UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES

Méthodes informatiques pour physiciens

introduction à C++ et
résolution de problèmes de physique par ordinateur

Corrigé du Contrôle Continu #2

`http://dpnc.unige.ch/~bravar/C++2015/CC2`

Professeur : Alessandro Bravar
Alessandro.Bravar@unige.ch

Université de Genève
Section de Physique

Semestre de printemps 2015

1 Questions

1. **Transformez le nombre décimal 153 en binaire, puis octal et hexadcimal. Transformez en dcimal le nombre octal 753.**
   La représentation en base *2* peut être obtenue par divisions successives par 2. La valeur du reste donne la valeur du bit à partir par le bit le plus a droit (poids faible). Pour reprśenter le nombre 153 en base 2 on a besoin de 8 bits ($\text{int}(\log_2 153) + 1 = 8$). Pour obtenir la valeur du bit $b-1$ il faut diviser le nombre par 2^{b-1}.

Bit	Division	Quotient	Reste
0	153/2	76	1
1	76/2	38	0
2	38/2	19	0
3	19/2	9	1
4	9/2	4	1
5	4/2	2	0
6	2/2	2	0
7	1/2	0	1

   Réponse : $153_{10} = 10011001_2$.
   Pour répresenter ce nombre en base octale il faut grouper les bits 3 par 3 et pour la base hexadécimale il faut grouper les bits 4 par 4.
   Réponse : $153_{10} = 10011001_2 = 231_8 = 99_{16}$.

2. **Quelle est la diffrence entre une variable et un pointeur à une variable ?**
   Une variable contient une valeur, tandis que le pointeur contient l'adresse mémoire d'une variable.

3. **Trouvez les erreurs dans le programme suivant :**
```
int main () {
  int *q;
  q = & 7.5;
  cout << q;
}
```
   Les erreurs sont les suivants :
   - on ne peut pas affecter l'adresse mémoire d'une constante (une constante n'a pas d'adresse mémoire) ;
   - il manque l'instruction `return 0;` à la fin du programme ;
   - pour accder à la valeur d'une variable il faut déréférencier le pointeur.
   Version correcte :
```
int main() {
  int *q;
  double x = 7.5;
  q = &x;
  cout << *q;
  return 0; }
```

4. **Considrez la dclaration et affectation suivante :**
```
double tab[4] = {1., 2., 3. };
```
   Quelles sont les valeurs de chaque élément du tableau ? Que vaut l'élément `tab[4]` **?**
```
tab[0] = 1.
```

```
tab[1] = 2.
tab[2] = 3.
tab[3] = 0.
```
L'élément `tab[4]` n'est pas défini, car il est dehors du tableau.

5. `int x = 6;`
   Qu'affichera l'instruction suivante : `cout << *&x;` ?
   Cette instruction affichera `6`, i.e. la valeur de la variable `x` à l'adresse mémoire `&x`.
   Et celle-ci : `cout << &*x;` ?
   Cette instruction générera un message d'erreur, car une variable ne peut pas être déréférenciée (`*x` n'est pas défini).

6. **Considérez le code suivant :**
   ```
   int *x = new int(6);
   delete x;
   x = 0;
   ```
   Quel type de variable est x ?
   `x` est un pointeur sur une variable de type entier initialisé à `6` créé sur le *pile* avec l'allocation dynamique de la mémoire.

7. **Expliquez l'importance des deux dernires lignes du code de la question 6.**
   D'abord on efface la variable du *pile* pour libérer l'espace mémoire occupé par cette variable, puis on initialise à zéro le pointeur sur cet espace mémoire.

8. **Jutilise la méthode du trapèze avec 100 pas pour intégrer une fonction. Suggérez deux idées pour améliorer la précision de mon intégration numérique.**
   Pour améliorer la précision de l'intégration numérique je peux p.ex.
   (i) augmenter la segmentation de l'intervalle d'intégration, ou
   (ii) utiliser une mèthode d'intégration plus précise comme la mèthode de Simpson.

2 Exercices

Exercice 1 : Tableaux et allocation dynamique de la mémoire
Voir le Corrigé 6 / **Moyenne.cpp**. Voici la solution :
Tableau.cpp

```
1  //moyenne des valeurs enregistree dans un tableau
2  //le tableau est cree pendant l'execution du programme
3  //par allocation dynamique de la memoire
4  #include <iostream>
5  #include <cstdlib>
6
7  using namespace std;
8
9  //declaration de la fonction qui calcule la moyenne d'un tableau
10 double moyenne(double tab[], int n);
11
12 int main() {
13   int dim;
14   cout << "Quelle est la taille du tableau ? " << endl;
15   cin >> dim;
16
17   //tableau dynamique
18   double *tab = new double[dim];
19   if (tab==0) exit(0);    //la creation du tableau a echoue
```

```
20
21    //seasi des valeurs du tableau
22    for (int i=0; i<dim; i++) {
23      cout << "Element " << i+1 << " : ";    cin >> tab[i];
24    }
25
26    //calcul de la moyenne avec la fonction moyenne
27    double m = moyenne(tab,dim);
28    cout << "La moyenne du tableau est : " << m << endl;
29
30    //affichage des elements du tableau en dessous de la moyenne
31    cout << "Elements du tableau en dessous de la moyenne" << endl;
32    for (int i=0; i<dim; i++) {
33      if (tab[i] < m) cout << "Element " << i+1 << " : " << tab[i] << endl;
34    }
35
36    delete[] tab;
37
38    return 0;
39  }
40
41  //definition de la fonction moyenne
42  double moyenne(double tab[], int n) {
43    double moyenne=0.;
44    for (int i=0; i<n; i++)
45      moyenne += tab[i];
46    moyenne = moyenne / n;
47
48    return moyenne;
49  }
```

