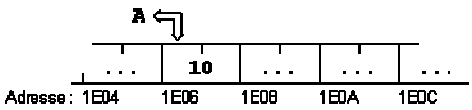
Les pointeurs et la gestion dynamique de la mémoire

1

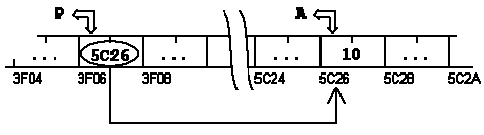
1. Adressage de variables

- **Adressage direct: Accès au contenu d'une variable par le nom de la variable.**

```
short A;
A = 10;
```



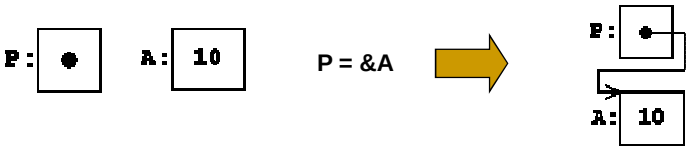
- **Adressage indirect: Accès au contenu d'une variable, en passant par un pointeur qui contient l'adresse de la variable.**



2

2. Opérations de base sur les pointeurs

&<NomVariable> fournit l'adresse de la variable <NomVariable>



***<NomPointeur>** désigne le contenu de l'adresse référencée par le pointeur **<NomPointeur>**


```
P = &A;
B = *P;
*P = 20;
```

3

3. Déclaration d'un pointeur

<Type> *<NomPointeur> déclare un pointeur <NomPointeur> qui peut recevoir des adresses de variables du type <Type>

Exemple

```
main()
{
      short A = 10;
      short B = 50;
      short *P;
      P = &A;
      B = *P;
      *P = 20;
      return 0;
}
```

```
main()
{
      short A, B,*P;
      A = 10;
      B = 50;
      P = &A
      B = *P;
      *P = 20;
      return 0;
}
```

4

4. Pointeurs et tableaux

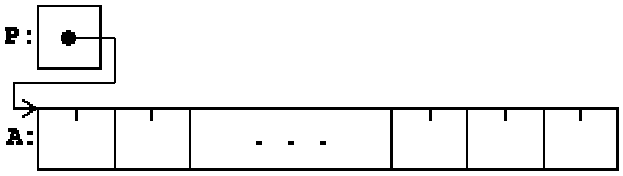
- Le nom d'un tableau représente l'adresse de son premier élément. en d'autre termes:

  &t [0] ⇔ t

Exemple

int A[10]; int *P;

P = A ⇔ P = &A[0];



5

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]

6

Formalisme tableau et formalisme pointeur

Formalisme tableau

```
main()
{
 int T[10] = {-3, 4, 0, -7, 3, 8, 0, -1, 4, -9};
 int POS[10];
 int I,J;
 for (J=0,I=0 ; I<10 ; I++)
   if (T[I]>0)
    {
       POS[J] = T[I];
       J++;
     }
}
```

Formalisme pointeur

```
main()
{
int T[10] = {-3, 4, 0, -7, 3, 8, 0, -1, 4, -9};
int POS[10];
int I,J;
for (J=0,I=0 ; I<10 ; I++)
    if (*(T+I)>0)
      {
         *(POS+J) = *(T+I);
         J++;
      }
}
```

7

5.Arithmétique des pointeurs

- Affectation de pointeurs

  P1 = P2; P1 et P2 doivent être de même type

  ➔ P1 pointe sur le même objet que P2

- Addition et soustraction d'un nombre entier

  P = &A[i] ➔ **P+n** pointe sur **A[i+n]**

  ➔ **P-n** pointe sur **A[i-n]**

- Incrémentation et décrémentation d'un pointeur

  P=&A[i] ➔ **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]

8

Arithmétique des pointeurs(2)

- Soustraction de pointeurs

  P1 et P2 pointent sue le même tableau

  P1-P2
  - <0 si P1 précède P2
  - >0 si P2 précède P1
  - ==0 si P1 = P2

- Comparaison de pointeurs

  <, >, <=, >=, ==, !=. Revient à comparer leurs indices respectifs dans le tableau

9

Exercices

- Afficher le maximum d'un tableau Tab en utilisant le formalisme pointeurs.
- Soit P un pointeur qui pointe sur un tableau A:

  int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90}; les adresses: xo,x2,x4,x6,x8,x10,x12,x14,x16.

  int *P; P = A;

