

13/11/2020

exercice 7

Traduisons l'algorithme de l'exercice en C.

```
#include < stdio.h >
```

```
int main ()
```

```
{
```

```
    int flag = 1;
```

```
    if (!flag)
```

```
        printf (" 1");
```

```
}
```

```
else {
```

```
    printf (" 0");
```

```
}
```

```
return 0;
```

```
}
```

exercice 8

Algorithme Conversion

Variables: entiers: i, a

Début

i ← 5

i ← i + 1

a ← i

i ← i + 1

a ← i

i ← i + 1

a ← i

afficher ("a = ", a)

fin

Frases

Algo i frases tienen que cumplir

Variables:

Caracteres a, b, c

Síntaxis:

Echar ("entier 3 caractères")

Escri (a, b, c)

si (a < b < c) alors

Echar (ordre)).

Fin

Echar ("descendre").

fin

En language C

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main ()
```

```
{ char a, b, c;
```

```
printf ("Ecrire entier 3 caractères");
```

~~scanf ("%c %c %c", &a, &b, &c);~~~~scanf ("%c %c %c", &a, &b, &c);~~

```
if (a < b < c)
```

```
    printf ("ordre");
```

```
}
```

```
else {
```

```
    printf ("descendre");
```

```
}
```

```
return 0;
```

exercice 10

Algorithme Traduction

Varables a, i entiers

Début

$$i \leftarrow 1$$

$$i \leftarrow 2 + 2 * i$$

$$i \leftarrow i + 4$$

ecrue(i)

fin

exercice 11

Algo: Traduction

Var entier i;

Début

$$i \leftarrow 0$$

Repéter

Ecrire "valeur de i", i)

jusqu'à i = 4

fin

exercice 14

Algorithme essai

Var : n entier

Const: CTE

Début

Repéter

Ecrire ("entrer un nbre")

lire(n)

Si ($n < \text{CTE}$) alors

Ecrire ("essayer plus grand")

Sinon

Si ($n > \text{CTE}$)

écrire ("écraser plus petit")
fini
jusqu'à (n = CTB)

2) Traduisons en langage C

exercice 15

1- Écrivons une fonction qui a comme argument un entier n. Calculer de façon itérative la valeur de sa factorielle et renvoyer le résultat.

2^e entier! fonction
* 1^e déclaration de la fonction *

3^e 1^e objectif : Calcul itératif de la factorielle !

3^e 1^e alg : un nbre entier n

4^e 1^e Résultat : $n! = n \times (n-1) \times \dots \times 2 \times 1$

5^e 1^e variable : entier r - 1^e Déclarat des variables

6^e Début

$r \leftarrow 1$

Pour i de 1 à n, faire

$r \leftarrow r * i$

pour

renvoyer (r)

fin

2- Appelons la fonction précédente et écrivons un algo pr le calcul des combinaisons

Algorithme : Combinaison

Var r, k, n ; entiers

Début

Écrire ("entrer deux entiers")

Lire (k, n)

Si ($k > n$) alors

Écrire ("combinaison k du n = 0")

Sinon

$r \leftarrow \text{factorielle}(n) / (\text{factorielle}(k) * \text{factorielle}(n-k))$

Écrire ("la C_n^k = r")

3e Traduction en langage C

include < stdio.h >

include < stdlib.h >

```
int facto (int n) {
```

```
    int r = 1;
```

```
    for (i=1; i<=n; i++)
```

```
        r = r * i;
```

```
    return (r);
```

```
}
```

```
int main ()
```

```
    int n, k, r;
```

```
    printf ("entre deux nbres");
```

```
    scanf ("%d %d", &k, &n);
```

```
    if (k > n) {
```

```
        printf (" la combinaison de %d dans %d  
        est 0", k, n); }
```

```
    else {
```

```
{
```

```
    printf ("Comb. %d dans %d est %d",  
    k, n, facto (n) / (facto (k) * facto (n - k)));
```

