Travaux Dirigés de C++.

Maitrise Informatique.

—*Julien Allali* - `allali@univ-mlv.fr` —

1 g++, References

▸ **Exercice 1.** *Hello World !*
Ecrire un fichier hello_world.cpp *qui affiche "Hello World !" sur la sortie standard en utilisant la bibliothèque* iostream, *on accédera aux objets et fonctions de cette bibliothèque en utilisant le prefixe de namespace* std::.

Compiler votre fichier avec la commande g++ -g -Wall hello_world.cpp

Positionner la variable d'environnement CXXFLAGS *sur* -g -Wall *puis entrer la commande* make hello_world.

▸ **Exercice 2.** *swap*
Ecrire une fonction Swap qui prend en argument 2 entiers et échange leurs valeurs.
Cette fonction aura comme type de retour void.

▸ **Exercice 3.** *Pointeurs et références:*
Ré-écrire la fonction suivante à l'aide de références et de const:

```
void PrintInt( int *i)
{
   if (i!=0)
      {
         cout<<*i;
      }
}
```

Les 2 fonctions suivantes, sont-elles équivalentes (justifier)?

```
void f1(char *t)
{
  if (t!=0)
     {
       while(*t != 0)
          { cout<<*t; t++;}
     }
}

void f2(char &t)
{
  while(t!=0)
    {cout << t; t++;}
}
```

▸ **Exercice 4.** *Affectation d'une valeur de retour.*
Ecrire une fonction **Nop** *telle que l'on puisse faire:*

```
#include<cstdio>
...
int a=5;
Nop(a)=10;
std::printf(‘‘%d’’,a); // --> affiche 10.
```

▸ **Exercice 5.** *Affectation d'une valeur de retour.*
Ecrire une fonction **Elements** *telle que l'on puisse faire:*

```
#include<cstdio>
...
int a[10];
Elements(a,0)=10;
std::printf(``%d'',a[0]); // --> affiche 10.
std::printf(``%d'',Elements(a,0)); // --> affiche 10.
```

▸ **Exercice 6.**
Dans quels cas doit-on utiliser les références et le const ?

2 ASSERT

▸ **Exercice 7.** *Factorielle :*
Ecrire la fonction factorielle en utilisant la macro assert pour verifier la valadité de l'argument ($\geq$ 0) (man assert). Comment activer ou désactiver l'utilisation de assert au moment de la compilation du fichier (man gcc, option -D)?

3 Export, Surcharge

▸ **Exercice 8.** *Export :*
Compiler le fichier precedent facto.cpp *en objet (*g++ -c *ou bien* make facto.o*). En utilisant le programme* nm *consulter la liste des symboles présents dans le fichier* facto.o *(nm facto.o et nm -C facto.o) et trouver celui correspondant à la fonction* factorielle*. Déclarer la fonction* factorielle *comme étant un fonction C en utilisant* extern "C"*, recompiler puis retrouver le symbole correspondant à la fonction.*

▸ **Exercice 9.** *affichage:*
Ecrire une fonction initLog *initialisant une session de journalisation avec* openlog *(man openlog). Cette fonction prendra l'identifiant en argument sous la forme d'un pointeur sur une chaîne de caractère, par défaut ce pointeur sera initialisé à* "anonymous"*. On veillera à activer l'option LOG_PERROR.*

Compiler et trouver le symbole correspondant à la fonction initLog *généré dans le binaire.*

Ecrire quatre fonctions logData*, la première prend un entier par référence en argument, la deuxième un caractère, la troisième une chaîne de caractère et la quatrième un entier par référence constante. Ces fonctions utilisent la fonction* syslog *pour effectuer l'affichage de l'argument.*

```
anonymous[6250]: logData: (char ) c
anonymous[6250]: logData: (const int &) 3
```

Compiler, tester avec différentes valeurs et regarder les symboles générés dans le binaire avec nm*.*

Dans quels cas la fonction logData(const int &)*est utilisée ? Que se passe-t-il si l'on modifie la première fonction afin qu'elle prenne un entier comme argument (par copie) ?*

4 Allocation.

▸ **Exercice 10.** *new:*
L'allocation de données se fait à l'aide de l'opérateur new*. On distingue deux types d'allocations: l'allocation d'objet de type A avec* a= new A *ou* a=new A(valeur) *et l'allocation de tableaux avec* t=new A[taille]*. La délétion dans le premier cas se fait a l'aide de l'opérateur* delete p *et à l'aide de* delete [] p *dans le deuxième cas.*

Ecrire un programme qui alloue un tableau d'entiers.

Declarer un deuxième pointeur sur le même tableau tel que l'on ne puisse pas modifier la valeur de ce pointeur par la suite. Tester.

Déclarer un troisième pointeur sur ce tableau tel que l'on ne puisse pas modifier les valeurs du tableau à l'aide de ce pointeur. Tester.

On verifiera que l'allocation/désallocation est bien faite à l'aide du programme valgrind *(*valgrind a.out*).*

5 Bibliographie:

- Le Language C++, Bjarne Stroustrup, CompusPress.
- Effective C++, Scott Meyers, Addison Wesley.
- More Effective C++, Scott Meyers, Addison Wesley.