  Quelles valeurs ou adresses fournissent ces expressions:

  a) *P+2
  b) *(P+2)
  c) P+1
  d) &A[4]-3
  e) A+3
  f) &A[7]-P
  g) P+(*P-10)
  h) *(P+*(P+8)-A[7])

10

Exercices

- **Afficher le maximum d'un tableau Tab en utilisant le formalisme pointeurs.**
- **Soit P un pointeur qui pointe sur un tableau A:**

  int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90}; **les adresses: xo,x2,x4,x6,x8,x10,x12,x14,x16.**

  int *P; P = A;

  Quelles valeurs ou adresses fournissent ces expressions:

  a) ***P+2 = 14**
  b) ***(P+2)=34**
  c) **P+1=x2**
  d) **&A[4]-3 =x2 c'est &A[1]**
  e) **A+3=x6 (&A[3])**
  f) **&A[7]-P &A[7]-&A[0]=7**
  g) **P+(*P-10)=p+2=x4**
  h) ***(P+*(P+8)-A[7]) =*(p+(90-89))=*(p+1)=23**

11

correction

Exercice1:

```
Main()
{
        int max,t[10],*p;
        max=*t;
        for(p=t;p<t+10;p++)
            if(*p>max)
                max=*p;
        printf(« %d »,max);
}
```

12

correction

Exercice2:

```
Main(){
        int a[10],*p,aux;
        // saisir
        for(p=a;p<a+10;p++)
            scanf(« %d »,p);
        for(p=a;p<a+5;p++)
            // il faut permuter *p et *(a+9-(p-a)) //
            {
            aux=*p;
            *p=*(a+9-(p-a));
            *(a+9-(p-a))=aux;}
```

13

Exercices

- **Ecrire un programme qui range les éléments d'un tableau A du type int dans l'ordre inverse en utilisant le formalisme pointeur.**

- **➔ Ecrire un programme qui lit un entier X et un tableau A du type int au clavier et élimine toutes les occurrences de X dans A en tassant les éléments restants. Le programme utilisera deux pointeurs P et P1 pour parcourir le tableau.**

14

Correction

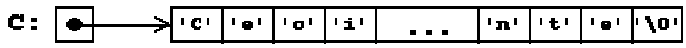
Exercice3:

```
main(){
        int a[10],*p,*p1,x,n=0;
        for(p=a;p<a+10-n;p++)
        {
           while (*p==x){
                 *p=*(p+1);
                 n++;}
        }
        //afficher le tableau// (p=a;p<a+10-n;p++)
```

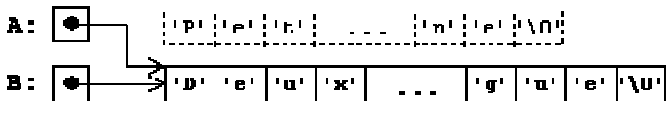
15

6. Pointeurs et chaînes de caractères

- **char *C;**
  C = "Ceci est une chaîne de caractères constante";



- **char *A = "Petite chaîne";**
  char *B = "Deuxième chaîne un peu plus longue";
  A = B;



16

Exercices

- Ecrire un programme qui lit une chaîne de caractères CH et affiche la longueur de la chaîne à l'aide d'un pointeur P. Le programme n'utilisera pas de variables numériques.

- Ecrire un programme qui lit une chaîne de caractères CH et détermine le nombre de mots contenus dans la chaîne. Utiliser un pointeur P, une variable logique, la fonction isspace et une variable numérique N qui contiendra le nombre des mots.

17

7. Fonctions de recherche sur les chaînes

- **strchr (chaîne, caractère)**

  Recherche, dans chaîne la première position où apparaît le caractère mentionné

- **strrchr (chaîne, caractère)**

  Réalise le même traitement que strchr, mais en explorant la chaîne mentionnée à partir de la fin.

- **strstr (chaîne, sous-chaîne)**

  Recherche dans chaîne la première occurrence complète de la sous-chaîne mentionnée.

18

Exemples

- Afficher le nombre d'occurrences du caractère 'A' dans une chaîne en utilisant strchr.
- Afficher le nombre d'occurrences de la sous-chaîne « OU » dans une chaîne en utilisant strstr.

19

8. Pointeurs et tableaux à deux dimensions

```
int M[4][10] = {{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9},
                {10,11,12,13,14,15,16,17,18,19},
                {20,21,22,23,24,25,26,27,28,29},
                {30,31,32,33,34,35,36,37,38,39}};
```

→ M pointe vers M[0] = {0,1,2,3,4,5,6,7,8,9}

→ M+1 pointe vers M[1] = {10,11,12,13,14,15,16,17,18,19}.

