関数の役割と作成方法

山本昌志*

2006 年 10 月 20 日

概 要

C 言語の関数を使ってプログラム例を示し，関数の役割を説明する．そして，基礎的な関数の記述方法を示す．これらを理解した後，練習問題と課題により関数を使えるようにする．

1 本日の授業内容

本日の学習範囲は教科書 [1] の p.162–179 である．教科書とは別に，プリントに沿って関数の内容—役割と記述方法—を説明する．本日の学習のゴールは以下のとおりである．

- 関数とな何か?—が分かる．そして，関数は便利そうだ—ということを実感する．
- 基本的な関数の記述方法が理解でき，プログラムに応用できる．

プリントを使って説明するが，教科書は何回も読み直せ．同じ内容も，別な観点から説明を受けるとよくわかる．プリントと教科書，両方とも重要である．

2 前回の復習

前回は「制御の流れ」の最後で，`break` と `continue`，`goto` の学習を行った．それぞれの内容は，次のとおりである．

ループからの脱出 ループ処理中に `break` に出会うと，その瞬間に継続条件式とは無関係にループから脱出する．通常は `if` 文を伴って使うことが多い．図 1 にフローチャートを示す．

*独立行政法人秋田工業高等専門学校電気工学科

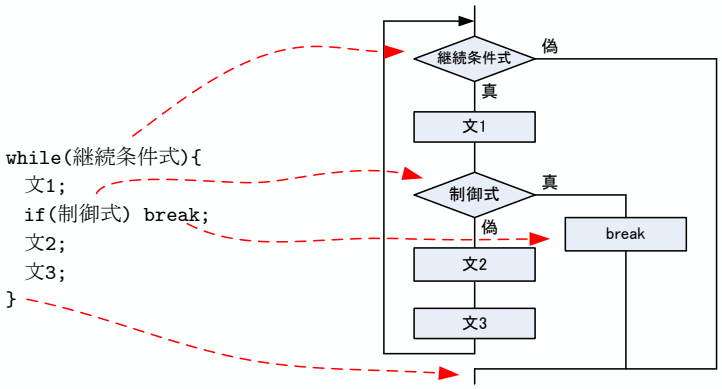

図 1: break 文を使って，ループからの脱出する構文

ループのスキップ　ループ処理中に continue に出会うと，それ以降のループブロックの処理がスキップされる．ただし，継続条件式が正しい限り，ループ処理は続けられる．通常は if 文を伴って使うことが多い．図 2 にフローチャートを示す．

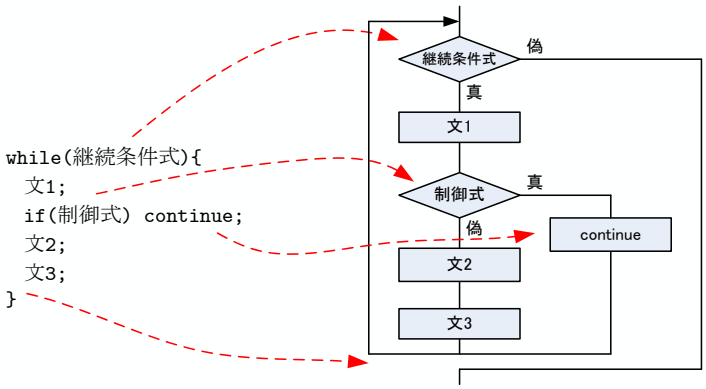

図 2: contine を使って，ループブロックをスキップする構文

無条件分岐　goto 文により強制的にプログラムの制御を移すことができる．この文に出会うと，goto 文が示すラベルに実行が移る．if 文と共に用いられることが多い．もちろん，単独で使用することも可能である．ただし，goto 文はプログラムの流れがわかりにくくなるので，使わないほうが良いとされている．しかし，二重以上のループから一気に脱出するときには有効である．break 文で脱出できるのはひとつのルー

