Mathieu GADENNE E13A

Rapport de travaux pratiques de Info1

Appréciation :

N.B : Tout les programmes sont à retrouver à l'adresse suivantes :

https://github.com/MatYeiv/INFO-BUT-GEII-S1

Table des matières

2.Types de variables

2.2 Types entiers

		Signed		Unsigned	
Type	Taille (en octets)	Valeurs		Valeurs	
		minimale	maximale	minimale	maximale
char	1	-128	127	0	255
short	2	-32 788	32 767	0	65 535
int	4	$-2^{15}+1$	$2^{15}-1$	0	$2^{16}-1$
long	8	$2^{31}+1$	$2^{31}-1$	0	$2^{32}-1$

2.3 Types réels

Une variable de types float peut accueillir une variable de type int.

char < short < int < long < float < double.

2.4.1 Capacités

Le résultat affiché est 1 312 alors qu'il devrait afficher 525 600, MpA est mal dimensionné, il doit être codé comme un int (ou un long).

2.4.2 Exercice 2 : tirelires.

- Déclarations des constantes entières :
    Nbp_2E ← 2
    Nbp_1E ← 2
    Nbp_50C ← 2
    Nbp_10C ← 2
- Déclarations des constantes réels :
    change_us
    change_uk
- Déclarations des variables réels :
    résultat
    résultat_us
    résultat_uk

DÉBUT
    résultat ← 2*Nbp_2E + 1*Nbp_1E + 0.5*Nbp_50C + 0.10*Nbp_10C
    résultat_us ← résultat*change_us
    résultat_uk ← résultat*change_uk
    Afficher ← résultat
    Afficher ← résultat_us
    Afficher ← résultat_uk
FIN

3. Entrés sorties standard

3.3 Exercice 1 : convertisseur secondes.

Cet exercice est la version amélioré de l'exercice 2.4.3.

- Déclarations des variables entières :
    secondes
    heures
    minutes
    sec

DÉBUT
    Afficher ← « Saisir le nombres de secondes »
    Saisir → ***secondes***
    heures ← secondes / 3600
    minutes ← (secondes % 3600) / 60
    sec ← secondes % 3600
    Afficher ← ***secondes*** « secondes correspondent à »
    Afficher ← ***heures*** « h, » ***minutes*** « min et » ***sec*** « secondes. »
FIN

3.4 Exercice 2 : volume et surface d'une sphère.

- Déclarations des constantes réels :
    pi = 3,141592
- Déclarations des variables réels :
    surface
    volume
- Déclarations des variables entières :
    rayon

DÉBUT
Afficher ← « Saisir r = »
    Saisir → ***rayon***
    surface ← 4*pi**rayon***rayon***
    volume ← (4.0/3)*pi**rayon***rayon***rayon***
    Afficher ← ***surface***
    Afficher ← ***volume***
FIN

4. La bibliothèque graphique Xgeii

J'ai passé les exercices nécessitant la bibliothèques graphiques.

5. Structures de contrôle

5.1 Les structures de contrôle conditionnelles

5.1.1 Équation du second degré

- Déclarations des variables réels :
    a
    b
    c
    X
    X1
    X2
    delta
    racdelta

DÉBUT
    Afficher ← Entrez les coefficients a, b et c du polynôme
    Saisir → ***a***
    Saisir → ***b***
    Saisir → ***c***
    delta ← (***b*******b***) – 4****a*******c***

    SI ***delta*** > 0
        racdelta ← sqrt(***delta***)
        X1 ← (-***b***–***racdelta***) / (2 * ***a***)
        X2 ← (-***b***+***racdelta***) / (2 * ***a***)
        Afficher ← ***X1*** « et » ***X2***

    SINON SI ***delta*** = 0
        X ← (-***b***) / (2 * ***a***)
        Afficher ← ***X***

    SINON SI ***delta*** < 0
        X ← (-***b***) / (2 * ***a***)
        racdelta ← sqrt(- ***delta***)
        X1 ← (***racdelta***) / (2 * ***a***)
        X2 ← (- ***racdelta***) / (2 * ***a***)
        Afficher ← ***X*** « + » ***X1*** « j »
        Afficher ← ***X*** « + » ***X1*** « j »

    SINON
        Afficher ← « Erreur »
FIN

5.2 Les structures de contrôles répétitives

5.2.3 Tables de multiplication partie III

La partie 1 et 2 des tables de multiplication étant contenue dans le programme je ne les ai donc pas transcrit en pseudo-code.

