Bases de la programmation impérative

Cour 2

Constantes littérales numériques :

Constantes valides :

123
+123
-123
0,18 (base 8) en octal
0x 15 hexadécimal

base 8
base 16

Réels s'écrivent sous ce format :

$-123.45e+6f$

- e : puissance
- f : float codé uniquement en mémoire 4 octets

Constantes valides :

- 3.14
- 3.
- .14
- .14e12
- 3.e-2

occupent 8 octets par chaque valeur

- 3e4f

occupe 4 octets

float codé en 4 octet

double codé en 8 octets

Constantes littérales en caractères :

Constantes valides :

- 'a'
- ','
- 'A'

1 caractère par quote(' ')
Cela occupe 1 octet en mémoire

non valides

- 'c
- c'
- 'ac'

Quelques Caractère Spéciaux :

- '\t' (tabulation)
- '\n' (retour a la ligne)
- '\"' (guillemets)
- '\'' (simple quote)
- '\0' (caractère nul) ⚠ important

Chaînes de Caractères :

"abc"

" ceci est une phrase"

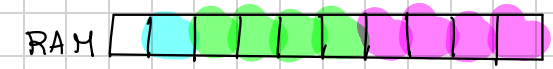
"ceci est une phrase \n sur deux lignes."

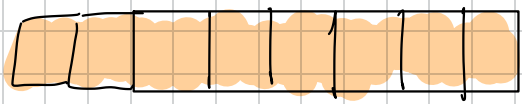
variable : Zone mémoire dans laquelle on peut mettre ce qu'on veut

⚠

une variable est définie par :

* un type : Char, int, float, double (8 octets)





* une adresse mémoire
* un identificateur : code source (nom pour accéder à l'adresse)
* une valeur que l'on peut modifier à souhait.

Les identificateurs de variable répondent aux règles strictes :

- peut contenir tout les caractères alphabétiques
- 1 2 2 7 / les chiffre
- / / / / que des tirets bas

Ne commence jamais par un chiffre ni par tiret bas_
sensible à la casse (majuscule/minuscule)

exemples : k | non valides : 2A
une_variable | a b
Variable2

- Pour choisir les identificateurs on essayera de donner des noms explicites

langage C est un langage déclaratif.

- type nom de l'identifiant ;
- int ℕ ;
- float réel ;

instruction d'affectation :
car = 'salut'; // commentaire
k = 123;
reel = .e-4f;
réel double = 3.14;
/* pour les lignes de commentaires */

Les pointeurs :) variable

variable dont la valeur est une adresse

&a désigne adresse de variable a

déclare : int *P;
S'initialise : P = &a;

*P = a

Conv 3

les pointeurs = variable

déclare : int *P;

S'initialise : P = &a;

*P = je vais voir l'adresse de P

Ce vers P est a alors c'est ce que j'allais à l'adresse de a

*P = 2 et a = 2

Portée des variable :

Dans un même bloc d'instruction on peut pas donner le même nom à toutes les variables. Donc pour cela on déclare le même nom dans des bloc différent donc 2 variable différente mais qui ont le même nom. donc en mémoire 2 adresse différentes

Variable constante :

const double Pi = 3.14159265;

Expression arithmétiques :

- constantes
- Variables
- opérateurs arithmétiques (+, -, *, / ou %)
  - % → module reste de la division entière
- parenthèses

double > float > int

1 / 2 = 0 pck entier
1. / 2 = 0.5 double

2. 2.5 (2. + 2.5) * 2

4.5 * 2 = 9.

Exemple d'affectation

int i, j; j = 2; i = 2 * j;

float f; f = 3.14f; f = f + f;

Expressions Booléennes: (Peut de faire du programme qui fait bcp d'choses)

Soit vrai soit faux

opérateur de comparaison:

== dit si c'est égal ou pas

< j > inférieur

<= inférieur ou égal

>= supérieur égal

et != différence

! négation (contraire de)

&& et

|| ou

1. ! negation
2. <, >, <=, >=
3. == et !=
4. &&
5. ||

} ordre de priorité

exemple :

1 < 2

a == 2

a <= b

k != j

a == 1 && b == 1

Diapo 31 ⚠

Les entrées / sorties : (Standard) ⇒ (clavier/écran)

Printf ("écrire une phrase \n");

diapo 34.

char (x), c = A; { Printf ("... %c";

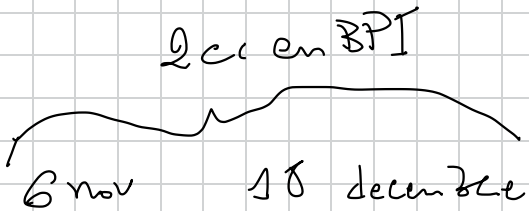
Char 'A'

Char "Hello"

printf("le mot est : %c", 'A');

Char a = "A"

2cc en BPI



1 note de TP

Examen TP en fin de Semestre + Travail au cours du Semestre

TD commencent le 11

TD 1 BPI

Exercice 1.1.

1) la formule représente le volume d'une sphère

2) donc il y a 2 opérations de multiplication

3) les constantes : $\frac{4}{3}\pi$ } variable : R }

$\frac{4}{3}$ | π | R^3

Exercice 1.2 : $x^2 + 2ax - 3a^2 = 0$

1)

$$x_1 = \frac{-2a - \sqrt{(2a)^2 - 4 \times 1 \times -3a^2}}{2}$$

$$x_2 = \frac{-b + \sqrt{(2a)^2 - 4 \times -3a^2 \times 1}}{2a}$$

$\Delta = b^2 - 4ac$

$\Delta = (2a)^2 - 4 \times (-3a^2)$

$\Delta = (2a)^2 + 12a^2$

$\Delta = 4a^2 + 12a^2$

$\Delta = 16a^2$

$$x_{1,2} = \frac{-2a \mp \sqrt{16a^2}}{2a}$$

$$x_{1,2} = \frac{-2a \mp 4a}{2}$$

$x_1 = \frac{-6a}{2}$

$x_1 = -3a$

$x_2 = 1a$

2) addition/multiplication / soustraction

3) $2a$; $-3a^2$ } constante x^2 ; x } variable

2/

Exercice 2.1:

1/ Peuvent contenir des lettres et chiffres, caractères simple pas de cédilles, d'accents, d'espace, et peut coder des textes bon ne commencant jamais par une majuscule et suivie a la casse.

