Tp 1 de C++. L3 Physique

M. Ismail

2012-2013

Les comptes-rendus sont à rendre au plus tard lors de la prochaine séance

1 Les conditions, opérateurs logiques

Exercice 1

1. Quelle est la différence entre les opérateurs `=` et `==` ? Essayez le programme suivant (fichier `tp1.exo1.cpp`)

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

int main()
{
   int x, y;
   x = 1;
   y = 2;
   if (x = y)
     {
        cout << "vrai\n";
     }
   else
     {
        cout << "faux\n";
     }
   return 0;
}
```

2. Quel résultat donne le programme ? Essayer d'initialiser les valeurs de x et de y à zéro
3. Pourquoi le compilateur accepte cette construction qui a priori semble incorrecte, en quoi est-elle effectivement correcte ?

2 Boucles

Exercice 2

Écrire trois programmes qui comptent de 0 à 10 en utilisant les boucles `for`, `while` et `do. . .while`. Vérifiez les conditions aux bords.

3 Pointeurs et tableaux

Exercice 3

Parfois, on entend dire qu'un pointeur et un tableau sont la même chose.

1. Quelle est la différence entre un pointeur `int *x` et un tableau `int x[10]`.
2. Dans quel cas l'un peut être utilisé à la place de l'autre ?

Exercice 4

Considérer le programme suivant, il définit un tableau `tab` contenant 10 entiers puis le remplit avec les nombres $1000, 1001, \cdots, 1009$. Ensuite il définit un pointeur `ptr` qui référence le premier élément de tab (fichier `tp1.exo4.cpp`).

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

int main()
{
   int tab[10];
   int *ptr;
   ptr=tab;
   for (int i=0;i<10;i++)
     {
        tab[i]=1000+i;
     }
   cout<<"\nvaleur de ptr "<<ptr;
   cout<<"\nvaleur de &ptr "<<&ptr;
   cout<<"\nvaleur de *&ptr "<<*&ptr<<endl;

   cout<<"\nvaleur de ptr+1 "<<ptr+1;
   cout<<"\nvaleur de *ptr+1 "<<*ptr+1;
   cout<<"\nvaleur de (*ptr)+1 "<<(*ptr)+1;
   cout<<"\nvaleur de *(ptr+1) "<<*(ptr+1)<<endl;

   cout<<"\nvaleur de tab+1 "<<tab+1;
   cout<<"\nvaleur de *tab+1 "<<*tab+1;
   cout<<"\nvaleur de (*tab)+1 "<<(*tab)+1;
   cout<<"\nvaleur de *(tab+1) "<<*(tab+1)<<endl;
```

```
   cout << "\nvaleur de l'element 0 "<<*ptr;
   cout << "\nvaleur de l'element 5 "<<*(ptr+5)<<endl;
   return 0;
}
```

1. Que représente `*ptr`, `&ptr`, `*&ptr` ?
2. Quelle est la différence entre : `*ptr+1`, `(*ptr)+1`, `*(ptr+1)`, `*tab+1`, `(*tab)+1`, `*(tab+1)`, `tab[1]` ?
3. En complément de cela, et pour s'aider, on pourra tester le programme (fichier `tp1.exo4bis.cpp`), et analyser les résultats, en faisant notamment un dessin expliquant ce que représente chaque expression visualisée.

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

int tab[3][4]={{100,102,104,106},
               {200,202,204,206},
               {300,302,304,306}};
int main()
{
   int *ptr;
   ptr=tab[2];
   cout<<"\nvaleur de tab[2] "<<tab[2];
   cout<<"\nvaleur de &tab[2][0] "<<&tab[2][0];
   cout<<"\nvaleur de ptr "<<ptr;
   cout<<"\nvaleur de *ptr "<<*ptr;
   cout<<"\nvaleur de &ptr "<<&ptr<<endl;

   cout<<"\nvaleur de tab[2][1] "<<tab[2][1];
   cout<<"\nvaleur de &tab[2][1] "<<&tab[2][1];
   cout<<"\nvaleur de ptr+1 "<<ptr+1;
   cout<<"\nvaleur de *(ptr+1) "<<*(ptr+1)<<endl;

   cout<<"\nvaleur de tab "<<tab;
   cout<<"\nvaleur de tab[0] "<<tab[0];
   cout<<"\nvaleur de &tab[0][0] "<<&(tab[0][0])<<endl;

   cout<<"\nvaleur de tab+1 "<<tab+1;
   cout<<"\nvaleur de &tab[1] "<<&tab[1];
   cout<<"\nvaleur de tab[1] "<<tab[1];
   cout<<"\nvaleur de &tab[1][0] "<<&tab[1][0];
   cout<<"\nvaleur de *(tab+1) "<<*(tab+1);
   cout<<"\nvaleur de tab[1][0] "<<tab[1][0]<<endl;

   cout<<"\nvaleur de **(tab+1) "<<**(tab+1);
   cout<<"\nvaleur de *(*tab+1) "<<*(*tab+1)<<endl;


   return 0;
}
```

Exercice 5

Modifier le programme de l'exercice précédent de sorte qu'il affiche sur la console le contenu du tableau en incrémentant le pointeur.

Exercice 6

1. Quelles sont les significations des définitions suivantes :
   - `char tab[20][10];`
   - `char *tab[20];`
   - `char **tab;`
2. Sachant que sur les machines utilisées en TP (architecture 32bit) un `char` occupe 1 octet et un pointeur occupe 4 octets, indiquer la quantité de mémoire occupée par chacune des définitions. Pour chaque cas faire un petit schéma de la disposition des variables dans la mémoire.

Exercice 7

1. Considérer le programme suivant :

```
char tab[20][10];
int main()
{
  char *ptr;
  ptr = tab[3];
  return 0;
}
```

   Sur quel élément du tableau pointe ptr ?

2. Modifiez le programme de façon à remplir le tableau avec les valeurs :

```
tab[0][0] = 0   tab[0][1] = 1   · · · tab[0][9] = 9
tab[1][0] = 100 tab[1][1] = 101 · · · tab[1][9] = 109
...
...
...
tab[19][0]=1900 tab[19][1]=1901 · · · tab[19][9]=1909
```

   Pour ce faire, n'indexez pas le tableau, mais utilisez le pointeur ptr ; pour passer d'un élément à l'autre incrémentez le pointeur. Cette construction rend l'exécution du programme beaucoup plus rapide que l'indexage du tableau. Pour s'assurer du bon fonctionnement du programme, ajoutez une procédure qui affiche le tableau sur la console, en utilisant directement `tab`.