Déboguer (dépanner) un programme (système) : Les meilleures pratiques de codage pour éviter d'avoir des problèmes

K. Boudjelaba
k.boudjelaba@carnus.fr

10 février 2021

1 Introduction

"Tout le monde sait que le débogage est deux fois plus difficile que l'écriture d'un programme. Donc, si vous êtes aussi intelligent que vous pouvez l'être quand vous écrivez votre code, comment allez-vous déboguer ?"

- Nous écrivons tous du code bogué, acceptez-le et faites avec.
- Écrivez votre code avec le test et le débogage à l'esprit.
- Keep It Simple, Stupid (KISS) (gardez-le simple, stupide).
  - Quelle est la chose la plus simple qui pourrait fonctionner ?
- Ne vous répétez pas.
  - Chaque bout de connaissance doit avoir une seule représentation autorisée et non ambiguë dans un système.
  - Constantes, algorithmes, etc.
- Essayez de limiter l'interdépendance de votre code. (Loose Coupling)
- Donnez à vos variables, fonctions et modules des noms explicites (pas des noms mathématiques).

2 Le travail de débogage

Si vous avez un bogue non banal, c'est là que les stratégies de débogage vont rentrer en ligne de compte. Il n'y a pas de solution miracle, les stratégies aideront. Pour déboguer un problème donné, la situation favorable est quand le problème est isolé dans un petit nombre de lignes de code, en dehors de framework ou de code d'application, avec un cycle court de modification, lancement, échec.

- Faites échouer le code de façon fiable : trouvez un cas de test qui fait échouer le code à chaque fois.
- Diviser et conquérir : une fois que vous avez un cas de test échouant, isolez le code coupable.
  - Quel module.
  - Quelle fonction.
  - Quelle ligne de code.
- Isolez une petite erreur reproductible : un cas de test.
- Changez une seule chose à chaque fois et réexécutez le cas de test d'échec.
- Utilisez le débogueur pour comprendre ce qui ne va pas.
- Prenez des notes et soyez patient, ça peut prendre un moment.

Une fois que vous avez procédé à cette étape, isolez un petit bout de code reproduisant le bogue et corrigez celui-ci en utilisant ce bout de code, ajoutez ce code dans votre suite de test.

3 Programmes C++

a. Soit le programme ci-dessous :

Ce programme prend un entier positif de l'utilisateur (supposons que l'utilisateur a entré n) puis, ce programme affiche la valeur de la somme $1 + 2 + 3 + \cdots + n$ et la moyenne de la séquence des nombres $(1, 2, \cdots n)$.

Programme 1 C++ BTS SN

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

int main()
{
    int n, sum = 0;
    cout << "Entrer un entier positif : ";

    for (int i = 1; i <= n; ++i) {
        sum *= i;
    }
    cout << "Somme = " << sum;
    cout << "Moyenne = " << sum/2;
    return 0;
}
```

— Corriger les erreurs de ce programme.

b. Soit le programme ci-dessous :

Programme 2 C++ BTS SN

```cpp
// Programme pour calculer la distance entre deux points
#include <iostream>
#include <cmath>
using namespace std;

// Fonction pour calculer la distance
float distance(float x1, float y1, float x2, float y2)
{
    // Calcul de la distance
    return pow(y1 - x1, 2) + pow(y2 - x2, 2) ;
}

// Code principal
int main()
{
    float xA, yA, xB, yB;
    cout <<"********* Point A **********"<<endl;
    cout <<"Donner l'abscisse du point A : ";
    cin >> xA;
    cout <<"Donner l'ordonnée du point A : ";
    cin >> yA;
    cout <<"********* Point B **********"<<endl;
    cout <<"Donner l'abscisse du point B : ";
    cin >> xB;
    cout <<"Donner l'ordonnée du point B : ";
    cin >> yB;
    cout <<"La distance entre le point A et B vaut : " << distance(xA, yA, xB, yB)<<endl;
    return 0;
}
```

— Le programme contient-il des erreurs ?
— Si oui, corrigez les erreurs.

4 Programmes Arduino

4.1 Mesurer le courant qui traverse une LED

On veut mesurer le courant qui traverse la LED à l'aide de la carte `Arduino UNO` .
Le programme pour effectuer cette mesure est donné ci-dessous :

Programme Arduino : Mesurer un courant — BTS SN