2/

3/ ordre de priorité des opérateurs:

1 - parenthèse : ( )
2 - negation booléens !
3 - multiplication, Division, modulo *, \, %.
4 - addition soustraction
5 - relation <=, >=, >, <
6 - comparaison == et !=
7 - et logique &&
8 - ou ||

Langage C / Algorithmique

Exercice 1.1:

$$V = \frac{4}{3}\pi R^3$$

1/ volume d'une sphère/boule

Sphère = Surface qui englobe un volume

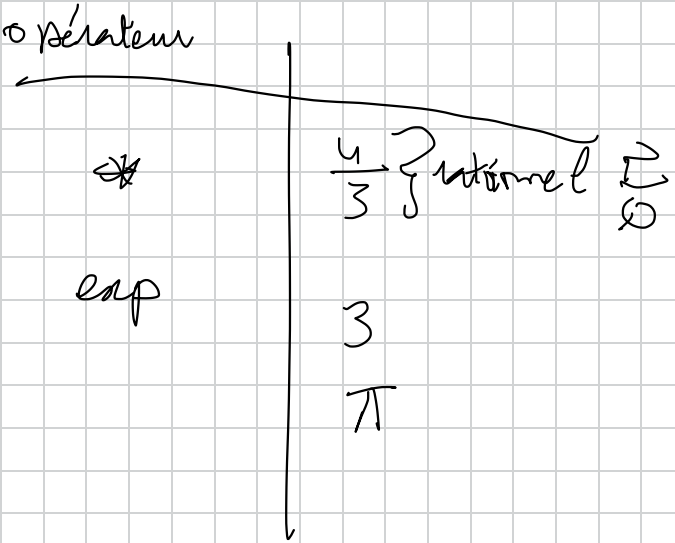
boule = volume

2/

opérande + opérateurs = formule

multiplication, exponentiation

"=" n'est pas une relation dans ce cas de figure



$\frac{4}{3}$ coté domaine des rationnels / π coté domaine $3,14\overline{159}\ldots$ / 3 entier

$1,7\overline{3}$ / 3

opérateurs
expo

addition
soustraction
multiplication

paramètre

$x^2 + 2ax + 3a^2 = 0$

constantes

2 } entiers
-3 }

Variables

a est une variable du domaine de $\mathbb{R}$

x n'est pas une variable attention

2) Variables, type, expression, instruction d'affectation :

- nom de variable commence toujours par lettre maj ou min ou underscore (_)

  Kamel Case (moyenneClasse)

- types de constantes :

Constantes numériques : entier (+4, -47, 5)

Flottants (2.0, .0, 0., 2.e3)

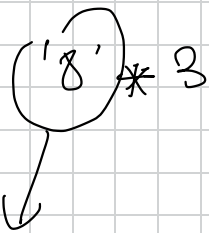
constantes de types caractère 'a'

constante littéral chaîne de caractères.

[+ -] [0-9]* [. [0-9]*] [e[+ -] [0-9]+]

facultatif

obligatoire



Caractère entier code Huskit 16,

ordre de priorité des opérateurs

$(3 * 4 / 5 / 3 - 5 / 3 / 3$

$2,4 / 3 - 1,6 / 3$

0 — 0

1 ( )

2 !

3 *, /, %

4 +, -

5 <=, >=, <, >

6 == !=

7 &&

8 ||

Exercice 2.5 :

Maths $\left\{ x \in [2,6[\cup [8,11] \cup]14,\infty[\right.$

$2 \leq x < 6 \wedge 8 \leq x \leq 11 \wedge x > 14$

C $\left[(x \geq 2 \text{ ss } x < 6) \text{ ll } (x \geq 8 \text{ ss } x \leq 11) \right.$

ll $(x > 14)$

$y \;]-\infty ; 2[$

$x \;]-\infty ; 7[$

$x \cdot y \quad (2;7) \;]+\infty, +\infty[$

$x \cdot y \quad (2; +\infty) \text{ et } (7; +\infty)$

$(y < 2 \text{ ss } x < 7)$

ou bien

$(y > 2 \text{ ss } x \geq 7)$

Cas 2 où on doit juste dire
ça peut prendre des valeurs.

$$\text{maths} \begin{cases} x \leq 1 \text{ \&\& } x \geq -4 \\ \text{\&\&} \\ y \leq +3 \text{ \&\& } y \geq -2. \end{cases}$$

C
$$\begin{cases} (x <= 1 \text{ \&\& } x >= -4) \\ \text{\&\&} \\ (y <= 3 \text{ \&\& } y >= -2) \end{cases}$$

des entrées sorties : (Il existe des modificateurs de format)

"%.Tf" lest T caractères.

"%.zf" e chiffres après la virgule

"%0Tf" remplace les espaces par des 0

```
int c;
Scanf("%c", &c)
```

↓ adresse de variable

```
int nb = Scanf("%d%f", &k, &f);
```

```
#include <stdlib.h>
<<, <stdio.h>
int main() {
float k;
int n = Scanf("%f", &k);
Printf("voici la valeur de retour scanf : %f", n);
return 0; }
```

Fonctions

fct permet d'exécuter à la demande un bloc d'instruction.

- types de fct :
  - fct avec retour de valeur
  - fct sans retour de val

type de retour nom de fct (type1 param1, type2 param2 ...) {
  // instructions
  return 0;
}

void nom de fct (type1 param1, type2 param2 ...) {
  /* instruction */
}

fonction Min : minimum

int min (int x, int y) { → objet change variable dans paramètre
  int m;
  if (x < y) {
    m = x;
  } else { m = y;
  }
  return m;
}

Exercice 2.6:

Soit x, y, z, 3 variables, exprimez sous la forme de 2 expression booléennes, la 1ère en maths, la 2ème en langage C les proposition suivantes.

1/ les valeur de x, y, z sont dans l'ordre croissant.

$x < y < z$

$x < y \ \&\& \ y < z$

2/ Les valeurs de x, y, z sont tout strictement superieurs à 0

$x > 0, \ y > 0, \ z > 0$

$x > 0 \ \&\& \ y > 0 \ \&\& \ z > 0$

3/ d'une au moins des valeur est ≥ 0

$x \geq 0 \ || \ y \geq 0 \ || \ z \geq 0$

$x >= 0 \ || \ y >= 0 \ || \ z >= 0$

4/ Les valeurs de x, y, z sont toutes les 3 différentes.

$x \neq y$ et $y \neq z$ et $x \neq z$

$x != y \ \&\& \ y != z \ \&\& \ x != z$

Exercice 2.7:

1/ j'ai faim && !=soif && != j'ai sommeil 3/ faim && soif && Sommeil

2/ j'ai faim && j'ai soif != j'ai Sommeil 4/ Voir page suivante

5/ (j'ai faim && j'ai soif) || (j'ai faim et j'ai soif) || (j'ai sommeil et j'ai soif)

4/ (j'ai faim || j'ai soif || j'ai sommeil)
(j'ai faim && j'ai soif && j'ai sommeil) ||
(j'ai faim && soif) || (j'ai faim && sommeil || (j'ai soif et sommeil)

6/ j'ai faim || j'ai soif || j'ai sommeil

7/ (j'ai faim && j'ai soif) || (j'ai faim && j'ai sommeil) |
(j'ai soif && j'ai sommeil)

8/ (!= j'ai faim) && (!= j'ai soif) && (!= j'ai sommeil).