→ M+i pointe vers le tableau M[i].

20

Accès au éléments d'un tableau à deux dimension via les pointeurs

```
int *P;
P = (int *)M;  /* conversion forcée */
```

➔traiter M à l'aide du pointeur P comme un tableau unidimensionnel de dimension 4*10.

```
/*Calculer la somme de tous les éléments du tableau*/
        int *P;
        int I, SOM;
        P = (int*)M;
        SOM = 0;
        for (I=0; I<40; I++)
          SOM += *(P+I);
```

21

Exercice

Ecrire un programme qui lit une matrice A de dimensions N et M au clavier, affiche A et sa transposée en utilisant le formalisme pointeur.

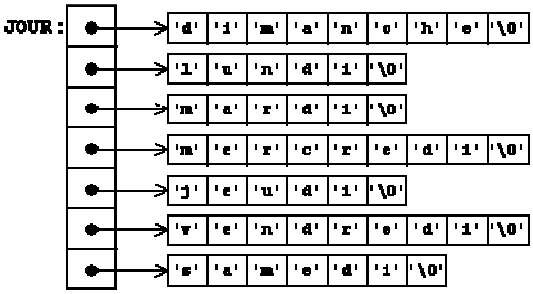
22

9. Tableaux de pointeurs

<Type> *<NomTableau>[<N>] déclare un tableau <NomTableau> de <N> pointeurs sur des données du type <Type>.

Exemple

```
char *JOUR[] = {"dimanche", "lundi", "mardi",
                "mercredi", "jeudi", "vendredi",  "samedi"};
```



23

Tableaux de pointeurs(2)

```
/*Afficher les 7 jours de la semaine*/
   int I;  for (I=0; I<7; I++)
   printf("%s\n", JOUR[I]);

/*Afficher les premières lettres des 7 jours de la semaine*/
    int I;
    for (I=0; I<7; I++)
          printf("%c\n", *JOUR[I]);

/*Afficher la troisième lettre de chaque jour de la semaine*/
     int I;
     for (I=0; i<7; I++)
          printf("%c\n",*(JOUR[I]+2));
```

24

Exercice

Considérez les déclarations de NOMS1 et NOMS2:

```
char *NOMS1[] =    {"Ali",  "Mohamed",  "Sirine", "Farah",
"Meriem" };

char NOMS2[][16] = {"Ali",  "Mohamed",  "Sirine", "Farah",
"Meriem" };
```

Représenter graphiquement la mémorisation des deux variables NOMS1 et NOMS2.

25

10. Allocation dynamique de mémoire

malloc(<N>) fournit l'adresse d'un bloc en mémoire de <N> octets libres ou la valeur zéro s'il n'y a pas assez de mémoire.

Exemple 1

```
char *T ;  T = malloc(4000);
```

Exemple 2

```
int X;
int *PNum;
printf("Introduire le nombre de valeurs :");
scanf("%d", &X);
PNum = malloc(X*sizeof(int));
```

26

Exercice

Ecrire un programme qui lit 10 phrases d'une longueur maximale de 200 caractères au clavier et qui les mémorise dans un tableau de pointeurs sur char en réservant dynamiquement l'emplacement en mémoire pour les chaînes. Ensuite, l'ordre des phrases est inversé en modifiant les pointeurs et le tableau résultant est affiché.

27

free(<Pointeur>) libère le bloc de mémoire désigné par le <Pointeur>; n'a pas d'effet si le pointeur a la valeur zéro(NULL).

Attention !

- **Ne pas libérer de la mémoire qui n'a pas été allouée par malloc.**
- **La fonction free ne change pas le contenu du pointeur; il est conseillé d'affecter la valeur zéro au pointeur immédiatement après avoir libéré le bloc de mémoire qui y était attaché.**
- **Si nous ne libérons pas explicitement la mémoire à l'aide free, alors elle est libérée automatiquement à la fin du programme.**

28

Exercices

1) Déclarer dynamiquement un tableau de N entiers (N étant introduits au clavier), saisir le contenu des éléments du tableau.

Afficher la valeur minimale et maximale du tableau.

2) <u>Allocation dynamique d'un tableau de pointeurs :</u>

Reprendre l'exemple de la page 19 en déclarant le tableau JOUR de manière dynamique.

3) <u>Allocation dynamique d'une matrice :</u>

Ecrire un programme qui permet de saisir une matrice d'entiers de dimensions L et C et d'afficher cette matrice.

29