プのみである．

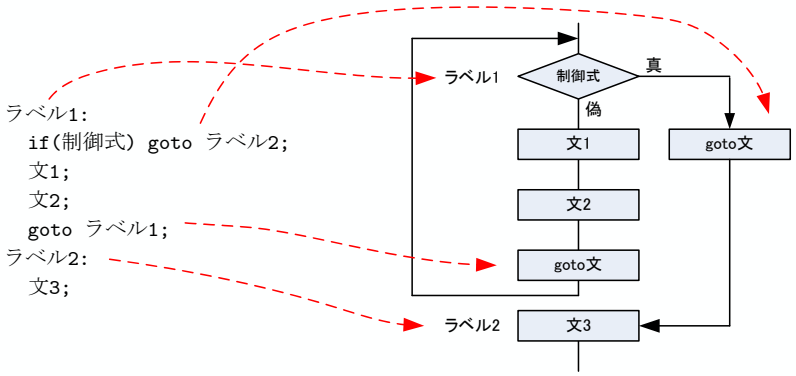

図 3: goto 文とラベルによる制御

3　関数

3.1　関数とは何か?

C 言語の関数 (function) とは，ある特定の処理を行うプログラムのことである．数学にも関数という単語を使うが，それと似ている場合もあるし，異なる場合もある．それぞれの場合について，例を使って説明する．

3.1.1　数学の関数と似ている C 言語の関数

最初は，数学関数を C 言語の関数を使って計算する．これまで，さんざん作成してきた次の関数

$$f(x) = -x^2 + 10 * x + 8 + 10 \sin^2 x \tag{1}$$

の最大値を求めるプログラムに C 言語のユーザー定義関数を適用する．この数学の関数を計算するために，C 言語のユーザー定義関数 `f()` を使うのである．リスト 1 にプログラムを示す．このプログラムは，次のようになっている．

- 4 行目はプロトタイプ宣言と呼ばれるものである．今のところ，プログラムの処理手続き—アルゴリズム—には関係ないので，気にしないことにする．プロトタイプ宣言については，後で説明する．
- 9–38 行目がメイン関数である．ここで，最大値を求めている．
  - 最大値を求める方法はこれまでと同じ．ただし，数学関数は，`f(xmin)` や `f(x)` で計算しており，分かりやすくなっている．

- 43–49 行目がユーザー定義関数である．ここで，式 (1) を計算している．図 4 にしたがい，このユーザー定義関数の構造を示す．
  - 戻り値の型は，計算結果の型を示す．式 (1) の計算結果は倍精度実数型なので，`double` としている．
  - 関数名は `f` である．関数名を指定することにより，ユーザー定義型関数の計算ができる．
  - 仮引数とは，呼出元—ここではメイン関数—から渡される計算に必要なデータを格納する変数である．
  - 44 行目で計算に必要な変数 `y` を用意して，46 行目で計算を行っている．
  - 48 行目で変数 `y` に格納されている値を呼出元へ渡している．`y` の値は，数学関数である式 (1) の計算結果となる．