9/ j'ai faim || j'ai soif || j'ai sommeil || (j'ai faim && soif) || (j'ai faim && sommeil) || (j'ai soif && sommeil)

Exercice 2.8:

int x, n; bloc 1

int x; int n; } les variables x et n sont declarés

x++ ++x x+=1 } x = x+1

trace d'execution:

x = 5; // x prend la valeur de 5

n = 3; // n prend la valeur de 3

--x; // x est decrementé donc on affiche 1 a la valeur de x le dépassement

x = -2 // on affecte la valeur -2 a x.

double x ;

double z ;

double y ;

x = 4.5 ; la valeur de x s'initialise à 4,5

z = 3.0 / 2 ; la valeur de z s'initialise à 1.5

y = -x ; la valeur de y est l'opposé de celle de x donc -4,5

++y ; la valeur de y est incrémentée en ajoutant 1 à la valeur de y précédente donc -3,5

Exercice 2.9 :

1) x = x + y ; x = 12 y = 5 → x = 12+5 = 17

y = x - y ; y = 17 - 12 = 5 y = 5 x = 17

x = 17 y = 12

x = x - y ; x = 17 - 5 = 12

x = 12 y = 5

2) oui, cette fonction fonctionne tant que les valeurs de x et y restent dans la machine par positif

Exercice 2.10 :

$$f_{n-1} = F_n + F_{n-1} \quad si \geq 2$$

$F_0 = 0$

$F_1 = 1$

$F_3 = F_2 + F_3$

$F_n = F_{n-1} + F_{n-2}$

U, V

U = 1	U = 2	U = 3
V = 0	V = 0	V = 1

F_n P_{n-1} F_{n+1} F_n

U V U V

U = 1

F_{n+1} F_{n-1}

U V

V = 0

Exercice 2.11

il faut juste mettre la fct temp (temperature) pr swap les valeurs

fait sur PC

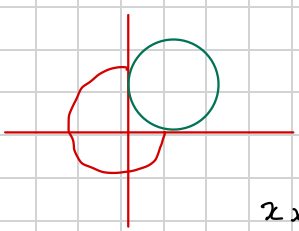
Exercice 2.15 :

Cas 1 :

$x^2 + y^2 = 4$ Pts appartenant au cercle

$x^2 + y^2 <= 4$ $x\hat{}2 + x\hat{}2 <= 2\hat{}2$

Cas 2 : $x*x + y*y <= 4$



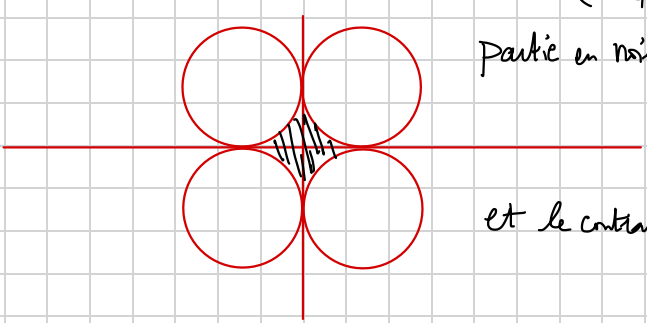
$x^2 + y^2 \leq 4$

et

$(x-2)^2 + (y-2)^2 \leq 4$

$x*x + y*y <= 4$ && $(x-2)*(x-2) + (y-2)*(y-2) <= 4$

Exercice Maison :



Partie en noir

et le contraire

à faire

(Suite) CM 20/09

Les fct ne retournant aucune valeur:

Lundi
CC2 6 novembre

Mercredi
CC2 20 décembre

```
Void affiche_moyenne (float x, float y) {
float m,
  m = (x+y)/2
  printf ("%f", m);
}
```

Pas besoin de return.

fct min.

```
int min (int x, int y) {
int m;
  if (y<x) {
     m = x;
  } else {
     m = y;
  }
  return m;
  }
```

fct qui retourne des valeurs
ne contient pas d'affichage.

Les fct passage des paramètres par adresses.

fonction echange en passant les valeurs des variables

TD 3 (Suite)

Exercice 3.2 :

le programme affiche :

%ff : --3.1416 } un double avec 4 chiffres après la virgule

%c : ; } le caractère est un virgule

%d : -131

le Printf va donc afficher :

vous avez entré 3.1416, -131

Exercice 3.3 :

```
int main (void){
  float variable1, variable2, variable3;
  printf("Veuillez entrer trois valeurs pour trois variables distinctes:");
  scanf("%f %f %f", &variable1, &variable2, &variable3);
  float moyenne;
  moyenne = (variable1 + variable2 + variable3)/3;
    printf("la moyenne est de : %f", moyenne);

  return 0;
}
```

```
bool Majuscule (char c) {
    return c >= 'A' && c <= 'Z';
}

bool Lettre (char c) {
    return (c >= 'A' && c <= 'Z') ||
           (c >= 'a' && c <= 'z');
}
```

Exercice 3.4 :

Sur le PC

4/

Exercice 4.1 :

int carré (int x); on affecte à x la valeur 2 dans le programme appelle le fct int carré et l'entier dans int main

Printf numéro 1 : $2^2 = 4$

Printf numéro 2 : $3^2 = 9$

Exercice 4.2 :

(1)

1/ double fmod (double x, double y) cette fct accepte 2 argum x et y de type double, elle renvoie la valeur de x et y

void (srand unsigned int seed); génère valeur aléatoire avec graine spécifique

int rand (void); génère valeur aléatoire entre zéro

2/ double fmod renvoie un double

int rand (void) renvoie un entier

(2)

long long labsolue (long x) {

if (x < 0) { return -x; }

else { return x; }

```
void parimpar (int x) if(x%2==0)
{ printf("entero par"); } else { printf("impar"); }
```

3637 37 37,00

1 0 37 1h 01 min 40s

3637	3600
37	1

Exo 4:

heure minute seconde

1 h 01 min 01 s

1 h 01 min 02 s

37	60
37	0

1 h → 3600 s

$1 s = \frac{1}{3600} h$

Si seconde > 60 ⇒ seconde = 0

minute + 1

if minute < Γg

seconde = Γg

minute = minute + 1

Si Seconde = Γg

minute < Γg

Seconde = 0

minute = minute + 1

if seconde = Γg, mn = Γg

Seconde = 0

mn = 0

heure = 0

if Sec < Γg

Seconde = Sec + 1

minute = minute

heure = heure

Si secondes = 59 , minutes < 59 , heures < 24

Secondes = 0
minutes = minutes + 1
heures = heures

Si Secondes < 59 , minutes < 59 , heures < 24
Seconde = secondes + 1
minutes = minutes + 1
heures = heures

Si Secondes = 59 , minutes = 59 , heures < 24

sec = 0
minute = 0
heure = heure + 1

Si secondes > 59 , minute, heures
Erreur

si secondes > 59, minute > 59
Erreur

Si secondes > 59 , minutes > 59 , heures > 24
erreur

Si secondes

if s = 59
m = m+1

if s = 59 , m < 59

m = m + 1

15 = 10 + 5
5 = 10 - 5

x = y + 5
x = 5 + 5

y = 15 - 10 = 5

3,15

1 pièce 2 euros
1 pièce 1 euro
1 pièce 10 centimes
1 pièce 5 centimes

6 { 5 centimes
10 centimes
20 centimes
50 centimes
1 euro
2 euros

2 { 2 centime
1 centimes

6 pièces différents au total

5€ 5 centimes

5€ = 2+2+1 prix : x

5 centimes : 10 + 5

5/2 = 2

5 % 2 = 1

5	2
1	2

5 5/10 = 1

(55 % 10)/10

pièce de 2 : prix/2

pièce de 1 : (int) prix - prix/2

pièce de 10 : prix -

Structure conditionnelle :

```
if ( expression booleene ) {
        /* instruction si expression vraie */
}
```

Structure condition avec Alternative :

```
if ( expression booleene ) {
        /* instruction vraie */ }
else { /* instruction si c'est faux */
}
```

Structures itératives (boucles) :

2 types pour l'instant, boucle for et while.

