ソフトウェア概論 B

数学科 吉開範章, 渡辺俊一 (栗野 俊一)

おしらせ

□本日は吉開先生は出張でおりません

○代わりに栗野が担当します。

□資料

http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino/2010/soft/20100618/20100618.html

本日の概要：関数 (前半)

- □ 関数 (6 章 : p.114　p.125) の前半
  - ○ 関数とは (6-1 : p.114)
  - ○ 関数の設計 (6-2 : p.122)
- □ 課題
  - ○ 6-2 (p.118) : 三つの int 型整数の最小値を返す関数
  - ○ 6-4 (p.121) : int 型整数の四乗値を返す関数
  - ○ 6-5 (p.123) : 警報を no 回だけ発する関数
  - ○ 6-6 (p.125) : 画面に「こんにちは。」を表示して改行を行う関数

関数とは (6-1 : p.114)

- □ 関数とは
  - ○ 定義:「幾つかの命令文の集まり」に名前(関数名)を付けた物
    - ▹ 関数定義 :「幾つかの命令文の集まり」に関数名を付けて関数が作成できる
    - ▹ 関数呼び出し : 関数名を使って「幾つかの命令文の集まり」が利用できる
  - ○ 例 1: main 関数 (cf. Fig. 6-1, p.114)
    - ▹ 「int main (void) {　} 」の部分は 「main 関数」だった
    - ▹ 今迄、知らない内に、関数を「作成(関数定義)」してきた
    - ▹ 「C 言語でプログラミング」==「main 関数の作成から始まる」
  - ○ 例 2: ライブラリ関数
    - ▹ printf, scanf, puts, putchar は「ライブラリ関数」だった
    - ▹ 今迄、知らない内に、関数を「利用(関数呼び出し)」してきた
- □ 目標 : 関数を「理解する」こと
  - ○ 自分で関数が作成できる事
    - ▹ 関数定義ができる
  - ○ (自分や他人が作成した) 関数が利用出来る事
    - ▹ 関数呼び出しができる

関数定義と関数呼び出し(具体例:maxof)

□ 関数定義と関数呼び出しの具体例

○ 目標 : 「二つの整数を受け取って、大きい方の値を返す関数」を作る

○ maxof という関数の定義と関数呼び出しの例 (List 6-1, p.115)

▹ 二つの関数 (maxof と main) からなるプログラムファイル

▹ maxof : 新しい関数を定義している (今回の焦点)

▹ main : maxof を利用している (これまでと変らない)

○ 関数定義の構造 (Fig. 6-2, p.115)

▹ <関数定義> ::= <関数頭部> <関数本体>

▹ <関数頭部> ::= <返却値型> <関数名> <仮引数の宣言>

int maxof (int x, int y)

▹ <返却値型> ::= 「この関数が返す値の型」: maxof は int 型の値を返す

▹ <関数名> ::= 「この関数の名前」: 新しく定義される関数の名前は maxof である

▹ <仮引数の宣言> ::= 「関数引数の名前と型宣言」: maxof は int 型の二つの引数を取る

▹ <関数本体> ::= 「命令の集まり」: 関数 maxof の本体は「{　}」の部分で、二つの数の大きい方を求める

○ 関数呼び出し式 (Fig. 6-3, p.115)

▹ <関数呼び出し式> ::= <関数名> '(' <実引数並び> ')'

maxof (na, nb);

▹ <実引数並び> ::= 「関数に与える引数の並び」: na と nb の二つの式を引数として与える

▹ <関数呼び出し演算子> ::= '(' と ')' の事

関数呼び出し：実引数と仮引数の関係

- □ 関数呼び出し
  - ○ 定義済の関数に引数を指定して呼出す (Fig. 6-4, p.116)
  - ○ 関数定義の本体部分が実行される
    - ▹ 実引数部分の式の値が、仮引数の変数に代入されているように振舞う(値渡し)
    - ▹ return 分の式の値が、関数の値として返ってくる
    - ▹ 関数と関数の呼び出し側の間では、「値のやり取り」しかできない
  - ○ 「return 文」の二つの役割
    - ▹ 関数の値を定める
    - ▹ その時点で、関数の実行を終了する
- □ 関数内の変数
  - ○ 関数の中でも自動変数が宣言できる (仮引数も自動変数)
    - ▹ 関数内で宣言された自動変数(名)は、他の関数(例えば呼出し元)とは無関係
    - ▹ 「同じ名前」でも「異なる物」として扱われる
    - ▹ cf. 大域変数

