informatique S1

passage par valeur/référence/pointeur

Problématique

Une fonction peut avoir plusieurs paramètres mais ne peut retourner qu'une seule valeur :

```
int fonction1()
{
    int valeur;
    // traitement
    return valeur;
}
```

```
int fonction2(int nb, char c)
{
    int valeur;
    // traitement dépendant
    // de nb et c
    return valeur;
}
```

EX : (openclassrooms)

```c
void decoupeMinutes(int heures, int minutes)
{
    heures = minutes / 60;  // 90 / 60 = 1
    minutes = minutes % 60; // 90 % 60 = 30
}

int main(int argc, char *argv[])
{
    int heures = 0, minutes = 90;

    /* On a une variable minutes qui vaut 90.
   Après appel de la fonction, je veux que ma variable
   "heures" vaille 1 et que ma variable "minutes" vaille 30 */

    decoupeMinutes(heures, minutes);

    printf("%d heures et %d minutes", heures, minutes);

    return 0;
}
```

Notion d’adresse

// on suppose que a sera affectée à la case mémoire 2

int a=51;

cout << &a <<” : “<< a ;

affichera

2 : 51

(ou 0x2 : 51)

Étiquette (programme)	**Adresse**	**Valeur**
	0	?
	1	?
a	2 : &a	51
	..	?
	0xFE(254)	?
	..	?
	max (suivant architecture)	?

```c
void decoupeMinutes(int heures, int minutes)
{
    heures = minutes / 60;  // 90 / 60 = 1
    minutes = minutes % 60; // 90 % 60 = 30
}



int main(int argc, char *argv[])
{
    int heures = 0, minutes = 90;

    /* On a une variable minutes qui vaut 90.
   Après appel de la fonction,
   je veux que ma variable
   "heures" vaille 1 et que
   ma variable "minutes" vaille 30 */

    decoupeMinutes(heures, minutes);

    printf("%d heures et %d minutes"
           , heures, minutes);

    return 0;
}
```

Étiquette (programme)	Adresse	Valeur
minutes (main)	&minutes	90
heures (main)	&heures	0

```c
void decoupeMinutes(int heures, int minutes)
{
    heures = minutes / 60;  // 90 / 60 = 1
    minutes = minutes % 60; // 90 % 60 = 30
}



int main(int argc, char *argv[])
{
    int heures = 0, minutes = 90;

    /* On a une variable minutes qui vaut 90.
   Après appel de la fonction,
   je veux que ma variable
   "heures" vaille 1 et que
   ma variable "minutes" vaille 30 */

    decoupeMinutes(heures, minutes);

    printf("%d heures et %d minutes"
           , heures, minutes);

    return 0;
}
```

Étiquette (programme)	Adresse	Valeur
minutes (decoupeMinutes)	&minutes	90
heures (decoupeMinutes)	&heures	0
minutes (main)	&minutes	90
heures (main)	&heures	0

```c
void decoupeMinutes(int heures, int minutes)
{
    heures = minutes / 60;  // 90 / 60 = 1
    minutes = minutes % 60; // 90 % 60 = 30
}



int main(int argc, char *argv[])
{
    int heures = 0, minutes = 90;

    /* On a une variable minutes qui vaut 90.
   Après appel de la fonction,
   je veux que ma variable
   "heures" vaille 1 et que
   ma variable "minutes" vaille 30 */

    decoupeMinutes(heures, minutes);

    printf("%d heures et %d minutes"
           , heures, minutes);

    return 0;
}
```

Étiquette (programme)	Adresse	Valeur
minutes (decoupeMinutes)	&minutes	30
heures (decoupeMinutes)	&heures	1
minutes (main)	&minutes	90
heures (main)	&heures	0

```c
void decoupeMinutes(int heures, int minutes)
{
    heures = minutes / 60;  // 90 / 60 = 1
    minutes = minutes % 60; // 90 % 60 = 30
}



int main(int argc, char *argv[])
{
    int heures = 0, minutes = 90;

    /* On a une variable minutes qui vaut 90.
   Après appel de la fonction,
   je veux que ma variable
   "heures" vaille 1 et que
   ma variable "minutes" vaille 30 */

    decoupeMinutes(heures, minutes);

    printf("%d heures et %d minutes"
           , heures, minutes);

    return 0;
}
```

Étiquette (programme)	Adresse	Valeur
minutes (main)	&minutes	90
heures (main)	&heures	0

Passage de paramètre par référence

```
void triple(int & value)
{
    value = value * 3;
    // ou value *= 3;
}

int main()
{
    int a = 5;
    triple(a);
    cout<<a<<endl;

    return 0;
}
```

Étiquette (programme)	Adresse	Valeur
a (main)	&a	5

Passage de paramètre par référence

```
void triple(int & value)
{
    value = value * 3;
    // ou value *= 3;
}

int main()
{
    int a = 5;
    triple(a);
    cout<<a<<endl;

    return 0;
}
```

