Pointeurs et tableaux

esprit
Se former autrement

Pointeurs et tableaux

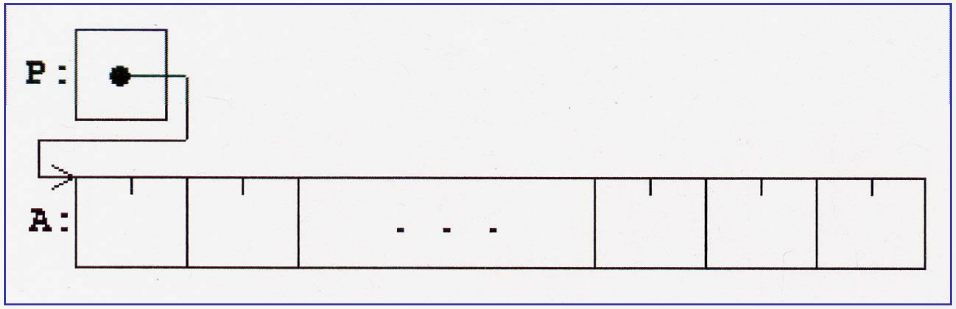
- Chaque opération avec des **indices** de tableaux peut être aussi exprimée à l'aide de pointeurs.
- Comme nous l'avons déjà vu dans le cours, le nom d'un tableau représente l'**adresse** du **premier élément** du tableau.

&tableau[0] et tableau représentent l'adresse du premier élément du tableau

.

- Le nom d'un tableau est un pointeur constant sur le premier élément du tableau.

```
int A[10];
int * P;
P = A;      /*est équivalente à P = &A[0]*/
```



Pointeurs et tableaux

Comme A représente l'adresse de A[0],

`*(A+1)` désigne le contenu de A[1]

`*(A+2)` désigne le contenu de A[2]

...

`*(A+i)` désigne le contenu de A[i]

- Un *pointeur* est une variable,
donc des opérations comme `P = A` ou `P++` sont permises.

- Le *nom d'un tableau* est une constante,
donc des opérations comme `A = P` ou `A++` sont impossibles.

Pointeurs et tableaux

Arithmétique des pointeurs

- On peut **déplacer** un pointeur dans un plan mémoire à l'aide des opérateurs : d'addition, de soustraction, d'incrémentation, de décrémentation.

Affectation par un pointeur sur le même type

- P1 et P2 deux pointeurs sur le même type de données

  P1 = P2; affecte l'adresse contenue dans P2 à P1.

  → P1 pointe sur la même variable que P2.

Addition et soustraction d'un nombre entier

- Si P pointe sur l'élément A[i] d'un tableau, alors

  P+n pointe sur A[i+n]

  P-n pointe sur A[i-n]

- le déplacement d'un pointeur par l'opérateur + ou – se fait par un nombre d'octets multiple de la taille de la variable sur laquelle il pointe.

Pointeurs et tableaux

Arithmétique des pointeurs

Incrémentation et décrémentation d'un pointeur

Si P pointe sur l'élément A[i] d'un tableau, alors après l'instruction

P++;	P pointe sur A[i+1]
P+=n;	P pointe sur A[i+n]
P--;	P pointe sur A[i-1]
P-=n;	P pointe sur A[i-n]

Pointeurs et tableaux

Arithmétique des pointeurs

Soustraction de deux pointeurs

- Soient P1 et P2 deux pointeurs qui pointent *sur deux cases du même tableau*:

  P1-P2 fournit le nombre de cases comprises entre P1 et P2.

  Le résultat de la soustraction **P1-P2** est

  - négatif, si P1 précède P2
  - nul, si P1 = P2
  - positif, si P2 précède P1

Comparaison de deux pointeurs

On peut comparer deux pointeurs par <, >, <=, >=, ==, **!**=.

La comparaison de deux pointeurs qui pointent *sur deux cases du même tableau* est équivalente à la comparaison des indices correspondants.

Récapitulatif

Soit un tableau A de type quelconque et i un indice d'une composante de A:

`A`	désigne l'adresse de	`A[0]`
`A+i`	désigne l'adresse de	`A[i]`
`*(A+i)`	désigne le contenu de	`A[i]`

Si P = A, alors

`P`	pointe sur l'élément	`A[0]`
`P+i`	pointe sur l'élément	`A[i]`
`*(P+i)`	désigne le contenu de	`A[i]`

Pointeurs et chaines de caractères

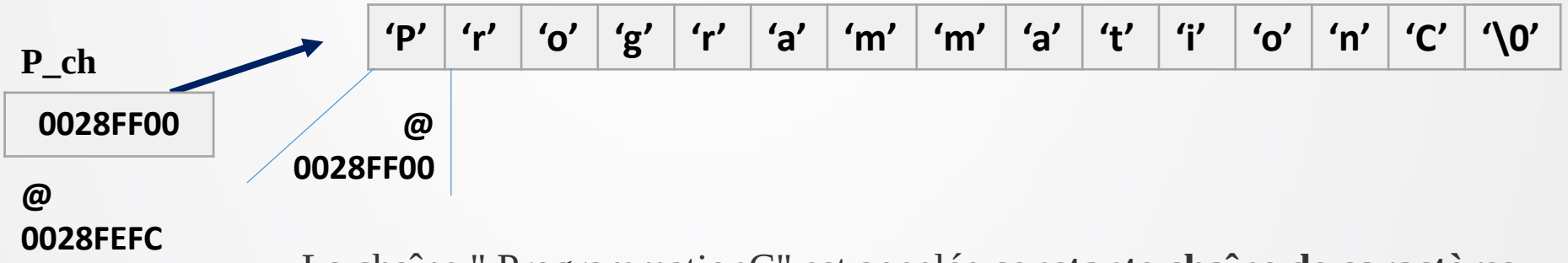
– Tout ce qui a été mentionné concernant les pointeurs et les tableaux reste vrai pour les pointeurs et les chaînes de caractères.

– Une chaine de caractère est un tableau de caractères qui se termine par le caractère spécial ‘ \0 ’.

– On peut attribuer l'adresse d'une chaîne de caractères constante à un pointeur sur char :

Exemple:

char* p_ch = " ProgrammationC "; //p_ch est un pointeur sur une variable de type char.



La chaîne " ProgrammationC" est appelée **constante chaîne de caractères.**

Pointeurs et chaines de caractères

- Le **compilateur** place la chaîne « ProgrammationC" dans une zone mémoire de 15 octets, c'est-à-dire un tableau de 15 caractères qui contient les 14 lettres du mot ProgrammationC et le caractère ‘ \0 ‘.

- **A l'exécution**, l'affectation p_ch = " ProgrammationC " met l'adresse de cette zone mémoire dans le pointeur p_ch → p_ch pointe donc sur le premier caractère de la chaîne.