Boucle for : (Suite par exemple) ⇒ Calcul du n'ième terme

```
for ( int k = 1 ; k <= 10 ; ++k ) {
        /* instruction */
}
```

```
for (int k = n; k > 0; --k) {

}
```

boucle while :

```
while (expression booléenne) {
    /* instructions */
}
```

```
while (k != n) { k = k + 1; }
```

Exercice 4.8 : TD

Etape d'exécution	Valeur de a	Affichage
1 (ligne 7)	2	2
2 (ligne 17)	2	2
3 (ligne 16)	1000	1000
4 (ligne 18)	1000	1000
5 (ligne 10)	2	2

ligne	a	mille int(a)	Affichage
4	?	/ /	/ /
7	a = 2	/ /	/ /
8	a = 2	/ /	2
9	1000	1000	/ /
10	/ /	/ /	1000

Exercice 4.9 :

a et b = 0

lorsque "incrementx" est utilisée a = 1
et b = *y = 1

donc la fct incrementx montre

1 1

Après la fct increment a été mis à 0 car elle n'a pas été changée par incrementx mais b si

b = 1

l'affichage est : 0 1

donc on aura : 1 1
0 1

Exercice 4.10 :

```
int a, b, c;

a = 3;
b = -8;
c = 12;
```

(3 -8 12) Printf 1

intervention de la fct modifie.

```
int t = x;
int *v = 3;
```

x = *y = -8 *z = t = 3

*y = *v

x = -8 y = 3 z = 12

(-8 12 3) dans la fct modifier

Apres la fct modifie le printf fait

(3 -8 3)

Car a a été changé mais b et c non car ils n'ont pas été passés en adresse.

Exercice 4.11 :

- int concat (char* dest, char *Src)

Cette fonction appellée concat prend en charge 2 variables en paramètres de type pointeurs vers un caractère et renvoie un entier

- void move (char *dest, char *Src, size_t n);

cette fonction appelée move prend en compte 2 paramètres chacun de type pointeurs et ne renvoie rien

- char *int2str (int n); renvoie pointeur vers caractère

C'est cette fonction convertie un entier en chaine de caractère. en utilisant un pointeur

- long strtol (char *nptr, char **endptr, int base);

c'est fct strcpy prend en charge, 3 paramètres

1 char de type pointeur, 1 char de type pointeur de pointeur de caractère

Variable de sortie

et un entier base

* = pointeur ** = pointeur de pointeur

2/

- void Seconde (int *secondestotales, *heure, *min, *sec)

{

*heure = *secondestotales / 3600;

*secondestotales = *heure % 3600;

*minute = *secondestotales / 60;

*secondes = *minute % 60; }

- void incrementé (int *x) {

*x += 1;

}

- void echange (float *a, float *b)

{ float temp = *a;
a = b;
*b = temp; }

- void quotient (int *a, int *b, float* reste)

{ float quotient = (float)(*a)/(float)(*b)
int quotient = (*a) / (*b)
*reste = (int) quotient - (float) quotient }

Exercice 4.12 :

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

void ajout3 (int *x);

int main (void){
int a;
a = 5;

ajout3 (&a);
printf("%d", a);

return EXIT_SUCCESS;
}

void ajout3 (int *x){
*x += 3;
}
```

Exercice 4.13 :

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

typedef int element;
void echange (element *x, element *y);

int main (void){
element x;
element y;
x = 10;
y = 5;
echange (&x, &y);
printf ("Apres echange : %d %d", x, y);

return EXIT_SUCCESS;
}



void echange (element *x, element *y){
element temp = *x;
*x = *y;
*y = temp;
}
```

Exercice 4.7

```c
int pieceMinimales(double somme) {
    int sommeEnCent;
    sommeEnCent = (int)(somme *100);
    int p2Eur, p1Eur, p50C, p20C, p10C, p5C, p2C, p1C;

    p2Eur = (int)(sommeEnCent / 200);
    sommeEnCent = sommeEnCent % 200;

    p1Eur = (int)(sommeEnCent / 100);
    sommeEnCent = sommeEnCent % 100;

    p50C = (int)(sommeEnCent / 50);
    sommeEnCent = sommeEnCent % 50;

    p20C = (int)(sommeEnCent / 20);
    sommeEnCent = sommeEnCent % 20;

    p10C = (int)(sommeEnCent / 10);
    sommeEnCent = sommeEnCent % 10;

    p5C = (int)(sommeEnCent / 5);
    sommeEnCent = sommeEnCent % 5;

    p2C = (int)(sommeEnCent / 2);
    sommeEnCent = sommeEnCent % 2;

    p1C = (int)(sommeEnCent / 1);

    int nombredepieces;
    nombredepieces = p2Eur + p1Eur + p50C + p20C + p10C + p5C + p2C + p1C;
    return nombredepieces;
}
```

Exercice 4.11 :

Q1 :

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

void convertirTemps(int *duree,int *heures,int *minutes,int *secondes) {
    *heures = *duree / 3600;
    *minutes = (*duree % 3600) / 60;
    *secondes = *duree % 60;
}

int main(void) {
    int duree;
    int heures, minutes, secondes;
printf("saisissez une duree en secondes :");
scanf ("%d", &duree);

    convertirTemps(&duree, &heures, &minutes, &secondes);
    printf ("l'heure est : %02d:%02d:%02d", heures, minutes, secondes);
    return 0;
}
```

Q2

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

void incremente(int *x);

int main (void){
int y;
printf("Entrez une valeure entiere :");
scanf("%d", &y);

incremente(&y);
printf("la valeure incrementee est : %d", y);


return EXIT_SUCCESS;
}


void incremente(int *x){
++*x;

}
```

Q4

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

int quotient(int a, int b, int *reste);

int main(void) {
    int entier1, entier2, reste;
    printf("Donnez 2 entiers: ");
    scanf("%d %d", &entier1, &entier2);

    int q = quotient(entier1, entier2, &reste);

    printf("Quotient: %d, Reste: %d\n", q, reste);
    return EXIT_SUCCESS;
}

int quotient(int a, int b, int *reste) {
    int quotient = a / b;
    *reste = a % b;
    return quotient;
}
```

Q 3

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

int change (float *x, float *y);

int main (void){
float a,b;
printf("donnez deux valeures pour des reels a et b :\n");
scanf("%f%f", &a, &b);
change (&a, &b);
printf("la valeure de a apres echange est :%6.3f la valeure de B apres echange est : %6.3f", a,b);
return EXIT_SUCCESS;
}


int change (float *x, float *y){

float temp = *x;
*x = *y;
*y = temp;

return *x,*y;}
```

Exo 1.1

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

int bidule (int x, int *py);

int main (void){

int a, b;

printf ("Donnez une valeure :");
scanf ("%d", &a);

bidule (a, &b);

printf ("b est : %d", b);
return EXIT_SUCCESS;
}

int bidule (int x, int *py)
{if (x <= 2 && x >= -2){*py = 1;}
else if (x == 10){*py = 2;}
else {*py = 3;}
return *py;
}