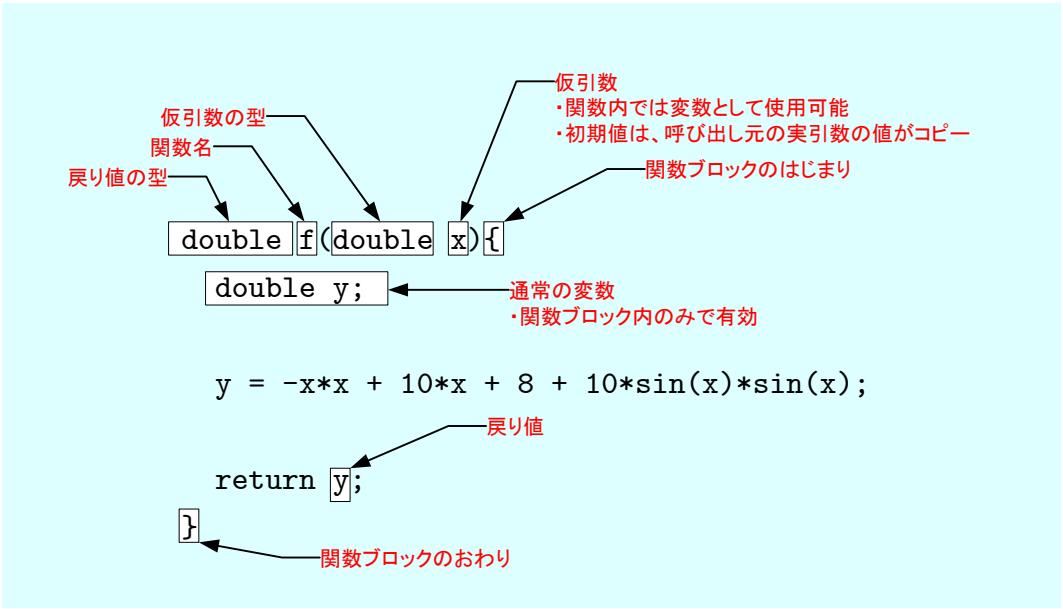

図 4: リスト 1 の関数の説明

リスト 1: 数学関数の最大値を求めるプログラム

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

double f(double x);             //プロトタイプ宣言

//=========================================================
// メイン関数
//=========================================================
int main(void){
  double x, dx, xmin, xmax, y;
  double max_y, max_x;
  int i, ncal;

  //--- 計算条件設定 ---
  xmax = 10;
  xmin = -10;
  dx = 0.0001;
  ncal = (xmax-xmin)/dx;

  //--- 暫定最大値 ---
  max_x = xmin;
  max_y = f(xmin);

  //--- 最大値検索 ---
  for(i=1; i<=ncal; i++){
    x = xmin + i*dx;
    y = f(x);
    if(max_y <= y){          //最大値が見つかった場合
      max_x = x;
      max_y = y;
    }
  }

  printf("%fのとき, 最大%fとなる. \n",max_x, max_y);

  return 0;

}

//=============================================================
// ユーザー定義関数
//=============================================================
double f(double x){
  double y;

  y = -x*x + 10*x + 8 + 10*sin(x)*sin(x);          // 関数の計算

  return y;
}
```

3.1.2　手続きの集まりとしての C 言語の関数

C 言語の関数は**処理の手続きをまとめたもの**—である．先に示した例は数学の関数を C 言語の関数を使って計算しており，両者は同一の形になっている．しかし，このように数学の関数を表現するために，C 言語の関数を使う例は特殊である．C 言語の関数はある特定の機能をはたす手続きをまとめたもの—となっているのが普通である．

リスト 2 の 27–47 行に記述している関数は，公約数に関する手続きをまとめている．この関数の機能は，(1)2 つの整数の公約数を表示，(2) 公約数の個数を返す—というものである．この関数の動作を見てみよう．

1. 2 つの整数を指定して，このプログラムを呼び出す (15 行)．2 つの整数は，変数 m と n にコピーされる．
2. 整数 m と n の大小関係を比較して，小さい方の値を変数 small に格納する．
3. 「m と n の公約数は，以下の通りです.」と表示する．
4. 1〜small までの整数が公約数になっているか—調べる．
5. 公約数が見つかれば，表示する．
6. 呼出元へ，公約数の個数を返す．

メイン関数では，2 個の整数をキーボードより読み込み，公約数を調べる関数 cmndiv を呼び出す．そして，公約数の個数を表示する．プログラムの実行結果は，以下のようになる．

実行結果

```
2 つの整数を入力してください．公約数を表示します.
84
48
84 と 48 の公約数は，以下の通りです.
1
2
3
4
6
12
公約数の数は，6 個です.
```

リスト 2: 公約数を求めるプログラム

