# TD 1 – Du C au C++

## Exercice 1

Écrire le programme suivant (correct en C comme en C++), en ne faisant appel qu'aux nouvelles possibilités du langage C++ :

```
#include <stdio.h>
#include <malloc.h>
#define NbrValeurs 25
void main()
{
      float *Valeurs, Somme=0.0;
      int i;

      /*réservation d'un espace mémoire pour les valeurs*/
      Valeurs=(float *)malloc(sizeof(float)*NbrValeurs);

      for(i=0;i<NbrValeurs;i++)
      {
            printf("donner la moyenne N°%d\n",i);
            scanf("%f",&Valeurs[i]);
            Somme= Somme + Valeurs[i];
      }

      printf("La somme de ces %d réels=%f",NbrValeurs,Somme);
      free(Valeurs);
}
```

## Exercice 2

Soient les déclarations (C++) suivantes :

```
int fct (int) ;                 // fonction 1
int fct (float) ;               // fonction 2
void fct (int, float) ;         // fonction 3
void fct (float, int) ;         // fonction 4
int n, p ;
float x, y ; char c ;
double z ;   short s ;    long t ;
```

Les appels suivants sont-ils corrects et, si oui, quelles seront les fonctions effectivement appelées et les conversions éventuellement mises en place ?

**a.** `fct(c) ;`
**b.** `fct(n,x) ;`
**c.** `fct(c,n ) ;`
**d.** `fct(n,p) ;`
**e.** `fct(n,c) ;`
**f.** `fct(n,z) ;`
**g.** `fct(z,z) ;`
**h.** `fct(c,z) ;`
**i.** `fct(t,x) ;`
**j.** `fct(t) ;`
**k.** `fct(s) ;`
**l.** `fct (t,z) ;`
**m.** `fct(t,c) ;`

## Exercice 3

Soient les prototypes de fonctions suivants :

1. `void f (int x) ;`
2. `void f (int &x) ;`
3. `void f (const int & x) ;`
4. `void f (int *x) ;`

et soient les déclarations suivantes :

```
int n1(7) ;
int *n2=new int(9) ;
int &n3=*new int (13) ;
const int n4(18) ;
const int *n5=new int(15) ;
const int &n6=*new int(14) ;
```

Remplir le tableau suivant en indiquant si chacun de ces appels est correct ou non pour les prototypes 1,2,3,4 :

| Appel | Prototype 1 | Prototype 2 | Prototype 3 | Prototype 4 |
|---|---|---|---|---|
| f(n1) | | | | |
| f(&n1) | | | | |
| f(n2) | | | | |
| f(*n2) | | | | |
| f(n3) | | | | |
| f(&n3) | | | | |
| f(n4) | | | | |
| f(&n4) | | | | |
| f(n5) | | | | |
| f(*n5) | | | | |
| f(n6) | | | | |
| f(&n6) | | | | |

**Exercice 4**

Soit le programme suivant :

```
int g1=0;
int g2=0;
int *ptr;
int *& f4(int x=0);

int f1(int *p)
{  p=&g2;
   (*p)++;
   return (*p);
}
int &f2(int *&p)
{  p=&g2;
   (*p)++;
   return *f4();
}
int *f3(int &x)
{  x++;
   return (&x);}
```

```
int *& f4(int x){
x++;
(*ptr)+=x;
return ptr;
}

void main()
{
      1. ptr=&g1;
      2. int &a=*ptr;
      3. int *ptr1=f3(a);
      4. (*ptr1)++;
      5. int *&ptr2=f4(f2(ptr1));
      6. (*ptr1)++;
      7. (*ptr2)++;
      8. a+=f1(ptr2);
      9. (*ptr2)++ ;
}
```

Donner la valeur des variables g1, g2, ptr, ptr1, ptr2, et a après les instructions numérotées de 1 à 9.

# TD1 - Du C au C++

## Exercice n°1

```
#include <iostream>
using namespace std;
const int NbValeurs 25;
Void main ()
{
    float Somme = 0.0;
    float * Valeurs = new float [NbValeurs];
    for (int i=0; i < NbValeurs ; i++)
    {   cout << " donner la moyenne N° :" << i << " \n";
        cin >> Valeurs [i];
    }   Somme = Somme + Valeurs [i];
        cout << " la Somme de ces " << NbValeurs
             << " réels = " << Somme ;
        delete [] Valeurs;
}
```

## Exercice n°2

a) fct (c) : fonction 1

b) fct (n, x) : fonction 3

c) fct (c, n) : ambiguité
   (fcn3, fc1)

d) fct (n, p) : ambiguité
   (f3, f1)

e) fct (n, c) ambigvité
   (f3, f1)

f) fct (n, z) : fonction 3
   (f3, f3)

g) fct (z, z) : ambiguité
   (f1, f3)

h) fct (c, z) : fonction f3
   (f3, f1 ou f2)

i) fct (t, x) : fonction 3
   (f1 ou f2, f3)

j) fct (t) ambiguité

k) fct (s) : fonction 1

l) fct (t, z) : ambiguité
   (f2 ou f1, f1 ou f2)

m) fct (t, c) : fonction 1

## Exercice n°3

passage par référence et correspondance exacte

void f(int &x) void f(const int &x) void f(int *x)

| Appel | Prototype 1 | Prototype 2 | Prototype 3 | Prototype |
|---|---|---|---|---|
| f(n1) | correct | correct | correct | non-comp |
| f(&n1) | non compatible | non-comp | non-comp | correct |
| f(n2) | non compatible | non-comp | non-comp | correct |
| f(*n2) | correct | correct | correct | non-comp |
| f(n3) | correct | correct | correct | non-comp |
| f(&n3) | non-comp | non-comp | non-comp | non comp |
| f(n4) | correct | non-comp | correct | non comp |
| f(&n4) | non-comp | non-comp | non-comp | non-comp |
| f(n5) | non-comp | non-comp | non-comp | non-comp |
| f(*n5) | correct | non-comp | correct | non-comp |
| f(n6) | correct | non comp | correct | non-comp |
| f(&n6) | non-comp | non comp | non comp | non comp |
| | | | | |

Passage par adresse

void f(int &x) : passage par référence

## Exercice n°4

| instruction | g1 | g2 | ptr | ptr 1 | ptr 2 | a |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | &g1 | x | x | ~ |
| 2 | 0 | 0 | &g1 | x | x | 0 |
| 3 | 1 | 0 | &g1 | &g1 | x | 1 |
| 4 | 2 | 0 | &g1 | &g1 | x | 2 |
| 5 | 7 | 1 | &g1 | &g2 | &g1 | 7 |
| 6 | 7 | 2 | &g1 | &g2 | &g1 | 7 |
| 7 | 8 | 2 | &g1 | &g2 | &g1 | 8 |
| 8 | 11 | 3 | &g1 | &g2 | &g1 | 11 |
| 9 | 12 | 3 | &g1 | &g2 | &g1 | 12 |