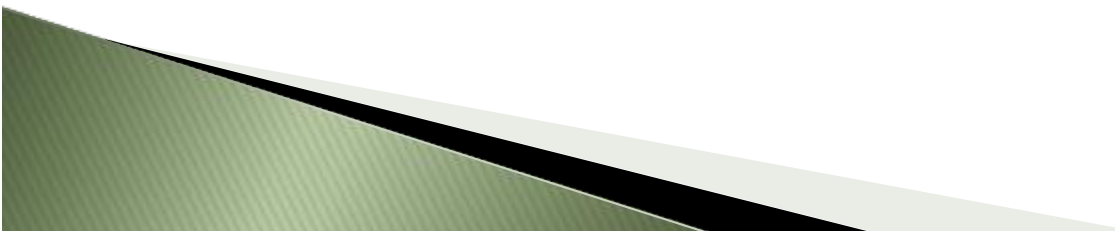


# Structures de données et algorithmes – TP5

Iulia-Cristina Stanica  
iulia.stanica@gmail.com

# Objectifs pour aujourd'hui

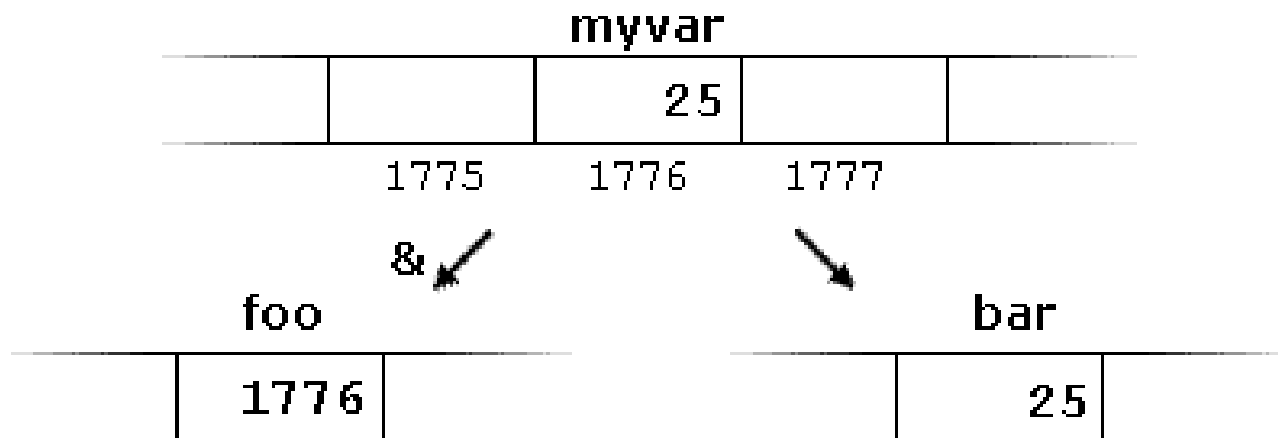
- ▶ Pointeurs :)
  - ▶ Pointeurs et fonctions
  - ▶ Pointeurs et tableaux/matrices
  - ▶ Pointeurs vers pointeurs
- 

# Pointeurs

- ▶ Un pointeur est une variable qui contient **l'adresse** d'une autre variable (une référence vers une autre variable)
- ▶ Les pointeurs mémorisent l'adresse, donc quand on attribue une valeur au pointeur, celle-ci doit être une **adresse** (on utilise &)

# Pointeurs

```
myvar = 25;  
foo = &myvar;  
bar = myvar;
```



# Pointeurs

- ▶ Déclaration pointeur vers une variable de type entier/double:
  - `int *px;` **ou** `int* px;`
  - `double *py;` **ou** `double* py;`
  - `int xVal = 5;`
  - `int *px = &xVal;` //l'adresse où se trouve xVal est sauvegardée dans le pointeur
- ▶ Pour obtenir le **contenu** de la variable indiquée par le pointeur, on utilise la déréréférence (on utilise \*):
  - `int contenu = *px;`

# Exemple 1

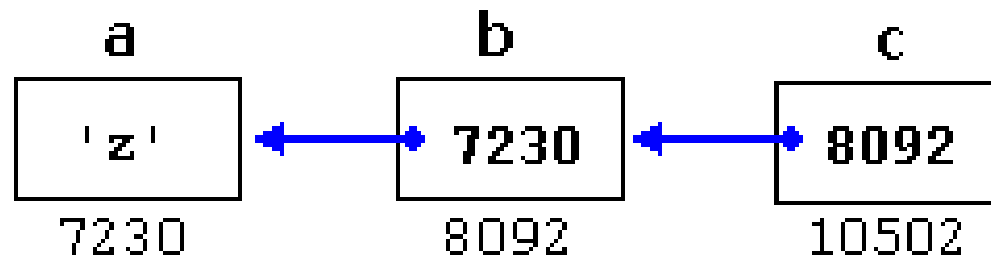
```
#include <iostream>
using namespace std;

int main () {
    int *px; //declaration pointeur
    int x = 5; //declaration variable entiere
    px = &x; //assignation adresse variable au pointeur

    cout<<"Adresse de la variable x: "<<&x<<endl;
    cout<<"Valeur de la variable x: "<<x<<endl;
    cout<<"Valeur pointeur px: "<<px<<endl;
    cout<<"Px point a: "<<*px<<endl;
    // & – reference, * – dereference
    return 0;
}
```

