# １１年度C言語講座

2011/06/09　第5回
講師：フェイス、ork、蟹男

# 見易いソースコードの書き方

- 結構重要な要素
  - 時折ソースコードを覗いてみると、頭を抱えたくなる構造になっていたりすることが。
  - 講師が発狂する原因になる以前に、バグの発生率があがってしまう。
  - ソースコードを見返してバグフィクスする時に自分が頭を抱えることになる。

# インデント

- まずは正しいインデントを。
  - インデントをつけるのにスペースを使うのは絶対にNG。
    - （　）←全角スペース
    - （　）←半角スペース二個

      →非常に分かりにくいバグの発生源。
  - インデントは、中括弧『{}』の中に入ったら一段。
    - 現状は問題ないが、多重ループなどのネスト（入れ子）構造が生じたときに、どの段に処理を入れればいいか等が判別しやすくなる。

# ソースコードの分割①

- 主にヘッダファイル（拡張子が.h）とソースファイル（拡張子が.c or .cpp）に大別される。
  - ヘッダファイル
    - 一般に構造体（来週にでも解説します）、関数のプロトタイプ宣言等を含む。
  - ソースファイル
    - 一般に関数の実部を含む。
- 大雑把な分割の指針
  - プログラムの前半の方に書くものをヘッダに
  - 関数をソースコードに

という形に分割する、と考えれば。

# ソースコードの分割②

- ソース及びヘッダファイルは1つのプログラムに複数個入れることが出来る。
  - 「プロジェクト」という概念を持つVSではソースコードのリンクは自動で行ってくれるが、コンパイラによっては手動でソースコードの関連付けを行う必要がある。
- 大雑把な分割の指針
  - １つの処理系統ごとに一対のソースとヘッダを用いる。
  - 余談だが、C++には『クラス』と『オブジェクト指向』という概念があり、比較的分割はしやすい。

# 関数化のコツ

- 繰り返し出てくる処理を関数化していくのが基本
  - コードに同じことを書く手間が省ける。
- 長くなった部分を関数化して分離
  - 例えば「キャラクタを動かす一連の命令」を
    move();という関数にまとめてしまう。
    コードの見た目上の複雑さが軽減され、
    さらにそこでどのような処理を行っているかが
    視覚的に見やすくなる。

    →適宜ソースコードの分割と合わせて。

# 課題

- ハイアンドローゲームを作成せよ。
  - プレイヤーから受け取るのは0～2
  - 初期は5，その後は1～9の数値をランダムに内部出力（乱数を使用）し、逐次次の数字が前の数字と比較（if）し、高いと予想すれば1を、低いと思えば2を、降りる場合は0を入力してもらう（scanf, printf）。これをプレイヤーが降りるか、予想が外れるまで繰り返す（ループ）。
  - 今日の講座を意識し、出来るだけ（少なくとも自分で）読みやすいコードを書くことに挑戦すること。
    プログラムの規模が小さいので、コードの分割については任意。