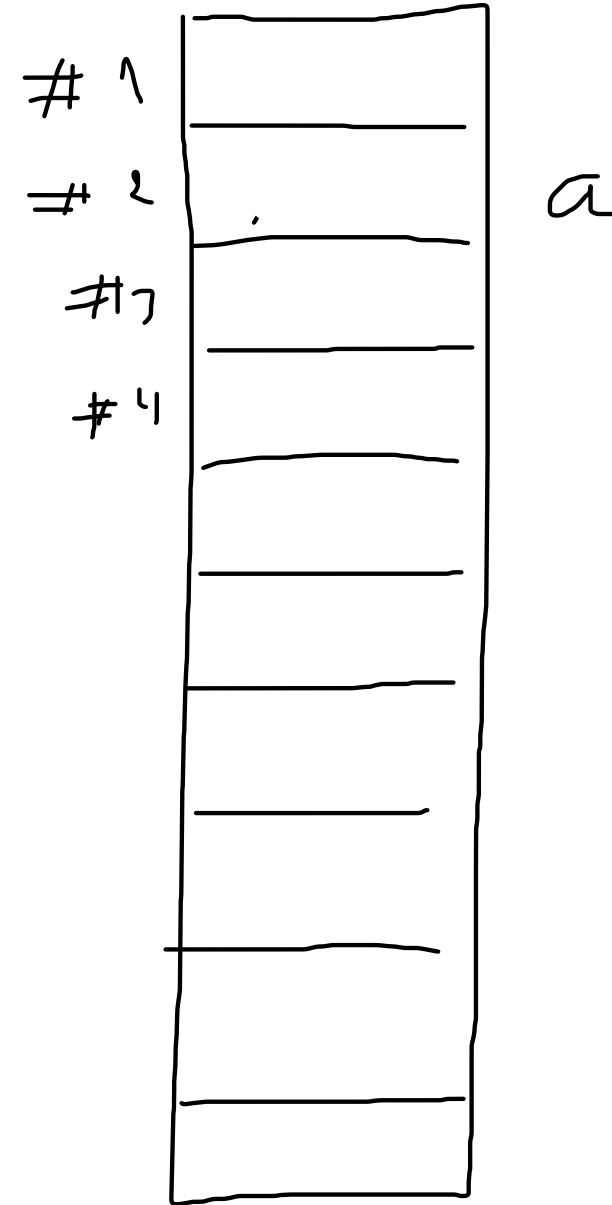


Chapitre 2. Les pointeurs

```
main()
{
  int a;
  a=10;
  printf("a=%d",a)
  printf("l'adresse de a %x",&a);
}
```

10 #2

8 a = #2



```
main()
{
int a;
int *q; //pointeur
a=10;
printf("a=%d",a);
p=&a;
printf("l'adresse de a %x",&a);
printf("la, valeur de p %p",q);
}
```

$$I_a = 0.065 \text{ fe} \wedge c$$

$\uparrow = \text{Ja}$
 $*q = \text{le contour pointé par } q$

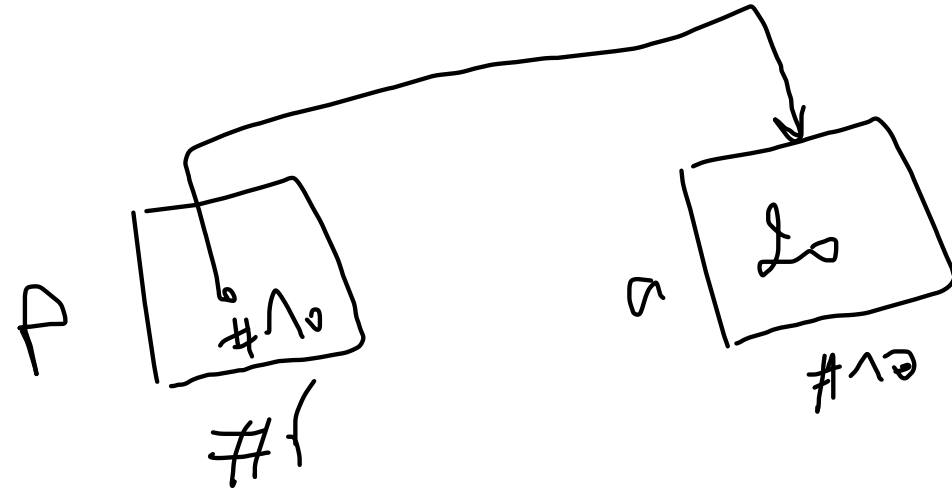
	adresse	contenu	
0062fe1c		00000000	a
0062fe1d		00000000	
1e		00000000	
1f		00000000	
20	.	.	Λ
	:	:	
	:	:	
	:	:	
0062ff00		00	q
		62	
		fe	
		1c	