- Déclarations des variables de types entières
nbr
val
uti
cons ← 0
erreur ← 0

DÉBUT
TANT QUE ***cons*** != 1
Afficher ← « Entrer une valeur entre 1
Saisir → ***nbr***
SI 1 <= ***nbr*** ET ***nbr*** <= 10
cons ← 1

SINON SI ***nbr*** < 1
Afficher ← « Le nombre est trop petit »

SINON SI 10 < ***nbr***

Afficher ← « Le nombre est trop grand »

SINON
Afficher ← « Erreur »

POUR **i** ALLANT DE 1 à 10 PAR PAS DE 1
val ← ***nbr*** * **i**
Afficher ← ***nbr*** « x » **i** « = »
Saisir → ***uti***

TANT QUE val != uti ET i <= 10
Afficher « Erreur »
erreur = ***erreur*** + 1
Afficher ← ***nbr*** « x » **i** « = »
Saisir → ***uti***

Afficher ← ***erreur***
FIN

6. Les fonctions

6.2 Années bissextiles

Programme 1 :

– Fonction vide **afficheSiBissextile**(entier ***année***) :
DÉBUT
    SI (année % 4 = 0 ET année % 100 != 0) OU année % 400 = 0
        Afficher ← « L'année est bissextile »

    SINON
        Afficher ← « L'année n'est pas bissextile »

FIN

– Fonction Principale :
- Déclarations des variables entières
    entreUti ← 0

DEBUT
    Afficher ← « Entrez l'année désirée. »
    Saisir → ***entreUti***
    afficheSiBissextile(***entreUti***)
FIN

Programme 2 :

– Fonction entière **isBissextile**(entier année) :
DEBUT
SI (année % 4 = 0 ET année % 100 != 0) OU année % 400 = 0
Renvoyer ← 2

SINON
Renvoyer ← 1

FIN

– Fonction Principale :
- Déclarations des variables entières
inUti

DEBUT
Afficher ← « Entrez l'année desiré »
Saisir → inUti
SI **isBissextile**(***inUti***) = 2
Afficher ← « L'année est bissextile. »

SINON SI **isBissextile**(***inUti***) = 1
Afficher ← « L'année n'est pas bissextile. »

SINON
Afficher ← « Erreur »

FIN

7. Les tableaux

7.1 La pyramide des âges

– Fonction entière **PluGndNbr**(entier pop[], entier longueur)

- Déclarations des variables entières :
valeurfin ← 0
compteur ← 0
n ← 0

DEBUT
TANT QUE ***n*** <= ***longueur***
SI ***valeurfin*** < ***pop***[n]
valeurfin ← ***pop***[n]
compteur = ***n***

n ← ***n*** + 1
Renvoyer ← ***compteur***
FIN

– Fonction réels **AgeMoyen**(entier pop[], entier ann[], entier longueur)

- Déclarations des variables réels :
moy ← 0
moyenne ← 0
temporaire ←0
massetemp ← 0
massemoy ← 0
- Déclarations des variables entières :
n = 0

DEBUT
TANT QUE ***n*** < ***longueur***
temporaire ← ***pop***[n]
massetemp ← ***massetemp*** + ***temporaire***
moy ← ***temporaire*** * (2021 – ***ann***[n])
massemoy ← ***massemoy*** + ***moy***
n ← ***n*** + 1

moyenne = ***massemoy*** / ***massetemp***
Renvoyer ← ***moyenne***
FIN

– Fonction vide :

tableau(entier pop[], entier longpop, entier ptrtab[], entier longueur)

- Declarations des variables entières :

j = 0

compteur

DEBUT

POUR **i** ALLANT DE 0 à 5****longueur*** PAR PAS DE 5

ptrtab[***j***] ← ***pop***[**i**] + ***pop***[**i**+1] + ***pop***[**i**+2] + ***pop***[**i**+3] + ***pop***[**i**+4]

j ← ***j*** + 1

compteur ← ***longpop*** % 5

CAS EN FONCTION DE ***compteur***

CAS 1 :

ptrtab[l] ← ***pop***[***longpop***-1]

