● ８回目・９回目の講義内容

【コンピュータの歴史】

- コンピュータ以前
  - アナログ計算機
  - 機械式計算機
- 電子計算機
  - リレー式計算機
  - ワンボード・コンピュータ

【コンピュータはなぜ動くのか】

★ ハードウェア

- ハードウェアの基本構成
- インターフェース

★ ソフトウェア

- オペレーティングシステムの役割
- 大きな量と小さな量の表現方法
- 日本語文字コード

【ネットワークはなぜつながるのか】

- TCP/IP と IP アドレス
- ルーティング
- プロトコル
  - DNS
  - SMTP
  - HTTP

【ネットワークセキュリティとは何か】

- パスワード
- セキュリティホールとは何か
- コンピュータウイルス
- メールの偽造
- WEB サーバの偽造
- パケットの盗聴
- 無線 LAN のセキュリティ
- 公開鍵暗号
  - メールの暗号化
  - PKI と SSL 暗号化

【その他】 電子メールで「今日の講義の感想や意見」を送ってください.

● 前回の講義のキーポイント及び補足

【関数プロトタイプ宣言の意味】

- 通常の考え方では, プログラムにあらわれる全てのオブジェクト（変数や関数）は**それが利用される以前に定義されていなければならない**. しかし, C言語における関数は, 定義以前にコード中に書かなければならない状況が存在する可能性がある. また, `printf` のような「ライブラリ関数」は, （いい加減な説明だが）実行時にその実体の検索が行われる. したがって, C言語においては, 関数を利用する以前に定義をすることは不可能である.
- そこで, C言語のコンパイラは, 定義されていない関数を見つけると, 次のような処理を行う.
  - 関数の実引数の個数と型をもとに, 仮引数の個数と型の推測を行う. もし, `foo(5,3.0)` として関数が利用されている場合には, `foo` は仮引数を２個とり, 第一引数は `int`, 第二引数は `double` であると推測する.
  - 関数の戻り値の型は `int` であると推測する. したがって, 上の例の `foo` は

    ```
    int foo(int, double)
    ```

    であると推測される.

- ところが, その関数が

  ```
  double foo(double, double)
  ```

  と定義されているとすると, コンパイラが推測したプロトタイプと異なった状況が発生する. これは, 関数が正しく動作しない原因となる.
- したがって, 関数プロトタイプ宣言を行わないとプログラムが正しく動作することを保証できない可能性がある.
- プログラムの冒頭にいつも記述している

  ```
  #include <stdio.h>
  ```

  とは, `printf` 関数などの「標準入出力関数」等のプロトタイプ宣言が記述されている, システムに付属する `stdio.h` という「ヘッダファイル」をプログラム中に取り込む命令（プリプロセッサマクロ）である. この他に “`double sin(double)`” のような「数学関数」を利用する場合には, 数学関数ヘッダ `math.h` を取り込む必要がある.

【変数のクラス】

- Cで利用される変数には, 次の２つの概念が付属している. （本当は他にも付属する概念がある.）
  - **★ 可視性 (scope)** そのオブジェクトがプログラム内のどの部分から「可視」であるかを示す概念. 「大域的」と「局所的」の２つに分けられる.
    - **▼ 大域的 (global)** 正しくは「ファイル大域的」と呼ばれる. ソースファイル内のどこからでも可視となっているオブジェクトを指す.
    - **▼ 局所的 (local)** 正しくは「ブロック局所的」と呼ばれる. そのオブジェクトが定義されている「ブロック」（中括弧でくくられた「複文」のこと）内部のみから可視となっているオブジェクトを指す. （関数仮引数定義にあらわれる変数も, その関数ブロック内に局所的である.）
  - **★ 寿命（存在期間）** そのオブジェクトがプログラムの実行期間中のどの期間で有効であるかを示す概念. 「静的」と「自動的」の２つに分けられる.
    - **▼ 静的 (static)** プログラムの起動時から終了時までを通じてメモリ内に存在するオブジェクトを指す.
    - **▼ 自動的 (auto)** （「動的 (dynamic)」と理解してもよい.）プログラムの動作中の一部分の期間にしかメモリ内に存在しないオブジェクトを指す.
- **変数の例**
  - ソース内のどのブロックにも属さない場所で定義された変数は, 大域的な可視性と静的な存在期間を持つ. したがって, （その定義以降であれば）ソース内のどこからでもオブジェクトを参照可能である.
  - 関数ブロックの先頭で定義された変数は, 関数内部に局所的な可視性と, 関数実行時から関数終了時までの自動的な存在期間を持つ. 関数が複数回実行される場合, 毎回の実行ごとにそのオブジェクトは異なる実体を持つ. （なお, 自動変数（局所変数）を定義可能な場所は「ブロックの先頭, 実行文が記述される以前」とされている.）
  - これ以外に「局所的な可視性を持ち, 静的な存在期間を持つ」変数を定義することができる.

- **変数名の隠蔽**
  ある識別子名を持つオブジェクトが２つ以上存在した場合には, より小さな可視性を持つオブジェクトを参照する．つまり, より近い場所で定義されたものを見に行く．これを「**隠蔽**」と呼ぶ.

【関数への引数の渡し方】

- 関数実引数は, 呼び出し時には**その値が評価され, 値のみが関数に渡される**．これを「**値渡し**」, 正しくは「**値による呼び出し** (**call by value**)」と呼ぶ.
- 関数（サブルーチンまたは手続き）などへの引数の渡し方はプログラム言語によって異なる.
- 実際には, 関数呼び出し時に新しいメモリエリアに実引数の値がコピーされ, 呼び出された関数側からは, そのメモリエリアに対して（仮引数定義で定義された）オブジェクトの参照が行われる.
- したがって, 呼び出された関数側で実引数の値を変更しても呼び出し側に戻ったときには, その変更は反映されない.

● 前回の課題の解説

★ exercise-07-2 次の仕様をみたす関数をつくり，２つの正の整数の最大公約数を求めるプログラムを書きなさい.

【形式】

unsigned int gcd(unsigned int a, unsigned int b)

【機能説明】

a, b がともに 0 でない場合にはそれらの最大公約数を返します. a, b のいずれか一方が 0 の場合には 0 を返します.

```
/* $Id: exercise-07-2.c,v 1.4 2004-05-31 14:01:38+09 naito Exp $ */
/* ユークリッドの互除法によって最大公約数を求める */
#include <stdio.h>
unsigned int    gcd(unsigned int, unsigned int) ;

int main(int argc, char **argv)
{
  unsigned int  a=40920U, b = 24140U ;

  printf("gcd(%u,%u) = %u\n", a,b, gcd(a,b)) ;
  return 0 ;
}

unsigned int gcd(unsigned int a, unsigned int b)
{
  unsigned int  r ;
  if ((a == 0U)||(b == 0U)) return 0U ;
  while((r = a%b)) {
    a = b ; b = r ;
  }
  return b ;
}
```