/* question 1 = Y = 15, Y = 107, Y = 100 */
```

Exo 1.2

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

int main (void){




return EXIT_SUCCESS;
}

1) if (a == 0) {printf("a est nul");}

2) if (a == 5 || a == 10) { a = a*2; }

3) if ((5 < a) && (a < 10)) { a = 2*a; }

4) if (a > 5) { a = a + 1;
                b = b+1;}
           else
              {a = a-1;
              }
```

Exercice 1.3 :

1)

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

int erreur;
int ausi ( int a, int b, int *presult, int *pause );
int main (void);
```

2)

```
int ausi ( int a, int b, int *presult, int *pause ) {
    int nbr de modif = 0;
if ( c == '+' || c == '-' || c == '*' ) {
        nbr de modif = 1;
        if ( c == '+' ) {
            *presult = a + b;
    } else if ( c == '-' ) {
    *presult = a - b; }
    else ( c == '*' ) {
        *presult = a*b; }
    else if ( c == '/' ) {
```

if (b != 0) { nbre modif = 2;
*pquot = a / b;
*preste = a % b; }

else { erreur = -1; }

} else { erreur = -2; }.

return nbre de modif; }

3/

```c
int main(void) {
    int v1, v2, result, aux;
    char op;

    erreur = 0;

    printf("Entrez l'opération (format : v1 op v2) : ");
    if (scanf("%d %c %d", &v1, &op, &v2) != 3) {
        printf("Erreur de saisie.\n");
        exit (EXIT_FAILURE);
    }

    int valeursModifiees = operation(v1, v2, op, &result, &aux);

    if (erreur == -1) {
        printf("Erreur : Division par zéro.\n");
    } else if (erreur == -2) {
        printf("Erreur : Opérateur non reconnu.\n");
    } else {
        if (valeursModifiees == 1) {
            printf("Résultat : %d\n", result);
        } else if (valeursModifiees == 2) {
            printf("Quotient : %d, Reste : %d\n", result, aux);
        }
    }

    return EXIT_SUCCESS;
}
```

4/

gcc -o Exercice5.4TD Exercice5.4TD.c

echo "5/2" > entree.txt

./Exercice5.4TD <entree.txt> Sortie.txt.

Quotient = 2 , reste = 1

Exercice 5.4 :

1)

```
int maximum (int a, int b) {
if (a > b) {
        return a; }
else { return b;
        }
}
```

2)

```
(include)
int maximum (int a, int b);

int main (void) {
int val1, val2, val3;
if (Scanf ("%d %d %d", &val1, &val2, &val3) != 3)
{ printf ("erreur de saisie"); exit (Exit_Failure); } else {
int max_1 = maximum (val1, val2);
int max glob = maximum (max_1, val3);
return Exit_Success; }
}
```

Exercice 7.7 :

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

int main (void){

   char entree;
if (scanf(" %c", &entree)!=1){
        printf("Erreur");
        exit (EXIT_FAILURE);
}else {
    getchar();

     char a = 'a', z ='z', A = 'A', Z = 'Z';

     bool minuscule = (entree >= a && entree <= z);
     bool majuscule = (entree >= A && entree <= Z );
     bool chiffre = (isdigit(entree));

if (minuscule){
      printf("ceci est un caractere minuscule");
}
else if (majuscule){
      printf("ceci est un caractere majuscule");
}
else if (chiffre){
      printf("ceci est un chiffre");
}
else {
      printf("Autre caractere");
}
return EXIT_SUCCESS;}
}
```

Exercice 7.6

```c
#include <stdio.h>

int main() {
    char c1, c2, c3;
    int distinct = 3;

    printf("Entrez le premier caractère: ");
    scanf(" %c", &c1);

    printf("Entrez le deuxième caractère: ");
    scanf(" %c", &c2);

    printf("Entrez le troisième caractère: ");
    scanf(" %c", &c3);

    if (c1 == c2 && c2 == c3) {
        distinct = 1;
    } else if (c1 == c2 || c1 == c3 || c2 == c3) {
        distinct = 2;
    }

    printf("Il y a %d caractère(s) distinct(s) dans la séquence.\n", distinct);
    return 0;
}
```

Exercice 7.7 :

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

int main(void) {
    int heure, minute, seconde;

    if(scanf("%d:%d:%d", &heure, &minute, &seconde) != 3) {
        printf("erreur de saisie");
        exit(EXIT_FAILURE);
    }

    ++seconde;
    if (seconde == 60) {
        seconde = 0;
        ++minute;
    }
    if (minute == 60) {
        minute = 0;
        ++heure;
    }
    if (heure == 24) {
        heure = 0;
    }

    printf("%02d:%02d:%02d", heure, minute, seconde);

    return EXIT_SUCCESS;
}
```

Exercice 7.8 :

$*(\&b) = b$

$*y = *v$

$*(\&b) = *(\&c)$

$b = c$

```
int *v = 3;
```

```
(int *)(*x)
```

x est un pointeur vers un pointeur vers un entier.

(Suite C H.)

boucles for et while:

compteur <

```
for (int k = 1; k <= 10; ++k) {
    /* instruction */ }
int k = 1;
while (k <= 10) { /* instruction */
                  ++k;
                }
```

scanf("%d %f"); valeur retour scanf (2).

scanf("%d"); valeur de retour (1).

(valeur de retour de scanf
Correspond au nombre de variable
correctement lues et associées)

```
int main (void) {
int minutes, Seconds

    EXIT_SUCCESS;
    }
```

```
if (a>B && a>c && b>c)
```

TP 4

1)

2	0	1
x	y	3

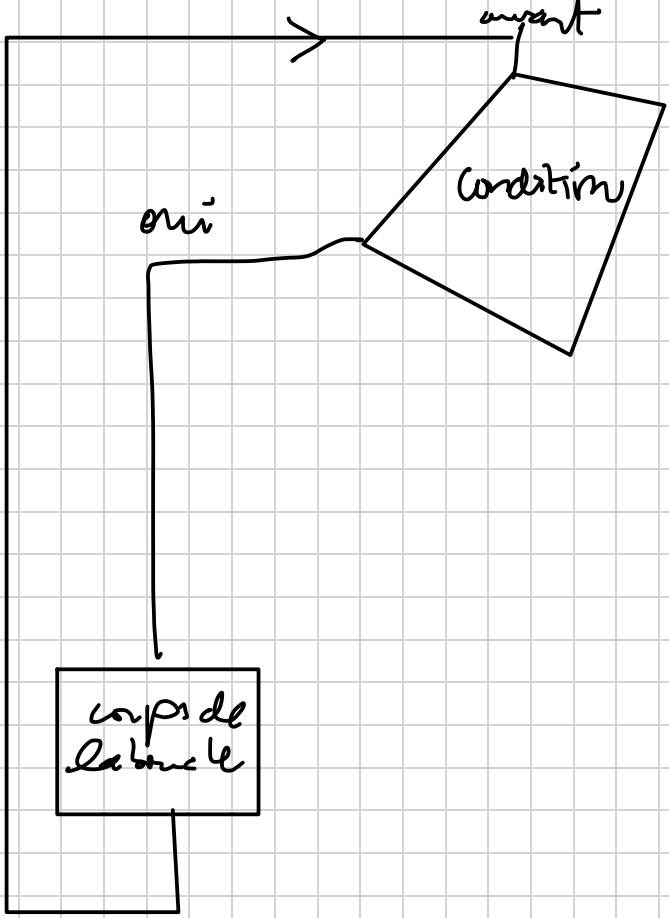
2)

0	-4	0
x	y	3

0
1
10
11
100 4
101 5
110 6
111 7
1000 8
1001 9

1010 A
1011 B
1100 C
1101 D
1110 E
1111 F

0, 1, 3, 15 et 30



while (condition) {

corps de la boucle

}

for (; condition d'arrêt ; exécuté à chaque fin)

{ corps. }