```
#include <stdio.h>

int cmndiv(int m, int n);       //プロトタイプ宣言

//=========================================================
// メイン関数
//=========================================================
int main(void){
  int hoge, fuga, num;

  printf("2つの整数を入力してください. 公約数を表示します. \n");
  scanf("%d", &hoge);
  scanf("%d", &fuga);

  num = cmndiv(hoge, fuga);

  printf("公約数の数は, %d個です. \n", num);

  return 0;
}


//=============================================================
// ユーザー定義関数
// 公約数(common diviser)を捜す
//=============================================================
int cmndiv(int m, int n){
  int i, small, count=0;

  if(m<n){
    small = m;
  }else{
    small = n;
  }

  printf("%dと%dの公約数は, 以下の通りです. \n",m,n);

  for(i=1; i<=small; i++){
    if(m%i == 0 && n%i ==0){
      count++;
      printf("%d\n", i);
    }
  }

  return count;

}
```

3.2 関数の役割

これまで，2 つの関数を使ったプログラムの例を示した．この例で示したプログラムでは，関数を使わないでも記述できる．それでは，なぜプログラムの記述に関数が必要なのであろうか?．**長いプログラムを効率よく記述する**ために関数を使う—というのが答えである．

そのために，コンピュータープログラムの関数には 2 つの役割がある．一つ目の役割は，同じような処理を一つにまとめることである．実際のプログラムの動作は，同じ処理，あるいは似たような処理が非常に多い．いちいちそれを書くとプログラムが長くなり，プログラマーは大変である．そこで，ひとつにまとめ，必要なときに呼び出す．