Sortie

CAS 2 :

ptrtab[l] = ***pop***[***longpop***-1] + ***pop***[***longpop***-2]

Sortie

CAS 3 :

ptrtab[l] = ***pop***[***longpop***-1] + ***pop***[***longpop***-2] + pop[***longpop***-3]

Sortie

CAS 4 :

ptrtab[l] = ***pop***[***longpop***-1] + ***pop***[***longpop***-2] + ***pop***[***longpop***-3] + ***pop***[***longpop***-4]

Sortie

CAS DÉFAUT :

Sortie

FIN

– Fonction vide **affiche**(entier tableau[], entier longueur)

DEBUT

Afficher ← « [» tableau[0] « , »

POUR **j** ALLANT DE 1 à (***longueur***-1) PAR PAS DE 1

Afficher ← tableau[j] « , »

Afficher ← tableau[longueur] «] »

FIN

– Fonction Principale

- Déclarations des variables entières :
    annee[] ← {…}
    femme[] ← {…}
    homme[] ← {…}
    longueurf ← sizeof(femme)/sizeof(entier)
    longueurh ← sizeof(homme)/sizeof(entier)
    longueurAn ← sizeof(annee)/sizeof(entier)
    taille ← 0
- Déclarations des pointeurs sur des entiers :
    cinqtab ← NULL

DÉBUT
    SI ***longueurAn*** % 5 != 0
        taille ← (***longueurAn*** / 5) + 1

    SINON
        taille = (***longueurAn***/5)

    cinqtab = (pointeur sur entier)calloc(taille, sizeof(entier))

    SI ***cinqtab*** = NULL
        Renvoyer -1

    Afficher ← « L'age le plus represente chez les femmes est : »
    Afficher ← 2021 - annee[**PluGndNbr**(femme, longueurf)]
    Afficher ← « L'âge moyen des femmes est : »
    Afficher ← **AgeMoyen**(***homme***, ***annee***, ***longueurh***)
    Afficher ← « Le tableau 5 par 5 est : »
    tableau(***homme***, ***longueurh***, ***cinqtab***, ***taille***-1);
    affiche(***cinqtab***, ***taille***-1);
FIN

8. Tableaux, fonctions et pointeurs

8.1 Fonctions et tableaux

8.1.1 Premier programme de test

1)

Lorsque que l'on essaie d'afficher seulement le premier élément du tableau, on voit apparaître 2 fois cet élément.

2)

Modification de la fonction affiche() :

– Fonction vide **affiche**(entier tab[ ], entier size)

DEBUT

SI ***size*** = 1

Afficher ← « [» ***tab***[0] «] »

SINON

Afficher ← « [»

POUR **i** ALLANT DE 1 à (***size***-1) PAR PAS DE 1

Afficher ← « , » ***tab***[i]

Afficher ← « , » ***tab***[***size***-1] «] »

FIN

8.1.2 Second programme de test

La variable « val1 » dans la fonction « exemple » est déclaré localement, elle n'aura donc aucune influence dans la fonction main même s'il lui est affecté une autre valeur. Pour changer la valeur d'une variable dans la fonction main en étant dans une autre fonction, il faut utiliser les « pointeurs ».

8.2 Les pointeurs

Ce code regroupe à la fois le code de la question 8.1.3 ainsi que celui de la question 8.2.1 et 8.2.2.

– Fonction entière **maximum**(entier tab[], entier size) :
- Déclarations des variables entières :
    memory ← 0
    n ← 0

DÉBUT
    TANT QUE ***n*** < ***size***
        SI ***memory*** <= ***tab***[***n***]
            memory ← ***tab***[***n***]
            n ← ***n*** + 1
        SINON
            n ← ***n*** + 1
    Renvoyer ← ***memory***
FIN

– Fonction vide **permute**(pointeur entier A, pointeur entier B)
- Déclarations des variables entières :
    temp

DEBUT
    temp ← ***pointeur A***
    pointeur A ← ***pointeur B***
    pointeur B ← ***temp***
FIN

– Fonction vide **croissant**(entier tab[], entier size) :

