簡単なデバッグ

1. デバッグ

   デバッグとは，プログラムのバグを見つけ，それを修正することである。デバッグの方法にもいろいろあり，主に次の 3 つが考えられる。

   ・ソースコードを自分で読んで，どこが間違っているのか修正する

   ・printf をコード中に埋め込んで，プログラムの挙動を把握する

   ・デバッガを利用する。

   以下，それぞれ軽く説明する。

2. 間違いを記憶しておく

   ソースコードを自分で読んで，どこが間違っているのかを修正することは，プログラムが複雑になればなるほど難しくなる。しかし，コンパイルエラーの場合や，printf デバッグやデバッガを必要としないようなわかりきったミスなら素早く発見できる。

   特に，コンパイルエラーの場合は，エラーが起こっている行が出力されるので，それを参考に修正すべきである。

   ある程度「どこがミスしやすい箇所なのか」を把握しておくのが良い。以下，参考までに間違えやすいような点を挙げておく。

   - セミコロンとコロンの書き間違い(キーの場所が近い)
   - カンマとピリオドの書き間違い(キーの場所が近い)
   - 変数の初期化忘れ。広義の意味での初期化(宣言と同時に代入，という初期化だけでなく，ループのたびに変数に 0 を代入など)
   - 配列の添え字によるミス
   - 不等号に=をつけるか否か
   - ループの継続条件
   - 再帰の return のタイミング

   もちろんこれ以外にも，バグを起こしやすいようなケースはたくさんある。それはたくさんのプログラムを組んでいれば分かってくることである。「こういう処理はバグりやすい」「ここは間違えやすい」というパターンをある程度記憶しておくのがよい。

3. printf デバッグ

   printf で変数や文字列を手軽に出力できる。これを用いて，かなり柔軟なデバッグが可能である。後述するが，このデバッグはあまり使うべきでないという人もいる。

   以下，いくつか例を挙げる。

   - プログラムが強制終了してしまったケースを考えてみよう。この場合，どの処理ま

でが上手く行っているのかを把握する他ために，printf をコードに埋め込むことがある printf には適当な文字列を出力させる。

- プログラムが意図しない動作を起こしているとき，その処理に関わっている変数の値がどうなっているのかを見たくなる。そこで，printf を用いてある地点での変数の値を出力させる。
- プログラムが無限ループに陥ってしまったとする。そんなとき，ループに抜けるための条件に関わる変数の値がループ内でどうなっているのかを確認するために，printf を用いる。
- プログラムが if 文内の処理をしているかどうかを確認するために，printf を if 文中に埋め込む。
- 配列の値がどうなっているのかを見たいときに，その値を for ループや printf を用いて出力させる。1 次元・2 次元配列の出力をすることがほとんどである。

printf を用いれば，知りたい情報をプログラム中から簡単に得られる。しかし欠点は，ソースコードが汚くなることである。特に競プロの場合は，デバッグ用の printf は提出するときに消さなければならない(うまく ifdef などを利用してマクロを書くと，実はこの問題は解決できるのだが，詳しくは割愛)。
また，デバッグのためにわざわざ新たな変数を導入することがあるのだが，こうなると元のプログラムの処理とデバッグの処理が癒着してしまい，後でデバッグ部分を取り除くのが面倒になる。
もしデバッガが使いこなせるならその方が便利だし安全だと思われる。

4. デバッガ

デバッガとは，デバッグを支援するためのプログラムである。
プログラムを 1 文ずづ実行してくれたり，ある地点で動作を一時停止することができたり，ある地点での変数の内容を見たりすることができる。
デバッガという独立したプログラムではないのだが，Visual Studio にはデバッグ機能が付属している。
GNU が提供しているデバッガに GDB(GNU Debugger)がある。MinGW で GCC をインストールした際に，付属としてインストールされているかもしれない。
デバッガは使いこなせれば非常に強力である。