Étiquette (programme)	Adresse	Valeur
a (main) value(triple)	&a &value	~~5~~ 15

EX : (openclassrooms)

```
void decoupeMinutes(int & heures, int & minutes)
{
    heures = minutes / 60;  // 90 / 60 = 1
    minutes = minutes % 60; // 90 % 60 = 30
}

int main(int argc, char *argv[])
{
    int heures = 0, minutes = 90;

    /* On a une variable minutes qui vaut 90.
   Après appel de la fonction, je veux que ma variable
   "heures" vaille 1 et que ma variable "minutes" vaille 30 */

    decoupeMinutes(heures, minutes);

    printf("%d heures et %d minutes", heures, minutes);

    return 0;
}
```

Exercice

- écrire une fonction qui permute 2 nombres entiers
- écrire une fonction qui ordonne 2 nombres “de type double”
- en utilisant cette fonction, ordonner 3 nombres

```
void ordonne(double & n1,double & n2)
{
    if (n2<n1)
    {
        double tmp = n1;
        n1=n2;
        n2=tmp;
    }
}
void ordonne(double & a,
             double & b,double & c )
{
    ordonne(a,b);
    ordonne(b,c);
    ordonne(a,b);
}
int main()
{
    double x1 = 9, x2 = 7,x3 = 5;
    ordonne(x1,x2,x3);
    cout<<x1<<" "<<x2<<" "<<x3<<endl;
    return 0;
}
```

Notion de pointeur

// on suppose que a sera affectée à la case mémoire 2

int a=51;

int * p = nullptr;

p = &a ;

cout << &a <<" : "<< a <<endl;

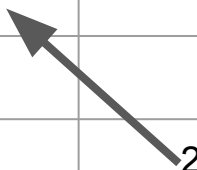
cout << p   <<" : "<< *p <<endl;

affichera 2 fois la même chose :

 2 : 51

ou (0x2 : 51)

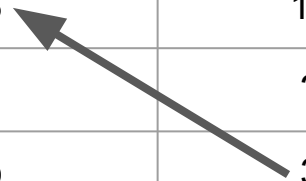
Étiquette (programme)	Adresse	Valeur
	0	?
	1	?
a	2 : &a	51
	..	?
* p	p	2 (&a)
	..	?
	max (suivant architecture)	?



!!! pointeur = danger !!!

```
// on suppose que a sera affectée à la case mémoire 2

int a=51;

int * p = nullptr;

p = &a ;

(*p)++; //ok, on incrémente la valeur pointée

p++ ;  // attention : on change d'@ pointée !!

*p = 10; // on modifie la valeur à l'@ 3 !!


cout << &a <<" : "<< a  <<endl;

cout << p   <<" : "<< *p <<endl;
```

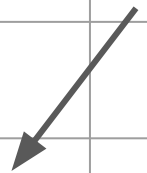
Étiquette (programme)	Adresse	Valeur
	0	?
	1	?
a	2 : &a	52
	3	10
	..	?
*p	p	3
	..	?



Passage de paramètre par pointeur

```
void triple(int * value)
{
    *value = *value * 3;
    // ou *value *= 3;
}

int main()
{
    int a = 5;
    triple(&a);
    cout<<a<<endl;

    return 0;
}
```

Étiquette (programme)	Adresse	Valeur
*value(triple)	value	&a (ex 0xFA8)
a (main)	&a (ex 0xFA8)	~~5~~ 15



EX : (openclassrooms)

```c
void decoupeMinutes(int * heures, int * minutes)
{
    *heures = *minutes / 60;  // 90 / 60 = 1
    *minutes = *minutes % 60; // 90 % 60 = 30
}

int main(int argc, char *argv[])
{
    int heures = 0, minutes = 90;

    /* On a une variable minutes qui vaut 90.
   Après appel de la fonction, je veux que ma variable
   "heures" vaille 1 et que ma variable "minutes" vaille 30 */

    decoupeMinutes(&heures, &minutes);

    printf("%d heures et %d minutes", heures, minutes);

    return 0;
}
```

Ex : qu'affiche ce programme ?

```
int main()
{
  int n1 = 10;           // on supposera que n1 est à l'@ 0x10
  int *n2 = nullptr;
  int n3 = 20;           // on supposera que n3 est à l'@ 0x12
  int *n4 = nullptr;
  cout << &n1 << " " << n2 << " " << &n3 << " " << n4 << endl;
  cout << n1 << " " << *n2 << " " << n3 << " " << *n4 << endl;  // crash du programme
  // ------------------------------------------------------
  n2 = &n1;
  n4 = n2;
  cout << &n1 << " " << n2 << " " << &n3 << " " << n4 << endl;
  cout << n1 << " " << *n2 << " " << n3 << " " << *n4 << endl;
  // ------------------------------------------------------
  n1 = 30;
  *n2 = 40;
  n4 = &n3;
  n3 = 5;
  cout << &n1 << " " << n2 << " " << &n3 << " " << n4 << endl;
  cout << n1 << " " << *n2 << " " << n3 << " " << *n4 << endl;
  return 0;
}
```

Ex

on considère 3 variables pour gérer le temps : heure/minute/seconde

écrire une fonction qui permet d’incrémenter le temps d’une seconde