```
    return 0;
```

```
}
```

4 - Fonction recursive (voir topo)

exercice 6

- 1) écrire FONCTION fibonaci (entier n)

Variable entiers
Debut

Si $n=0$, alors $r=0$ ✓ On initialise mais on

Sinon si $n=1$, alors $r=1$ fait suivre

Sinon $r = \text{fibonacci}(n-1) + \text{fibonacci}(n+2)$

$f(n)$

return r

fin

- 2) Algorithme qui affiche les n premiers nbres de Fibonacci

Algorithme: suite

Variabiles i, n; entiers

Debut

Ecrire ("entrer un nbre entier")

Lire (n)

Ecrire ("les", n, "premiers nbres de Fibonacci sont")

Pour i allant de 0 à n, faire

afficher (fibonacci(i))

fpour

fin

exercice 17

- 1) Écavoir une fonction qui calcule $n^{\frac{p}{2}}$ 2 façons

Entrée: puissance (entier : n, p)

Variable: entier prdt

Debut

Prdt $\leftarrow 1$

Pour i de 1 à P , faire

Prdt \leftarrow Prdt * n

fin pour

renvoyer (prdt)

fin

b) De façon récursive

Entier_puissance (entier n , p)

variable : entier prdt

Échut

Si $P = 0$, prdt = 1

Sinon $n * \text{puissance}(n, P-1)$

Prdt \leftarrow prdt * puissance($n, P-1$)

fin

fin

2) a) Valeur affichée

i) nbre = 7, b = 2

ii) nbre 3, b = 2

iii) $n \leftarrow \text{nbre} = 7$.

a $\leftarrow 0$

i $\leftarrow 0$

$a + (7 \bmod 2) * 10^0 = 1, n = 3$

$a + (3 \bmod 2) * 10^1 = 11, n = 1, i = 2$

$a + (1 \bmod 2) * 10^2 = 11, n = 0, i = 3$

tant que $a \neq 11$

iii) $n \leftarrow 3 a = 0 + (3 \bmod 2) * 10^0 = 0, n \neq 4, i = 1$

$a \leftarrow 0 a = 0 + (4 \bmod 2) * 10^1 = 0, n = 2, i = 2$

$a = 0 + (2 \bmod 2) * 10^2 = 0, n = 1, i = 3$

$a = 0 + (1 \bmod 2) * 10^3 = 10^3, n = 0$

$a = 10^3$

* Objectif de l'algorithme

Les verbes à certains pas

l'ordre

exercice 13

Construire le programme qui fait ce que demande TOPIC.

- Donner les résultats dans le tableau :

* A: $\emptyset = (N \geq P, N++ \text{ ou } P+3, P++)$
et $N=6, P=3$ $\emptyset = 1, P=2$ (ou)

et $\emptyset = 0, P=3$ (ou) et $\emptyset = 1, P=2$ (ou)

* B: $\emptyset = (N \leq P, N++ \text{ ou } P+3, P++)$

$\emptyset = 1, N=6, P=3$

* C: $\emptyset = +N=3 \text{ et } P+3=3$

$\Rightarrow \emptyset = 0, N=6, P=3$

* D: $\emptyset = N+1=6 \text{ et } P+1=3$

$\Rightarrow \emptyset = 1, N=6, P=3$

exercice 14

exercice 15

1) Écrire une fonction qui calcule le pgcd de 2 entiers.

entier Fonction pgcd (entiers a, b)

Variable r: entier

statut si $a \neq b = 0$, alors

```

r ← b
fonction
    r ← pgcd(a, b)
    fin
    renvoyer(r)
fin

```

* Si on factorise l'itération, on aura

celle-ci fonction pgcd (entiers a, b)

variable : r entier

rept debut

 répéter

 r ← a mod b

 a ← b

 b ← r

 tant que r ≠ 0

 fin répéter

 renvoyer(r)

 fin.

2) Algorithme : mdpremiers

variables a, b, r; entier

rebut

écrire ("entier > entier > positif")

lire (a, b)

r ← pgcd(a, b)

si ($a = 1$) alors

 écrire ("a est premier")

sinon

 écrire ("il ne faut pas")

 non premier

fin

exercice 20

4) écrivons cette fonction

en C : fonction (caractere *ch, entier n)

Var P, i, j : entiers

Tcaractere = T[5] de caractere

Ttray = Tab[5][5] de caractères = T[5][5]

Debut

P ← 0

Pour i de 0 à (n-1) faire

Pour j de 0 à 4, faire

Si ch[i] = Ttray[i][j] alors

P ← P+1

faisi

f pour

f pour

fin

Programme C

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int nbtry( char *ch, int n)
```

```
{
```

```
    int P, i, j;
```

```
    char Tab[5][5];
```

```
    P = 0;
```

```
    for (i=0, i<n, i++)
```

```
        for (j=0, j<4, j++)
```

```
            if (ch[i] = Tab[i][j]) {
```

```
                P = P+1;
```

```
} } } return P;
```

ant main ()

int P, i, j ;

char *ch ;