```
for (i = 2; i <= 10; ++i) {
    P = P * q * i;
}
```

```
i = 2
while (i <= 10) {
    P = P * q * i;
    ++i;
}
```

$$\sum_{i=1}^{n} i$$

$$S_k = S_{k-1} + k$$

$$1 \le k \le n$$

```
S = 0;
for (int k = 1; k <= n; ++k)
{
    S = S + k;
}
```

$$\sum_{i=0}^{n} q^i$$

$$S_k = \sum_{i=0}^{k} q^i \qquad S_{k-1} = \sum_{i=0}^{k-1} q^i$$

$$S_k = S_{k-1} + q^k \quad 1 \leq k \leq n$$

```
S = 1;
for (int k = 1; k <= n; ++k) {
    S += puissance(q, k);
```

Exercice 1 :

1) nom | type

2) if (condition) {
    /* instructions */ }
   else { /* instructions */ }

3)

```
for (int k=1; k<=n; ++k) {
    printf("%d", k);
}
```

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

int main(void) {
    char c;
    if (scanf("%c", &c) != 0) {
printf("errore");
exit(EXIT_FAILURE);}

    if (c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z')
    {printf("lettera");}

    else if (c >= '0' && c <= '9') {
    printf("cifra"); }

    else {printf("altro");}

    return EXIT_SUCCESS; }
```

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

int main(void) {

float n1, n2, n3;

    scanf
```

in	x	y	z
1	2	3	4
2	3	4	1

```
if ( a>=c && a<=b || a>=b && a<=c)
else if ( b>=a && b<=c || b>=c && b<=a)
else if ( c>=a && c<=b || c>=b && c<=a)
```

$b \leq c \leq a$

$a \leq b \leq c$

$c \leq a \leq b$

$b \leq a \leq c$

$a \leq c \leq b$

$c \leq b \leq a$

$(a = b \;||\; b = c \;||\; a = c)$

Fonctions.

Exercice 4.1 :

a	carré	Affichage	ligne
2	—	—	7
2	2*2=4	2^2 = 4	8 -> 15
3	—	—	9
3	3*3=9	3^2 = 9	10 -> 15

Exercice 4.3 :

1) float fct moyenne (float a, float b) {
   float moyenne = (a + b) / 2;
   return moyenne;
}

2) #include <stdlib.h>
#include <stdio.h>.

float moyenne (float a, float b);

int main (void) {
float reel1, reel2;
Printf ("Veuillez saisir deux valeurs reelles: \n");
if (scanf("%f %f", &reel1, &reel2) != 2) {
printf ("Erreur de saisie \n");
exit(EXIT_FAILURE); }
float moy = moyenne (reel1, reel2);
Printf("la moyenne de %f et %f et: %f", reel1, reel2, moy);

retun EXI_SUCCESS;

}.

Exercice 4.4 :

a)

```
bool estMajuscule(char c) {
    return (c >= 'A' && c <= 'Z');
}
```

Exercice type CC BPI.

Exercice 1 :

1) Les caractéristiques d'une variable : - Son identifi-
- cateur
- Son type
- Son adresse mémoire
- La valeur (modifiable à souhait).

2) Ordre décroissant prio :

/
+
%
||

3) La valeur retournée par scanf correspond au

```
int puissance (int n, int m) {

    for (int k = 1; k <= m; ++k) {
        P = P * n;
    }

    return P;
}

int longueur (int n) {
    int compteur = 0;
    if (n == 0) { return 1; }
    while (n != 0) {
        n = n / 10;
        Compteur = compteur + 1;
    }
    return Compteur;
}
```

```
reste = 0;
for(int k = 1; k <= n; ++k) {
    reste = n % 10;
    n = n / 10;
    if (reste / 2 == 0) {
        printf("%d", reste); }
    else {
        reste = 0;
    }
reste = 0;
for (int k = 1; k <= n; ++k) {
        reste = n % 10;
        n = n / 10;
```

écrire un fct qui calcule la somme des n premiers nombres ayant la propriété ID IT.

un programme qui fusionne 2 nombres

à faire { n = 2456
m = 1236

2142536 TD1Γ(n)

la somme des chiffres pairs : Σ des chiffres impairs] n premiers nombres

Ecrire une fonction qui test si un nombre est premier (divisible par lui même et 1).

```
bool etpremier (int n) {
if (n == 1) { return false; }.
for (k=2; k < n/2 ; ++k) {
if (n % k == 0) {
return false ; }
else { return true ;
}.
}
int s = 0;
for (k=n; k< m ; ++k) {
if (etpremier (k)) {
somme += k ;
}
```

```
int SommeDiviseurs(int n) {
int somme = 0;
for(int k = 1; k <= n; ++k) {
    if( n % k == 0) {
        somme += k; }
    else {
        somme = somme + 0;
    }
}
```

```
bool amis( int n, int m)

void afficher(int m){
    int cpt = 0;
    int k = 0;
while ( cpt <= m) {
if ( estparfait(k)) {
        printf("%d", k );
    }
        ++ k ;
}
```

Exercice 6.4 :

```
bool estPremier(int n) {
    if (n == 1) {
        return false;
    }
    for(k = 2 ; k <= n/2 ; ++k ) {
        if (n % k == 0) {
            return false;
        } else {
            return true;
        }
    }
```

Exercice 6.5 :

```
int sommeDiviseurs(int n) {
    int S = 0;
    for (int k = 1; k <= n; ++k) {
        if (n % k == 0) {
            S = S + k; }
    } return S;
}
```

Exercice 6.6 :

```
bool estami(int n, int m) {
return (somme diviseurs stricts(m) == m &&
        somme diviseurs stricts(n) == n);
}
```

Tableaux:

Exercice 7.1:

il multiplie par 2 le contenu de chaque case dans le tableau

Exercice 7.2:

1)
```
void affiche_tableau(int t[], size_t nb){
    for(size_t i=0; i<nb; ++i){
        printf("%d", t[i]);
}
int main(void){
    int tab[5] = {1, 2, 3, 4, 5};
affiche_tableau(tab, 5);
```

2)
```
void saisir_tableau(int t[], size_t nb){
    for(size_t i=0; i<nb; ++i){
        if(scanf("%d", &t[i]) != 1){
            printf("erreur");
            exit(EXIT_FAILURE);
}}}
```

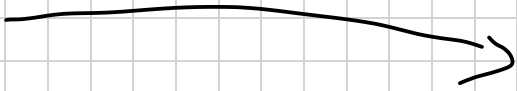
Exercice 7.3:

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

int main(void) {
 int taille;
 printf("Entrez le nombre de note a saisir:\n");
 if(scanf("%d", &taille) != 1){
            exit(EXIT_FAILURE);
 }
 printf("Veuillz saisir les note:\n");
 int tab[taille];
  for( size_t k=0; k < taille ; ++k) {
       if (scanf("%d", &tab[k]) != 1) {
                 exit(EXIT_FAILURE);
}  int nombre = 0;
   float m = moyenne (tab, taille);
printf("la moyenne est : %.1f", &m);
for(int k=0; k < nb; ++k) {
if ( tab[k] >= m){  ++ nombre; }
```

```
printf(" le nombre d'étudiant dans la note et
>= a moyenne : %d", nombre);
    return EXIT_SUCCESS;
}

double moyenne (int [ ], int nb ) {
double s=0.0;
  for (int k=0; k < nb ; ++k) {
      s += tab[k];
}
   return  s/ nb;
}
```