分かりやすいプログラムの記述—が関数の二つ目の役割である．長いプログラムになると，処理の内容がわかり難くなる．例えば，ソースプログラムが 4000 万行だと言われている Windows 2000 について，それぞれの実行文の役割など分からない．コンピューターは大量のトランジスターからできているが，それぞれの役割が分からないのと同じである．このように大量の部品 (実行文) から構成されるコンピューター (プログラム) の動作を考える際に重要なことは，モジュール[1]に分解することである．モジュールとは，ある機能を果たす部品のことである．複雑な機械も，モジュールに分割すると動作の内容が分かるようになる．長いプログラムを作る場合も同じで，モジュールに分け，分かりやすくすることが重要である．プログラムをモジュール—機能単位—に分割し，プログラムの動作内容をわかりやすくするために，C 言語では関数を使う．

まとめると，関数の役割は

- 何度も同じことを書くのは面倒なので，処理を一つにまとめる．
- プログラムの内容を分かりやすくするために，処理を機能単位に分割する．

である．特に，2 番目が重要で,「プログラムのソースは分かりやすくなくてはならない」ということを，肝に銘じておかなくてはならない．

3.3 関数の作り方と使い方

関数をつくり，使うためには，ソースプログラムを図 5 のように記述する．必要な記述は，

- **プロトタイプ宣言**．関数の入出力の仕様を示す．
- **関数の定義**．関数での処理の内容を示す．
- **関数のコール (Call:呼び出し)**．関数を動作させる．

3.3.1 プロトタイプ宣言

プロトタイプ宣言はコンパイラー[2]に関数の引数の型と個数，それから戻り値の型を知らせる役割がある．これにより，コンパイラーがソースプログラム中で関数の使い方の間違いをチェックする．これは，プログラマーにとって，非常にありがたい機能である．

[1] module:単位
[2] 諸君が書いた人間に分かる C 言語をコンピューターに分かる機械語に直すプログラム

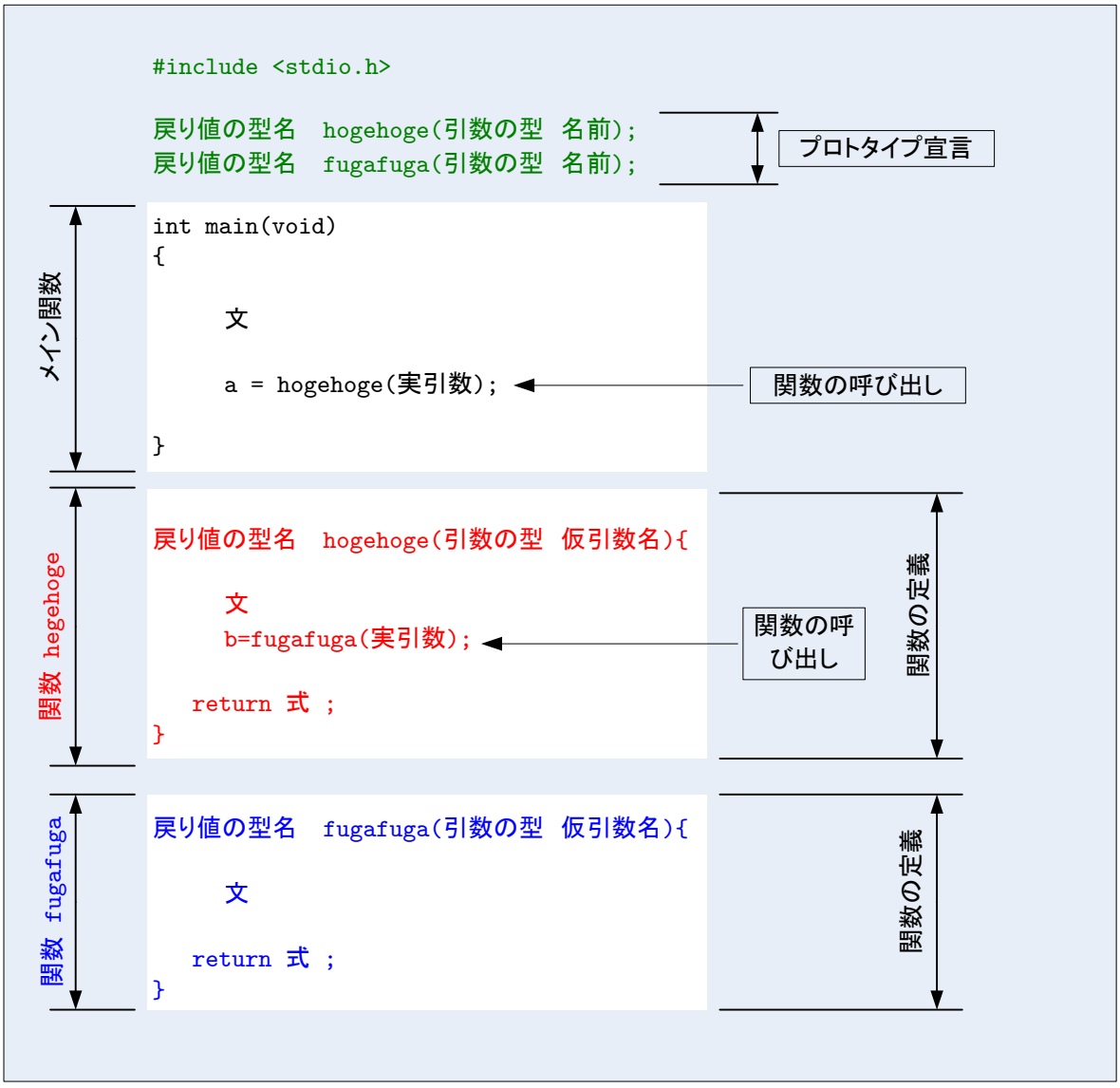

図 5: ユーザー定義関数を含んだソースプログラムの書き方

プロトタイプ宣言は，図 5 のように関数の定義より前に記述しなくてはならない．実際には，コールよりも前に関数の定義を書けば，このプロトタイプ宣言を省くことは可能である (教科書 [1] のリスト 5.3)．しかし，それは良くないスタイルとされている．このプロトタイプ記述は簡単で，関数の定義の先頭部分をコピーして，セミコロンをつければ良い．

プロトタイプ宣言を書くことにより，ソースプログラムを読みやすくなる．現在では，複数のプログラマーによりひとつのプログラムが作成されるため，読みやすいあるいは分かり易いプログラムを書くことは重要である．

プロトタイプ宣言をまとめると

- 書式は，戻り値と関数名，それから引数を順番に書く．
- 関数が正しく記述されているか，コンパイラーが文法をチェックするために使う．

となる．

3.3.2 関数の定義

プログラマーが関数に要求する動作の内容の記述を，ここでは関数の定義と言う．動作といっても，(1) 引数を受け取り，(2) それを処理して，(3) その結果を呼び出し元へ返す— という一連の処理の内容をプログラムソースに記述するだけである．

関数の定義をまとめると，次のようになる．