Printf ("entre une chaîne de caractère"),

scanf ("%s", ch);

printf ("le caractère de l'index %d est, %c, %c (%d))

return;

}

exercices

1)

Caractère fonction plage (caractère : Ch, nombre : C)

Variable j, i : entier, Ch : entier n

Début : C

Pour i de 0 à n-1 faire

 joi ch[i] = "0"

 afficher

 Pour j à 0 à i-1,

 Cj ← ch[j] - fpr

 Pour j de i à n

 Cj ← ch[j]

 ffpour

 écrire(C, C, C)

 fin

 tirette ("pas trouvé")

 ffoi

 ffpour

 fan

Sous forme de pointeurs (D'abord la fin de la liste)

exercice 1

2) Ventre : P

2) Véhicule : P 3) Générateur : P

exercice 2

Le tableau affiche :

exercice 2

1. 5

2. DXIA52

3. 0 XIA44

4. 5 *

Si le pointeur des mots est mis à zéro, Q) il n'y a pas de

5. DXIA164 *

Car il est un pointeur qui pointe vers une variable non

6. DXIA44 *

D * XIA52

7. 8

92

exercice 3

Algorithme :辗转相除法

Variable

↓ en tiers P, q

entiers a, b

Réultat

a ← 45

b ← 15

P ← @a

q ← @b

Exercice ("Résultat": 15, "P": 15, "@P", "@q")

Étape: ("a": 45, "b": 15)

fin

exercice 4

Donnons le type et la valeur des variables

variables:

1) Var1 = tab[1]:

2) Résultat

	Type	Valeur
1)	Pointeur	0x1A40
2)	Entier	4
3)	Entier	3
4)	Pointeur	0x1A156

exercice 5

Donner en illustrant les affichages de l'algorithme suivant. (cas d'un type)

Il affiche

- *après f_1 * : 3

- *après f_2 * : 3

- *après f_3 * : 3

exercice 6

L'exercice de faire ce travail pour chaque type. Ici, on traite le sav (cas

cas double

RAPPEL (sur les types)

Consulter continuellement la table 1-15 (Page

Algorithme: affichage

Const: MAX entier

Tab[MAX]: entier, entor: n

Début

Ecrire "Entrer le nombre maximal de
valeurs pour ce type")

Etre (MAX);

Tab ← Max.

Ecrire ("entrez un nombre strictement
positif et inférieur à MAX");

Etre (n);

Affection (← T-fab (n) , T-fab (m)) ;
← (a + T-fab (m)) - (a + T-fab (n))

En primer orden

exercice 7

Algorithme : exercice 7

Type tTab = tableau C à 3 entiers

VentTab : Tab[3] au d'entiers

Variables : tTab : Tab

entier n, nombre, i, p

Début

Ecrire ("Entrer le nombre d'elts")

Lire (n)

tTab \leftarrow allouer ($n * \text{nb octets} (\text{entiers})$)

Ecrire ("Entrer les "n" entiers positifs")

Pour i $\leftarrow 0$ à $n-1$, faire

Lire (nombre)

Tab[i] \leftarrow nombre

fin pour

Ecrire ("Entrer un nbre compris entre 1 et n")

Lire (p)

Ecrire ("Le ", p^e "ème élément entré est", Tab[p-1])

desallouer (Tab)

-fin-

End language C

int main

{ int *Tab = NULL, n=0, p=0, i=0;

printf ("Entrer le nbre d'elts");

scanf ("%d", &n);

Tab = malloc (n * sizeof(int));

printf ("Entrer les %d entiers positifs", n);

for (i=0, i<n, i++)

{ scanf ("%d", Tab+i); }

Exercice 3

1) Trouver une formule qui calcule C_p par la
formule $C_p = C_{p-1} + C_{p-1}$
utilisant fonction Combinatoire (entiers p, n)

Var

Debut

Si $P = n = 0$, afficher 1 et arrêter

afficher si

Sinon

renvoyer (Combinatoire (P-1, n-1) + Combinatoire
fin

(P, n)