# Pointeurs

```
1 char a;  
2 char * b;  
3 char ** c;  
4 a = 'z';  
5 b = &a;  
6 c = &b;
```



# Exemple 2

```
// more pointers
#include <iostream>
using namespace std;
```

```
int main ()
```

```
{
    int firstvalue = 5, secondvalue = 15;
    int * p1, * p2;
```

```
    p1 = &firstvalue; // p1 = address of firstvalue
    p2 = &secondvalue; // p2 = address of secondvalue
    *p1 = 10;          // value pointed to by p1 = 10
    *p2 = *p1;          // value pointed to by p2 = value pointed
                        // to by p1
    p1 = p2;           // p1 = p2 (value of pointer is copied)
    *p1 = 20;          // value pointed to by p1 = 20
```

Quel est le resultat?

```
    cout << "firstvalue is " << firstvalue << '\n';
    cout << "secondvalue is " << secondvalue << '\n';
    return 0;
}
```



# Exemple 3 – pointeurs et fonctions

```
#include <iostream>
using namespace std;
```

```
int swap1(int a, int b) //parametres transmis par valeur
```

```
{
    int x = a;
    a = b;
    b = x;
}
```

```
int swap2(int &a, int &b) //parametres transmis par adresse
```

```
{
    int x = a;
    a = b;
    b = x;
}
```

```
int swap3(int *a, int *b) //les parametres sont des pointeurs,
on doit faire l'appel avec des adresses
```

```
{
    int x = *a;
    *a = *b; *b = x;
}
```

```
int main()
{
```

```
    int a = 15;
    int b = 38;
```

```
    int *pa;
    pa = &a;
```

```
    int *pb;
    pb = &b;
```

```
    swap1(a,b);
    cout<<a<<"|"<<b<<"\n";
```

```
    swap2(a,b);
    cout<<a<<"|"<<b<<"\n";
```

```
    swap3(&a,&b); //appel avec
les adresses
```

```
    cout<<a<<"|"<<b<<"\n";
```

```
    swap3 (pa,pb);
    cout<<a<<"|"<<b<<"\n";
```

```
    return 0;
```

```
}
```

# Ex 4 – pointeurs et tableaux de caractères

**OBS: Un array est toujours un pointeur vers son premier élément!**

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{  
  char s[] = "Hello world"; //tableau de caractères
```

```
  char *ps;
```

```
  ps = &s[0]; //ps – pointeur vers le premier élément du string
```

```
  char *pa;
```

```
  pa = s; //même chose, pointeur vers le premier élément du string
```

```
  cout<<"ps point a:"<<*ps<<endl; //par la déréréférence, on obtient la première lettre (référéncée)
```

```
  //Cas spécial: on affiche une séquence de caractères au lieu d'une adresse (l'opérateur << interprète  
  char* comme un string (surcharge des fonctions))
```

```
  cout<<ps<<endl;
```

```
  cout<<(void *)ps<<endl; //en vérité le pointeur contient une adresse (on doit faire la conversion  
  forcée pour afficher l'adresse)
```

```
  while (*ps) //si le pointeur référence une valeur
```

```
{
```

```
  cout<<"L'adresse: "<<(void *) ps<<" et le contenu: "<<*ps<<endl;
```

```
  ps++; //on passe a l'adresse suivante
```

```
}
```

```
  return 0; }
```

# Exemple 5 – pointeurs et matrice

```
#include <iostream>
using namespace std;
```

```
int main() {
    int mat[3][3] = {{2,3,4}, {5,6,7}, {8,9,10}};
    int *pmat = mat[0]; //mat[0] contient l'adresse
                        //du premier element de la matrice
```

```
// int *pmat = &mat[0][0]; – ca fait quoi?
```

```
    for (int i=0; i<9; i++)
        cout<<*(pmat++) << " ";
```

```
return 0;
}
```

# Exemple 6 – pointeurs, fonctions et tableaux

```
#include <iostream>
using namespace std;

void modify1 (int a[])
{
    a[2]=15;
}

void modify2 (int *a)
{
    *(a+2) = 15; //meme effet
                 que modify1
}
```