Définition:

un pointeur est une variable qui contient l'adresse d'une autre variable

int a, *p;

p = &a



*p => a

&p = #1

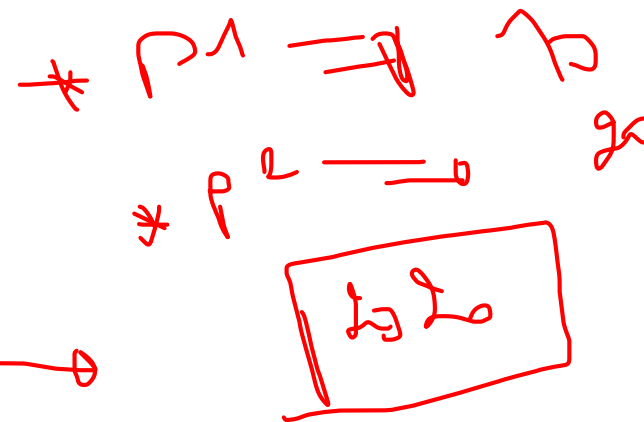
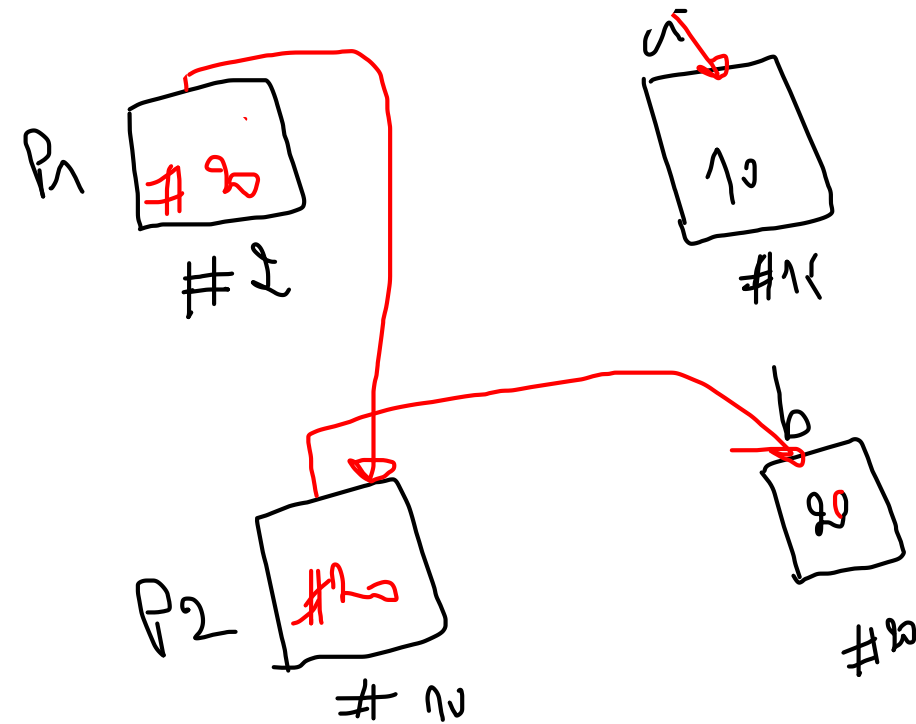
&a = #10

p = #10

```

#include<stdio.h>
main()
{
    float a,b,*p1,*p2;
    a=10;
    b=20;
    p1=&a;
    p2=&b;
    printf("la valeur de p1 est %p\n",p1);
    printf("le contenu pointé par p1 est %f\n",*p1);
    printf("la valeur de p2 est %p\n",p2);
    printf("le contenu pointé par p2 est %f\n",*p2);
    p1=p2;
    print("%f %f",*p1,*p2);
}