Exercice 7.4 :

on s'inspire du premier donc on calcule la moyenne puis on compare



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

int main(void) {
  int taille;
  printf("Entrez le nombre de note a saisir:\n");
  if(scanf("%d", &taille) != 1){
        exit(EXIT_FAILURE);
  }
  printf("Veuillz saisir vos note:\n");
  int tab[taille];
  for (size_t k=0; k<taille; ++k){
      if (scanf("%d", &tab[k]) != 1) {
          exit(EXIT_FAILURE);
  } int nb2 = 0;
    int nombre1 = 0;
    float m = moyenne(tab, taille);
  printf("la moyenne et : %.1f", &m);
  for (int k=0; k<nb; ++k){
  if (tab[k] > m){ ++nombre1; }
  else if (tab[k] < m) { ++nombre2;
  }
    if (nombre1 == nombre2) {
```

```
    Printf("equilibré");
else {
        printf("non equilibré");
    }.
```

Exercice 7.5:

```
int fct (int t[], int val, int taille){
    for (size_t k=0; k<taille; ++k) {
        if (tab[k] == val) {
            return 1; }
        else { return 0; }
    }
}
```

Exercice 7.6:

```
int premier_occ ( int t[], size_t n {
    int maxim = t[0];
    int indice = 0;

    for (size_t k=0, k<n; ++k){
        if ( t[k] > maximum) {
            maxim = t[k];
            indice = k; }
```

```
    }
  }
  return indice;
}
```

Exercice 7.7 :

```
int minimum ( int t[ ], size_t n ) {
    int minimum = t[0];
    int indice = 0;
    for (size_t k=0 , k<n ; ++k ){
        if ( t[k] < minimum ) {
            minimum = t[k];
            indice = k; }
    }
  }
  return indice;
}
void supprMin ( int t[ ], size_t n) {
    size_t indice_min = indice_plus_petite_val (t, n);
    for (size_t k=indice_min ; i<n-1 ; ++k) {
            t[i] = t[i+1];
    }
```

```
--(*n);
}
```

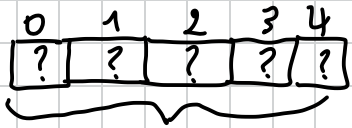
```
int sum(int t[], int n){
int sum = 0;
for(int k=0; k<n; ++k){
    sum += t[k];
}
return sum;
}
```

```
int count=0;
for(size_t k=0; k<n; ++k){
if(t[k]%2==0){
    ++count;
}
return count;
}
```

Tableaux Rappels:

1) Déclaration d'un tableau:

```
int tab[5];
```



2) Déclaration et initialisation:

```
int tab[5] = {100};
```

toutes les cases initialisés avec la valeur 100.


```
int tab[5] = {4, 2, 6, 8, 3}
```

3) Parcourir un tableau:

```
for (size_t k=0; k < taille; ++k) {

}
```

4) Tableau Comme parametre de fonction:

```
fct (int t[ ], size_t taille) {
```

5) Appel d'une fonction prenant en param un tableau:

```
fct (t, n);
```

Quelques fct de manipulation des Tableaux:

```
void lire_Tableau(int t[], size_t taille){
    for (size_t k=0; k < taille ; ++k) {
  if (scanf("%d", &t[k]) != 1){
        exit(EXIT_FAILURE);
    }
}
}
void Affiche_Table(int t[], size_t taille) {
    for (size_t k=0; k<taille;++k) {
        printf("%d ", t[k]);
}
    printf("\n");
}
```

```
void fctdecrler (int t[ ], size_t *n, size_t
                                    k) {
for (size_t i=k; i<*n-1; ++i){
        t[i]=t[i+1];
  }
--(*n);
}
```

Exercice 8.1

L'exécution du programme suivant provoque l'affichage qui suit. Pouvez-vous expliquer pourquoi ?

```
int main(void) {
  char msg[] = "Bonjour";
  strcat(msg, "␣tout␣le␣monde");

  size_t n = strlen(msg);
  printf("Message␣de␣longueur␣%zu␣:␣%s\n", n, msg);

  return EXIT_SUCCESS;
}
```

Affichage produit :

```
Message de longueur 21 : Bonjour □
```

Exercice 8.1 :

il faut mettre 30 caractères par exemple car l'espace ne suffit pas.

```
Char msg [30] = "Bonjour"
Strcat (msg, "␣ tout ␣ le ␣ monde");
Size_t n = strlen (msg);
```

Exercice 8.2

Un seul de ces programmes fonctionne. Pouvez-vous dire lequel et expliquer pourquoi ?

```
int main(void) {
  char msg[] = "bonjour␣tout␣le␣monde";
  msg[0] = 'B';
  printf("%s\n", msg);
  return EXIT_SUCCESS;
}
```

```
int main(void) {
  char *msg = "bonjour␣tout␣le␣monde";
  msg[0] = 'B';
  printf("%s\n", msg);
  return EXIT_SUCCESS;
}
```

Exercice 2 :

C'est le premier programme msg est declaré comme un tableau de caractère et on modifie le premier caractère donc `b` pour `B`.

le deuxième ne fonctionnera pas car c'est un pointeur vers une chaine de caractères constante, donc juste pour la lecture.

Exercice 8.3

Sachant que la fonction **sizeof** renvoie la taille en octets de son opérande et que la taille d'un **char** est de 1 octet, qu'affichent les deux programmes suivants ? Expliquez.

```
int main(void) {
  const char *msg = "bonjour a tous";
  printf("La variable est de taille %zu\n", sizeof(msg));
  printf("La chaine est de taille %zu\n", strlen(msg));
  return EXIT_SUCCESS;
}
```

```
int main(void) {
  const char msg[] = "bonjour a tous";
  printf("La variable est de taille %zu\n", sizeof(msg));
  printf("La chaine est de taille %zu\n", strlen(msg));
  return EXIT_SUCCESS;
}
```

Exercice 8.3 :

Prog 1 { 8 / 14

Prog 2 { 15 / 14

Exercice 8.5 :

```
int main (void) {
int longueur = 0;
const char *Chaine = "Bonjour";

while (chaine[longueur] != '\0') {
        ++ncaracteres
}
printf("%d", longueur);
```

Exercice 8.6 :

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

int main(void) {

    char chaine[100];

    printf("Entrez une chaine de caracteres : ");
    if(scanf("%s", &chaine)!= 1){
printf("erreur de saisie\n");
exit(EXIT_FAILURE);
}
    size_t longueur = strlen(chaine);

    for (size_t i = 0; i < longueur; ++i) {
        if (chaine[i] >= 'A' && chaine[i] <= 'Z') {
            chaine[i] = chaine[i] + ('a' - 'A');
        }
    }

    printf("Chaine transformee : %s\n", chaine);

    return EXIT_SUCCESS;
}
```