関数利用の例

- 関数利用の例
  - 三値の最大値 (List 6-2, p.118)
  - 二乗の差 (List 6-3, p.119)
  - べき乗 (List 6-4, p.120)

関数の設計

- □ 関数の設計 (6-2 : p.122)
  - ○ void 型 : 値を返さない関数 (6-7 : p.122)
    - ▹ 「手続き」としての関数
  - ○ 引数のない関数 (6-9 : p.124)
    - ▹ 引数の void 宣言
- □ スコープ: 名前の有効範囲
  - ○ 自動変数の有効範囲は、ブロック内(複合文内)のみ
    - ▹ (特に) 関数間では独立
    - ▹ 引数も実は関数ブロック内の自動変数
- □ 値渡し
  - ○ 実引数から仮引数への情報を伝達の仕方の一つ
    - ▹ C 言語では「値渡し」になる
    - ▹ 実引数の式の「値」が仮引数の自動変数へ「代入」される
    - ▹ 仮引数で宣言された変数の値を変更しても、呼び出し側には影響しない
  - ○ 値渡しの効用
    - ▹ 関数内での出来事が、呼び出し側に影響しない(副作用がない)
    - ▹ 呼び出し側から関数 : 引数を利用する (幾つでも可能)
    - ▹ 関数から呼び出し側 : 返値を利用する (一つだけしかできない)

関数利用の効用

- □ 関数利用の効用
  - ○ 問題が分割される(分割統治)
    - ▷ 関数を利用するまで : main 関数一つで、全ての事を行う
    - ▷ 関数を利用すると : 作業分担させれば、個別に考える事ができる (cf. 変数名の管理)
  - ○ 「分割する」だけで「問題が易しく」なる
    - ▷ 一度に把握すべき内容が少い(個々の関数は小さいので把握し易い)
    - ▷ 易しくなるような分割を考える必要がある
  - ○ プログラムの一部に名前を付ける事ができる
    - ▷ (意味のある)名前を使う事により読み易くなる
  - ○ プログラムの作成順序が自由になる
    - ▷ main (全体像)から作っても、関数(詳細像)から作ってもよい
    - ▷ 「関数」を利用して、「問題を棚上げ」できる(後から作ればよい)
    - ▷ 作業を分担してもよい (他人数で作れる)
  - ○ 再利用できる
    - ▷ 一度定義した関数は、何度でも利用できる (cf. ライブラリ関数)
    - ▷ 生産性が上る (関数部分はもう作らなくてもよい)
    - ▷ 正確性が上る (関数部分は正しい事が判っている)

関数の抜き出し方

- □ リファクタリング
  - ○ プログラムを整理して、判り易くする
    - ▹ 関数を抜き出す
    - ▹ 名前を付けなおす
    - ▹ 手順を整理する
    - ▹ 無駄を省く
  - ○ プログラムの内容そのものは書き換えない
  - ○ プログラムを理解り易くし、今後の変更に備えるための手段
- □ 関数の抜き出し方
  - ○ 関数として抜き出すプログラム部分を選ぶ
  - ○ その部分をブロックとして取り出し、名前を付ける
  - ○ 引数の設定
    - ▹ 実引数 : 関数に渡す値
    - ▹ 仮引数 : 関数で利用する変数
  - ○ 帰り値の調整
    - ▹ 関数の返値の宣言
    - ▹ return 文

課題

□ 課題は、次の Web Page の内容を参照してください。

http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino/2010/soft/20100618/20100618.html

- ○ 6-2 (p.118) : 三つの int 型整数の最小値を返す関数
- ○ 6-4 (p.121) : int 型整数の四乗値を返す関数
- ○ 6-5 (p.123) : 警報を no 回だけ発する関数
- ○ 6-6 (p.125) : 画面に「こんにちは。」を表示して改行を行う関数

おわり

終了