```



Les opérations arithmétiques sur les pointeurs

int a, *p;

$$a = 10$$

$$p = \&a$$

$$p + 2 = \#4$$

$$\begin{aligned} *p + 2 &= 10 + 2 \\ &= 12 \end{aligned}$$

$$p + 1 = \#3$$

$$*(p + 1) = 5$$

#1		
#2	10	a
#3	5	
#4	6	
#5	12	
#6	#2	p

La règle:

type * p

P + i =

P + i x sizeof(type)

sizeof(char)=1

sizeof(float)=4

sizeof(int)=4

```
int a,*q;  
a=10;  
q=&a;
```

$q = 0062fe1c$

$q + 1 = 0062fe1c + 1 \times 4$
 $= 0062fe20$

$q + 2 = 0062fe24$

0062fe1c

0062fe1d

1e

1f

20

0062ff00

le par q

adresse	contenu	
	00 00 00 00	a
	00 00 00 00	
	00 00 00 00	
	00 00 00 00	
	.	
	⋮	
	⋮	
	00	q
	62	
	fe	
	1c	

$P \longrightarrow \text{address}$

$*P \longrightarrow \text{Content}$

$P + 2 \longrightarrow \text{address}$

$*P + 2 \longrightarrow \text{Content}$

int $a, *p1, *p2, *p3$

$$p1 = &a$$

$$p2 = p1 + 2$$

$$p3 = \cancel{p1} + p2$$

! impossible

adresse	contenu
00FE1C	00 00 00 00
00FE1D	00 00 00 00
00FE1E	00 00 00 00
00FE1F	00 00 00 00
00FE20	.
...	...
00FE24	00
	6E
	8C
	AC

00FE1C
00FE1D
00FE1E
00FE1F
00FE20

00FE24

par 4

```
#include<stdio.h>
main()
```

```
{
    char cc,*p;
    cc='A';
    p=&cc;
    printf("l'adresse de cc est %x\n",&cc);
    printf("la valeur de p est %p\n",p);
    printf("la valeur de p+1 est %p\n",p+1);
    printf("la valeur de p+2 est %p\n",p+2);
    printf("la valeur de p+3 est %p\n",p+3);
    printf("la valeur est %c\n",*p);
    printf("la valeur est %c\n",*p+1);
    printf("la valeur est %c\n",*p+2);
    printf("la valeur est %c\n",*(p+1));
}
```

Exemple

00FE1D2B
00FE1D2C
00FE1D2D
00FE1D2E
00FE1D2F

A
B
C

z

00FE1D2A	.
00FE1D2B	'A'
00FE1D2C	?z
00FE1D2D	
00FE1D2E	
00FE1D2F	
00FE1D30	00
31	FE
32	AD
33	2B

CC
p+1
p+2
p+3

↑ p

Les pointeurs et les tableaux

int T[5] = {2, 4, 6, 8, 10}

int *p;

p = T

*p = 2

p + 1 = &T[1]

*(p + 1) = T[1]
= 4

*p + 1 = 2 + 1 = 3

p + 2 = &T[2]

⚠ T = &T[0]

p + i = p + i * sizeof(type)

⇕

T + i = T + i * sizeof(type)

*p = *T = T[0]

```
int T[5]={2,3,4,7,9},*p;
p=&T[0]
```

$$p + 1 = \&T[0] + 1 \times 4$$

$$= \&T[0] + 4 = \&T[1]$$

$$p + 2 = \&T[0] + 2 = \&T[2]$$

$$*(p + 1) = 3 = T[1]$$

4 bytes

#1	2	T[0]
#2	3	T[1]
#3	4	T[2]
#4	7	T[3]
#5	9	T[4]
#6	#1	p
#7		
#8		

Si T est un tableau

et on fait $P = T$

$$P + i = T + i = \&T[i]$$

$$*(P + i) = \cancel{*}(T + i) = T[i]$$

Remarque:

un tableau est un pointeur

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

```
main()
```

```
{
```

```
    int T[5],i;
```

```
    //remplir le tableau
```

```
    for(i=0;i<5;i++)
```

```
    {
```

```
        printf("T[%d]=",i);
```

```
        scanf("%d",&T[i]);
```

```
    }
```

```
    //Affichage
```

```
    for(i=0;i<5;i++)
```

```
        printf("%d\n",T[i]);
```

```
}
```

8 T[i]

T[i]