```
// Entrée analogique utilisée pour mesurer le courant
#define entreeAnalogique 2

void setup()
{
   // Initialisation de la connexion série avec le moniteur
   Serial.begin(115200) ;
}

void loop()
{
   // Lecture de la valeur analogique à mesurer
   int valeurLue = analogRead(entreeAnalogique) ;
   // Conversion de la valeur lue en une tension en centi-Volts
   float tensionLue = map(valeurLue, 0, 511, 0, 500);
   // Envoi pour affichage sur le moniteur série de la tension mesurée
   Serial.print("Tension : ") ;
   Serial.print(tensionLue / 50.0) ; // Afficher la valeur en Volts
   Serial.println("Volts") ;
   // Attente d'une seconde (1000 ms) entre deux mesures et affichages
   delay(1000) ;
}
```

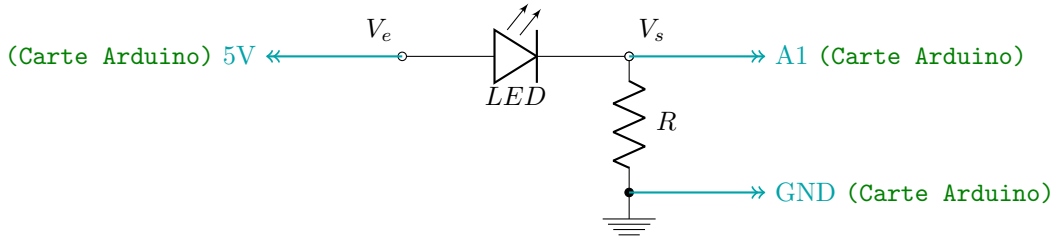

FIGURE 1 – Mesure de courant

Avec $R = 330\ \Omega$

Vérifier et corriger le programme pour mesurer le courant à travers la LED.

4.2 Mesurer le temps ...

Réaliser le montage ci-dessous :

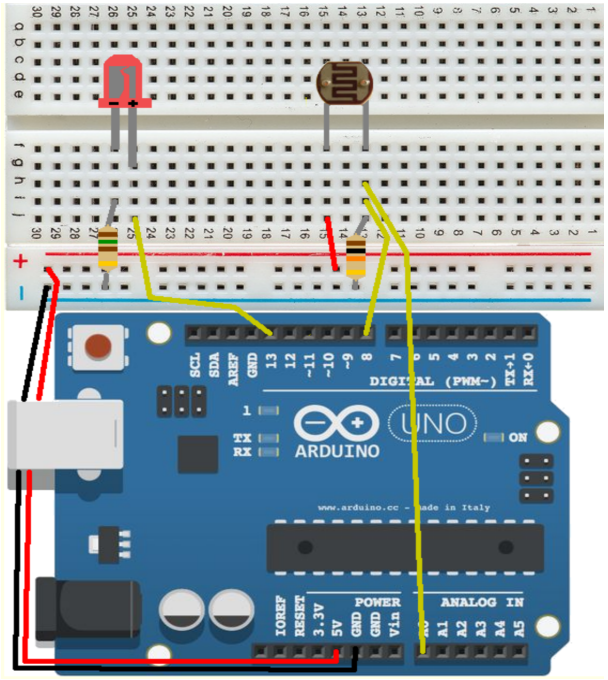

FIGURE 2 – Mesure du temps

Programme 2 Arduino — BTS SN

```
int nSensorValue = 0;    // Valeur lue sur le port analogique A0
int nDigitalValue = 0;   // Valeur lue sur l'entrée digitale 8

void setup() {
pinMode(13, OUTPUT); // Initialise la PIN digital 13 comme OUTPUT
pinMode( 8, INPUT);  // Initialise la PIN digital  8 comme INPUT
digitalWrite(13, LOW);
Serial.begin(115200);
} // setup

void loop() {
nSensorValue = analogRead(A0); // Lecture de la tension sur le port analogique A0

nDigitalValue = digitalRead( 8); // Lecture de l'état de l'entrée de la PIN 8

// La LED indique l'état digital lu.
if (nDigitalValue > 0) digitalWrite(13, HIGH);
else                   digitalWrite(13, LOW);

Serial.print("A0 = ");
Serial.print(nSensorValue);
Serial.print("   PIN8 = ");
Serial.println(nDigitalValue); // Ici, l'affichage passe à la ligne
delay(500);
} // loop
```

Quelles mesures sont réalisées par ce programme ?