DÉBUT
    POUR **i** ALLANT DE 0 à (***size***-1) PAR PAS DE 1
        POUR **j** ALLANT DE **i**+1 à (***size***-1) PAR PAS DE 1
            SI ***tab***[**j**] < ***tab***[**i**]
            permute(adresse ***tab***[**j**], adresse ***tab***[**i**]
    Afficher ← « [» ***tab***[0] « , »
    POUR **i** ALLANT DE 1 à (***size***-2) PAR PAS DE 1
        Afficher ← ***tab***[**i**] « , »
    Afficher ← ***tab***[***size***-1] «] »
FIN

– Fonction entière **rech_max**(entier tab[], entier size, pointeur ptrindice) :

- Déclarations des variables entières :
    memory ← ***tab***[0]
    n ← 1

DÉBUT
    TANT QUE ***n*** < ***size***
        SI ***memory*** <= ***tab***[***n***]
            memory ← ***tab***[***n***]
            pointeur ***ptrindice*** ← ***n***
            n ← ***n*** + 1

        SINON
            n ← ***n*** + 1

    Renvoyer ← ***memory***
FIN

– Fonction Principale
- Déclarations des variables entières
    tableau[] ← {0,12,11,13,15,9,5,4,2,7,8,1,3,10,14}
    longueur ← sizeof(tableau) / sizeof(entier)
    indice

DÉBUT
    Afficher ← **maximum**(***tableau***, ***longueur***)
    Afficher **rech_max**(***tableau***, ***longueur***, adresse de ***indice***)
    croissant(***tableau***, ***longueur***)
FIN

9. Les chaînes de caractères

9.1 Saisie et affichage de chaînes de caractères

Afin de pouvoir récupérer l'entrée utilisateur même si celle-ci contient un espace, j'ai fait le choix d'utiliser la fonction **fgets();**, qui permet de récupérer une chaîne de caractères jusqu'à un « entré », ou « \n ».

- Déclarations des variables de types caractères :
    prénom[10]

DÉBUT
    Afficher ← « Entrez votre prénom »
    Saisir 10 caractères dans le stdin → ***prénom***
    Afficher ← « Votre prénom est : »
    Afficher ← ***prénom***
FIN

9.2 Calcul de valeur d'un résistance

– Fonction vide **affichage**(caractères tab[][7])

DÉBUT
        POUR **i** ALLANT DE 0 à 9 PAR PAS DE 1
            Afficher ← **i**
            Afficher ← ***tab***[i]
FIN

– Fonction entière **isTrue**(caractères tab[][7], caractères text[])
- Déclarations des variables entières :
    comparaison ← 0

DÉBUT
    POUR **i** ALLANT DE 0 à 9 PAR PAS DE 1
        SI string ***tab***[**i**] = string ***text***
            comparaison ← ***comparaison*** + 1

        SINON
            comparaison = ***comparaison***

    SI ***comparaison*** = 1
        Renvoyer ← 1

    SINON
        Renvoyer ← 0

FIN

– Fonction vide **troixAnneaux**(caractères tab[][7], caractères _1Ann[],
caractères _2Ann[], caractères _3Ann[], int E12[])

- Déclarations des variables entières :
    defo1
    defo2
    defo3
    defop ← 0
    val1Ann
    val2Ann
    val3Ann
    taille
    correspondance
    valAnn1_2
- Déclarations des variables caractères :
    st1Ann[256]
    st2Ann[256]
    answer[256]
- Déclarations des pointeurs sur caractères :
    Ann1_2 ← NULL

DÉBUT
    TANT QUE ***defop*** = 0
        defo1 ← 0
        defo2 ← 0
        defo3 ← 0
        val1Ann ← 0
        val2Ann ← 0
        val3Ann ← 0
        taille ← 0
        correspondance ← 0
        TANT QUE ***defo1*** = 0
            Afficher ← "Indiquez la couleur du premier anneau : "
            Saisir 256 caractères dans le stdin → ***_1Ann***

        SI **isTrue**(***tab***, ***_1Ann***) = 1
            defo1 ← 1

        POUR **i** ALLANT DE 0 à 9 PAR PAS DE 1
            SI string ***tab***[**i**] = string ***_1Ann***
                val1Ann = **i**