Exercice 2 : Passage des arguments aux fonctions

Voir les notes du cours. Voici le programme :

Arguments.cpp

```
1  #include<iostream>
2  #include<cmath>
3
4  using namespace std;
5
6  //declaration de les fonctions
7  void MoyennesParReference(double x, double y, double &moyenne, double &geo)
       ;
8  void MoyennesParPointer(double x, double y, double *moyenne_ptr, double *
       geo_ptr);
9
10 int main() {
11
12   double a = 173., b = 206.;
13   double moyenne, moyenne_geometrique;
14
15   MoyennesParReference(a, b, moyenne, moyenne_geometrique);
16   cout << "En utilisant le passage par reference la moyenne est " <<
         moyenne << endl;
17   cout << "et la moyenne geometrique est " << moyenne_geometrique << endl;
18
19   MoyennesParPointer(a, b, &moyenne, &moyenne_geometrique);
20   cout << "En utilisant le passage par pointeur la moyenne est " << moyenne
         << endl;
```

```
21    cout << "et la moyenne geometrique est " << moyenne_geometrique << endl;
22
23    return 0;
24  }
25
26  //fonction passage par reference
27  void MoyennesParReference(double x, double y, double &moyenne, double &geo)
        {
28    moyenne = (x+y)/2.;
29    geo = sqrt(x*y);
30  }
31
32  //fonction passage par pointeur
33  void MoyennesParPointer(double x, double y, double *moyenne_ptr, double *
      geo_ptr) {
34    *moyenne_ptr = (x+y)/2.;
35    *geo_ptr = sqrt(x*y);
36  }
```

Exercice 3 : Intégration numérique

Voir les exercices **Integrale2.cpp** (méthode du rectangle), **Integrale3.cpp** (méthodes du trapèze) et **Integrale4.cpp** (formule de Simpson). Pour résoudre l'exercice on peut combiner les trois méthodes dans le même programme. On peut calculer e^{-x^2}de deux manières différentes : `y = exp(-pow(x,2))` ou `y = pow(`M_E`,-x*x)`. Enfin, il faut comparer les résultats de l'intégration numérique avec $\sqrt{\pi}$, p.ex. pour des segmentations différentes de l'intervalle d'intgration (100, 1'000, 10'000, etc.).

Integrale.cpp

```
1   #include <iostream>
2   #include <cmath>
3
4   using namespace std;
5
6   //declaration de la fonction a integrer
7   double fonc(double);
8
9   int main() {
10    //saisi des parametres
11    double min, max;
12    cout << "Entrez un interval d'integration : " << endl;
13    cout << "min = ";    cin >> min;
14    cout << "max = ";    cin >> max;
15    int div;
16    cout << "Entrez un nombre de suos-divisions :" << endl;
17    cout << "N = ";    cin >> div;
18    double deltax = abs(max-min)/div;
19
20    double intRectangle = 0.;
21    double intTrapeze = 0.;
22    double intSimpson = 0.;
23    for (int i=0; i<div; i++) {
24      //methode du rectangle a gauche
25      intRectangle += (fonc(min+i*deltax));
26      //methode du trapeze
27      intTrapeze += (fonc(min+i*deltax) +
28                     fonc(min+(i+1)*deltax));
29      //methode de Simpson
30      intSimpson += (fonc(min+deltax*i) +
```

```
31                      4.*fonc(min+(i+0.5)*deltax) +
32                      fonc(min+(i+1)*deltax));
33    }
34    //la fonction est paire, donc je multiplie par 2
35    intRectangle *= deltax;
36    intTrapeze *= deltax/2.;
37    intSimpson *= deltax/6.;
38    cout << "Integrale avec la methode du rectangle : " << intRectangle
39         << " valeur de pi : " << sqrt(M_PI) << endl;
40    cout << "Integrale avec la methode du trapeze   : " << intTrapeze
41         << " valeur de pi : " << sqrt(M_PI) << endl;
42    cout << "Integrale avec la methode de Simpson   : " << intSimpson
43         << " valeur de pi : " << sqrt(M_PI) << endl;
44
45    return 0;
46  }
47
48  //definition de la fonction a integrer
49  double fonc(double x) {
50    double y = pow(M_E,-x*x);
51    return y;
52  }
```