2) Algorithme triangle de Pascal

Var n, P, ~~i~~ entiers

Debut

↓ ↓ entier tTab

Debut

lire (Entier un nbre)

.lire (n)

tTab ← allouer (n^* nboctet (tEntiers))

Pour $i \leftarrow 0$ à $n-1$, faire

 tTab[i] ← allouer ($(i+1)^*$ nboctet (tEntiers))

 tTab[i] ← Combinatoire (i, n)

fin

Pour $i \leftarrow 0$ à $n-1$, faire Pour i de 0 à n-1

 desallouer tTab[i]

 Pour j ← 0 à $i-1$, faire

 tTab[i][j] ← C_i^j

 fin

 desallouer tTab

fin

fin

exercice 10

(G machine)

exercice 11

Ces lignes des erreurs

- ① L'erreur se trouve au niveau de la 3^e ligne au niveau de l'allocation dynamique de p.

On devrait faire : $p = \text{malloc}(10 * \text{sizeof}(\text{int}))$
 ↳ 2^e erreur
 3^e ligne: ~~$\star p = 1$~~

② i) Le nbre d'elts du tableau n'est pas déclaré

ii) Les Elts du tableau ne sont pas entre accolades

↳ 2^e erreur : tab doit être déclaré avec un ptr à la 3^e ligne

③ Le problème se trouve dans la boucle for, C++ vient après "i <= 100".

④ Le seul souci est dans le if car le C ne reconnaît pas b < c, il faut écrire "abc < bc"

exercice 12

Donnez le résultat sans exécuter, expliquer son algorithme

* Résultats sans exécution

1) $\text{for}(j=0; j < 3; j++)$

$$\text{tab}[j] = 5$$

2) Après cette boucle, $\text{tab}[0] = \text{tab}[1] = \text{tab}[2] = 5$

$$*(\text{tab} + 1) = 3$$

3) $\text{tab}[1] = 3$

$$*(\text{ptr} - 1) = 3$$

4) $\text{tab}[2] = 3$

Donc $\text{tab}[0] = 5$, $\text{tab}[1] = 3 = \text{tab}[2]$

→ L'output donne $[5 3 5]$

Q) La je ne savais pas quoi, après m'élémuration,
je p trouve bc

③ La je n'ai pas compris mais la machine affiche 5 et 2

④ Le programme n'affiche rien

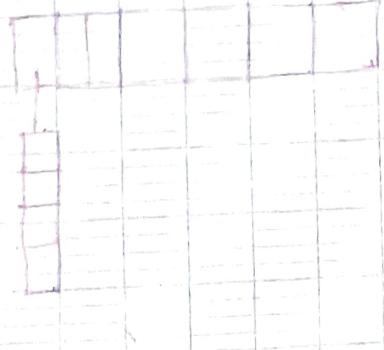
⑤ 36 0 5

⑥ Un bug

Ecrive les algo ④

- * Commente les points
 - Les entiers sont à l'origine, adresses et contenus
 - Les pointeurs
- ① P : adresses de P = tout qui contient P
- P = On contient de P ce qui a l'adresse n
- ② P = n : le contenu de P est l'adresse de n

Contenu des pointeurs



0x10 A

exercice 13

- Fonction qui initialise les 2 matrices
- Type def fonction initialiser (~~meilleur~~ T₁[n][n]
- debut
 - Pour i ← 0 (1) n-1 faire
 - Pour j ← 0 (1) n-1 faire
 - écrire ("entrer les valeurs t_{ij})
 - lire T₁[i][j]

fpr

fin

- fpr
- Pour i ← 0 (1) n-1 faire
- Pour j ← 0 (1) n-1 faire
- Renvoyer T₁[i][j]

fpr

fpr

fin

Type def fonction produit (entier $T_1[n]$, $T_2[n]$)

Varijken tels
Debut

~~for i = 0 to n-1 faire~~

Pour $i \leftarrow 0$ à $n-1$ faire

Pour $j \leftarrow 0$ à $n-1$ faire

Pour $k \leftarrow 0$ à $n-1$ faire

$T[i][j] = T_1[i] T_2[j] * T_3[k]$

fpr
fpr

Renvoyer T

fin

Type def fonction afficherage ($T_1[n]$: réels)

$T_2[n]$

Debut

Renvoyer produit ($T_1[1:n], T_2[1:n]$)

fin

* Implementation