```
int main()
{
    int x[]={1,2,3,4};
    cout<<"Avant l'appel de la fonction:
"<<x[2]<<endl;
    modify1(x);

    int *p=x; //même chose que:
              //int *p = &x[0];
              // modify2(p);
              // modify2(x);
    cout<<"Après l'appel de la fonction:
"<<x[2]<<endl;

    return 0;
}
```

# Exemple 7 – Pointeurs vers pointeurs

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string day[] = {"Lundi", "Mardi", "Mercredi"};
    string *pday = day; //string *pday = &day[0]; -> meme chose
    string **ppday = &pday; //pointeur vers un pointeur

    cout<<"A: Valeur: "<<*pday<<" , a l'adresse: "<<pday<<endl;

    cout<<"B: Valeur: "<<**ppday<<" , a l'adresse du pointeur niveau 1 (pday): " << *ppday
        <<endl;
    cout<<" et l'adresse du pointeur niveau 2 (ppday): "<< ppday<<endl;

    if (*ppday==pday)
        cout<<"Meme valeur d'adresses!";

    return 0;
}
```

# Exercices

- Soient la séquence suivante:

```
int *p;  
int i;  
int k;  
i = 42;  
k = i;  
p = &i;
```

Après laquelle de ces assignations la valeur de i sera 75?

- A. `k = 75;`
- B. `*k = 75;`
- C. `p = 75;`
- D. `*p = 75;`
- E. Deux ou plusieurs réponses vont changer la valeur de i.

Ex 1). Ecrire un programme qui lit du clavier 6 valeurs entières. Les 6 valeurs seront stockées dans un tableau **à l'aide d'un pointeur**. Ensuite, affichez les éléments du tableau sur l'écran (pour voir si l'utilisation du pointeur a stocké les valeurs dans le tableau).

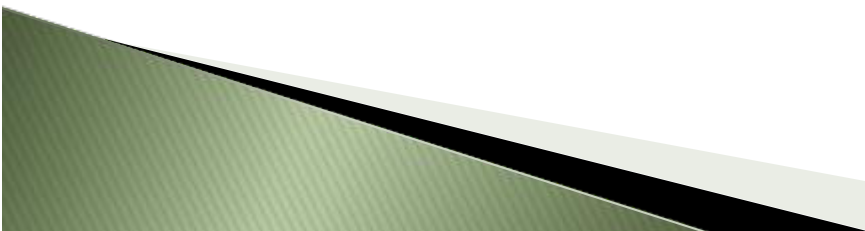
### Hint: (étapes)

- ▶ Déclaration pointeur vers tableau
- ▶ Sauvegarder les valeurs dans le pointeur avec une boucle for (attention à l'incrémentement des adresses du pointeur!!)
- ▶ Affichage éléments tableau avec une boucle for classique (utilisez le tableau, pas le pointeur).

Ex 2). Modifiez la solution de l'exercice 1 pour imprimer les éléments du tableau dans l'ordre inverse **à l'aide d'un pointeur**.

Ex 3). Inverser le tableau en utilisant uniquement des pointeurs. Utilisez **une fonction** avec 2 paramètres: un pointeur au tableau et la taille du tableau. Affichez le contenu du tableau sur l'écran.

Ex:  $a = \{12, 3, 4, 6, 10, 15\}$  deviendra après l'inversion:  $a = \{15, 10, 6, 4, 3, 12\}$



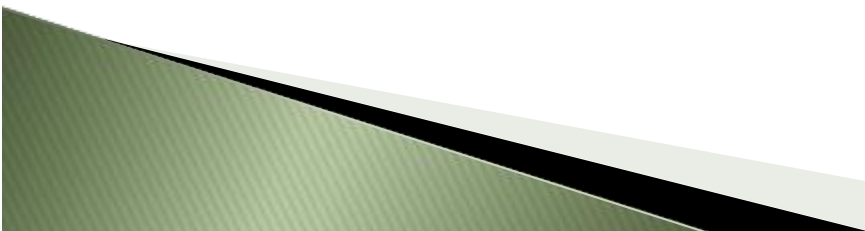


Ex. 4) Écrivez un programme qui extrait le département d'une suite. Utilisez **une fonction** de type:

**char \*getDep (char \*p)**

qui retourne un pointeur vers la première lettre du département. Le paramètre p est le pointeur à la suite.

Ex: On donne la suite: "Slatina, OT" et on affiche à l'aide du pointeur: "Le département est OT".



Ex. 5) Écrivez un programme qui remplace les virgules d'une suite de caractères par des espaces blancs. Utilisez une fonction du type:

**char \*commaReplacer(char \*p)**

Ex: "College,of,London," devient "College of London".