaîne de caractères
void stringPointer() {
        char str[] = "Bonjour, Monde!";
        char *ptr = str; printf("Chaîne complète : %s\n", ptr);
        printf("Premier caractère : %c\n", *ptr);
        while (*ptr != '\0') {
                printf("%c ", *ptr); ptr++; // Avancement du pointeur
        }
        printf("\n");
}

// Exemple avec une struct
struct Person {
        char name[32];
        int age;
};

void structurePointerExample() {
        struct Person p = {"Alice", 25};
        struct Person *pPtr = &p;
        printf("Nom : %s\n", pPtr->name);
        printf("Âge : %d\n", pPtr->age); // Modification via pointeur
        strcpy(pPtr->name, "Bob");
        pPtr->age = 30;
        printf("Nom modifié : %s\n", pPtr->name);
        printf("Âge modifié : %d\n", pPtr->age);
}
```