        Conversion de ***val1Ann*** en string dans ***st1Ann***
        TANT QUE ***defo2*** = 0
            Afficher ← "Indiquez la couleur du deuxième anneau : "
            Saisir 256 caractères dans le stdin → ***_2Ann***

        SI **isTrue**(***tab***, ***_2Ann***) = 1
            defo2 ← 1

POUR **i** ALLANT DE 0 à 9 PAR PAS DE 1
    SI string ***tab***[**i**] = string ***_2Ann***
        val2Ann = **i**

Conversion de ***val2Ann*** en string dans ***st2Ann***
TANT QUE ***defo3*** = 0
    Afficher ← "Indiquez la couleur du troisième anneau : "
    Saisir 256 caractères dans le stdin → ***_3Ann***

SI **isTrue**(***tab***, _3***Ann***) = 1
    defo3 ← 1

POUR **i** ALLANT DE 0 à 9 PAR PAS DE 1
    SI string ***tab***[**i**] = string ***_3Ann***
        val3Ann = **i**

taille ← longueur de la string ***st1Ann*** +1
taille ← ***taille*** + longueur de la string ***st2Ann*** +1
Ann1_2 = (pointeur sur caractères)malloc(***taille*** * sizeof(caractères))
SI ***Ann1_2*** = NULL
    Sortie du programme

Recopie de la string ***st1Ann*** dans la string ***Ann1_2***
Concaténation de ***Ann1_2*** et ***st2Ann***
Conversion de ***Ann1_2*** en entier dans ***valAnn1_2***

POUR **i** ALLANT DE 0 à 12 PAR PAS DE 1
    SI ***E12***[**i**] = ***valAnn1_2***
        correspondance ← 1

    SINON
        correspondance ← ***correspondance***

SI ***correspondance*** = 1
    Afficher ← « Résistance dans la série E12. »
    Afficher ← ***valAnn1_2*** * 10^***val3Ann***

SINON
    Afficher ← « Résistance pas dans la série E12. »
Afficher ← Voulez-vous recommencer ? oui/non

Saisir 256 caractères dans le stdin → ***answer***
SI string ***answer*** = « oui »
    defop ← 0

SINON SI string ***answer*** = « non »
    defop ← 1

FIN

– Fonction Principale

- Déclarations des variables de types caractères
    tabcodecoul[][7] = {"noir", "marron", "rouge", "orange", "jaune", "vert", "bleu",
                                                    "violet", "gris", "blanc"}
    ent1Ann[256]
    ent2Ann[256]
    ent3Ann[256]
- Déclarations des variables de types entières
    tabvalE12[] = {10, 12, 15, 18, 22, 27, 33, 39, 47, 56, 68, 82}

DÉBUT

    affichage(***tabdecoul***)
    troixAnneaux(***tabcodecoul***, ***ent1Ann***, ***ent2Ann***, ***ent3Ann***, ***tabvalE12***)

FIN

10. Structure de données

Les exercices utilisant les structures de données nécessitant la bibliothèque graphiques, je ne les ai donc pas réalisés.

11. Les Fichiers

11.5 Exercice d'écriture d'un fichier texte

- Déclaration des pointeurs sur FILE :
    tableX ← NULL
- Déclarations des variables de types caractères :
    nom1[] ← {"table"}
    nom2[256]
    nom3[] ← {".txt"}
- Déclarations des pointeurs sur caractères :
    nom
- Déclarations des variables entières :
    num ← 0
    resultat ← 0
    taille ← 0

DEBUT
    Afficher « Numéro de la table ? : «
    Saisir → ***num***
    Conversion de ***num*** en string dans ***nom2***

    nom = (pointeur sur caractères)malloc(***taille*** * sizeof(caractères))

    SI ***nom*** = NULL
        Renvoyer ← -1

    Concaténation de ***nom1*** + ***nom2*** + ***nom3*** dans ***nom***
    tableX = Ouverture du fichier « nom » en mode « écriture »

    POUR **i** ALLANT DE 0 à 10 PAR PAS DE 1
        resultat ← **i** * ***num***
        Ecrire dans le fichier ← **i** « x » ***num*** « = » ***resultat***

    Fermeture du fichier ***tablex***
FIN