Exercice 8.7 :

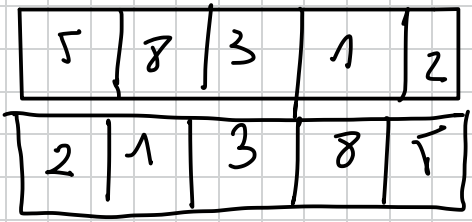
1)
```c
int checkchar (char chain[], char c){
for (int k = 0; chn[k] != '\0'; ++k){
        if ( c == chain[k]) {
            return 0;
        }
    }
    return 1;
}
```

2)
```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

int main (void) {
char chaine[100];
int nb_mots;
if (scanf("%d", &nb_mots) != 1) { exit; }

char x;
if (scanf("%d", &x) != 1) {
        printf("Erreur de saisie\n");
        exit(EXIT_FAILURE);
    }

Cherchecha (&chaine; x);
```

Exercice :

Ecrire une fonction permettant d'inverser les éléments d'un tableau de n entiers.

Exemple :

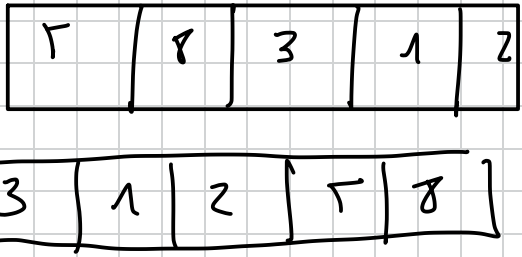



```
int inverser (int tab[], int taille){
  for (int k=0; k< taille; ++k){
    tab[k] = tab[k-1];
```

Exercice 7 :

Ecrire une fonction rotation qui permet de décaler les éléments d'un tableau de n entiers, de k position.

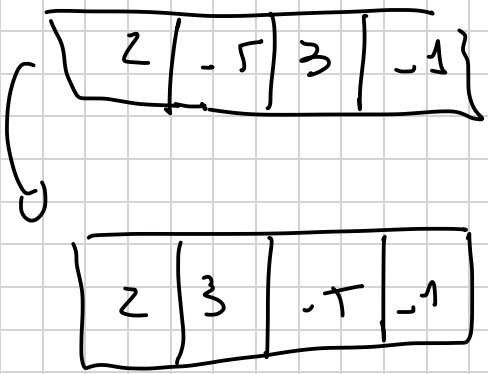
Exemple :



Exercice 3 :

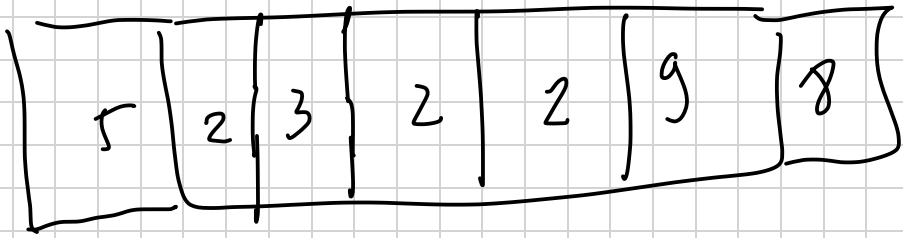
Ecrire une fonction qui permet de ranger les éléments positifs d'un tableau au début des cases et les autres négatifs à la fin.

Exemple :



Exercice 7.8 :

```
void efface ( int tab[], int *nb, int val) {
    for (int k=0; k<*nb; ++k) {
        if (tab[k] == val) {
            decale(tab, k);
            *nb = *nb - 1;
        } k = k - 1;
    }
```




```
int search(int tab[], int val, int taille) {
    int index = 0;
    for (int k = 0; k < taille; ++k) {
        if (tab[k] == val) {
            return k;
        }
    }
    return -1;
}
```

```
size_t decaler ( int tab[ ], size_t taille, int val ){
    for (size_t k = 0; k < n - 1 ; ++k) {
      if ( tab[k] == val ) {
          tab[k] = tab [k + 1] ;
      }
    }
```

k + [:] → i pour taille

∴ taille

```
void decaler (int tab[ ], int *n, int val) {
  for(int k = 0; k < *n - 1; ++k) {
      tab[k] = tab[k + 1];
    }
  --(*n);
}
```

un nb premier

```
bool estpremier(int n) {
if (n==0 || n==1) { return false; }
for (int k=2; k<n; k++) {
        if (n%k==0) { return false; }
}
return true;
}
```

```
int somme_diviseurs(int n) {
  int s=0;
  for(int k=0; k<=n; k++) {
   if (n%k==0) {
        s+=k;
   }
  }
```

```
void lire_tableau(int t[], size_t n){
    for(size_t k=0; k<n; ++k){
    if(scanf("%d", &t[k]) != 1){
            printf("Erreur\n");
            exit(EXIT_FAILURE);
    }
    }
}

void affiche_tableau(int t[], size_t n){
    for(size_t k=0; k<n; ++k){
        printf("%d", t[k]);
    }
        printf("\n");
}
```

```
void decaler(int t[], size_t *n,
             size_t k) {
  for (int i = k; i < n - 1; ++i) {
    t[i] = t[i+1];
  }
  *n = *n - 1;
}

size_t indice_val(int t[], int val, int n) {
  for (size_t k = 0; k < n; ++k) {
    if (t[k] == val) { return k; }
  }
  return -1;
}
```

```
void supp_char(char mot, char c){
    size_t pos;
    pos = indice(mot, c);
    while (pos != -1) {
        decaler(mot, pos);
        pos = indice(mot, c);
    }
}
```

134 134

1 < 3 < 4 134

```
int nb=0;
while (chiffre != 0) {
    chiffre /= 10;
    ++nb;
}
```

1 20

```
unite = n % 10;
n = n / 10;
```

```
while ((nb-1)!=0) {

unité = n % 10;
n = n / 10;  if (unité > n) {
        --nb; }
}
```

1)

```
bool Estanagram(char ch, char ch2
                , size_t taille1, size_t taille2)
{
int cpt=0;
if ( taille1 != taille2 ) { return false ; }

for (size_t k = 0; k < taille1 ; ++k) {
for (size_t i = 0, i < taille2 i ++i) {
 if ( ch[k] = ch[i] ) {
        ++cpt; }
 if ( cpt != 1) { return false ; }
}
}
```

```
    return true;
}

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

#define LEN_MAX 100
#define LEN_MAX_STR "100"

bool anagramma(const char *s1, size_t
                                len1,
               const char *s2, size_t len2);

int main(void) {
  char s1[LEN_MAX + 1];
  if (scanf("%" LEN_MAX_STR "s", s1) != 1) {
      puts("Errore");
      exit(EXIT_FAILURE);
  }
  char s2[LEN_MAX + 1];
```

```
if(scanf("%" LEN_MAX_STR "s", s2)!=1){
        printf("Error\n");
        exit(EXIT_FAILURE);
    }

if (are_anagram(s1, strlen(s1), strlen(s2),
                                    s2)){
    printf("are anagram\n");
} else {
        printf("Are not\n");
    }
    return EXIT_SUCCESS;
}
```