- メイン関数と同様，処理内容を書けば良い．
- 呼び出し元からのデータ (実引数) を関数の仮引数にコピーすることで，動作に必要なデータが渡される．
- `return` 文で呼び出し元へ，データを返す．`return 式;`—と記述する．式は，変数だけでも良い．

3.3.3 関数のコール

実際に関数を使う場合，それを使いたい場所で，引数を伴って関数名を書く．関数を使う動作を「関数のコール」，あるいは「関数の呼出し」と言う．関数をコールは `main` 関数のみならず，他の関数からも可能である．また，自分自身の関数からもコールできる (再帰呼び出し)．再帰呼び出しについては，2 年生で学習する．

- 関数を呼び出すためには，引数を伴って関数名をコールするだけである．
- どこからでも，何回もコールすることができる．

4　プログラム作成の練習

[**練習 1**]　数学関数

$$f(x) = \frac{1}{x^2+8} - 2x^2 + 4x - 20\cos x \qquad -100 \leqq x \leqq 100 \tag{2}$$

の最大値を計算するプログラムを作成せよ．数学関数の計算には，C 言語のユーザー定義関数を使うこと．計算する x の精度は 0.0001 とする．

[**練習 2**]　3 つの整数をキーボードから読み込み，公約数とその個数を表示するプログラムを作成せよ．C 言語のユーザー定義関数を使うこと．

5　課題

次回の講義の日 (10 月 25 日) の AM8:45 までに，以下の課題をレポートとして提出すること．表紙等は，いつもの通り．表紙のタイトルは「関数の役割と作成方法」とすること．

[**問 1**]　(復) 教科書の p.162–179 を 3 回読め．重要なところには，赤線あるいは蛍光ペンで印を付けよ．不明な点があれば，クラスの者に聞け．それでも分からない場合は，Office hours を利用して私 (山本) に質問しろ．

[**問 2**]　本日，配布したプリントを 3 回読め．

[**問 3**]　(復) 数学関数

$$f(x) = \frac{1}{x^2+5} + x^2 + 10x - 8\cos x \qquad -100 \leqq x \leqq 100 \tag{3}$$

の最小値を計算するプログラムを作成せよ．数学関数の計算には，C 言語のユーザー定義関数を使うこと．計算する x の精度は 0.0001 とする．の最大値を計算するプログラムを作成せよ．

[**問 4**]　(復)2 つの整数をキーボードから読み込み，公約数の個数と最大公約数を表示するプログラムを作成せよ．C 言語のユーザー定義関数を使うこと．

[**問 5**]　(復) プロトタイプ宣言が必要な理由を説明せよ．

[**問 6**]　(予) 教科書の p.180–188 を 3 回読め．

参考文献

[1] 内田智史監修, (株) システム計画研究所編. C 言語によるプログラミング　基礎編　第 2 版. (株) オーム社, 2006.