# C 言語の学習　関数

山本昌志*

2007 年 5 月 29 日

**概 要**

関数について学習する．はじめ，関数とはなにか—ということを説明する．C 言語は関数からできていることと，関数を使う理由を述べる．その後，関数の書き方を学習する．引数の渡し方はいろいろあるので，教科書と併用して学習を進めることにする．

## 1　本日の学習内容

本日の内容は，教科書の 11 章である．以下のことをよく理解して欲しい．

- 関数を使う理由が分かる．
- 関数の書き方が分かる．
- データの受け渡しが分かる．

## 2　関数とは何か

### 2.1　`main()` 関数と関数の構造

1 年生の時に学習した FORTRAN を覚えている人は，サブルーチンを思い出して欲しい．それが，C 言語の関数に相当する．いかなるプログラミング言語でも，このサブルーチンに相当する機能がある．それだけ便利な機能であるということである．

これまでの学習で，諸君は知らない間に関数を使っていた．おまじないと言っていたものの一部は `main()` 関数と呼ばれるものである．いままでのプログラムを思い出してほしい．必ず，`main` とか言うものを書いていたはずである．

`main()` 関数にかぎらず，これから学習する関数は同じような構造である．`main()` 関数の構造を図 1 に示す．それぞれの役割は，以下のとおりである．

- **返り値の型**　関数での処理が終わった後，`return` 文により値を返す．これを戻り値と言い，型を指定しなくてはならない．
- **関数名**　関数を呼び出すときに使う．関数名を書くことにより，関数の処理を呼び出すことができる．

*独立行政法人　秋田工業高等専門学校　電気工学科

- **引数の型と変数名**　関数でデータを処理するときに与える変数．場合によっては，呼出元と呼び出された関数でメモリーを共有して，データの交換に使うこともできる[1]．
- **返り値**　return 文に引き続いた変数，あるいは値を呼出元へ返す．

この main() 関数の例でも分かるように，関数での処理の内容はブロック—中括弧 { } で囲まれた部分—中に書かれなくてはならない．今まで，学習してきたプログラムでは，main 関数のみがあり，その中に実行文が書かれていたはずである．プログラムでは，このブロックを処理のひとかたまりと考える．

main 関数は特別で，C 言語のプログラムには，必ず 1 個必要で，そこから実行されることになっている．

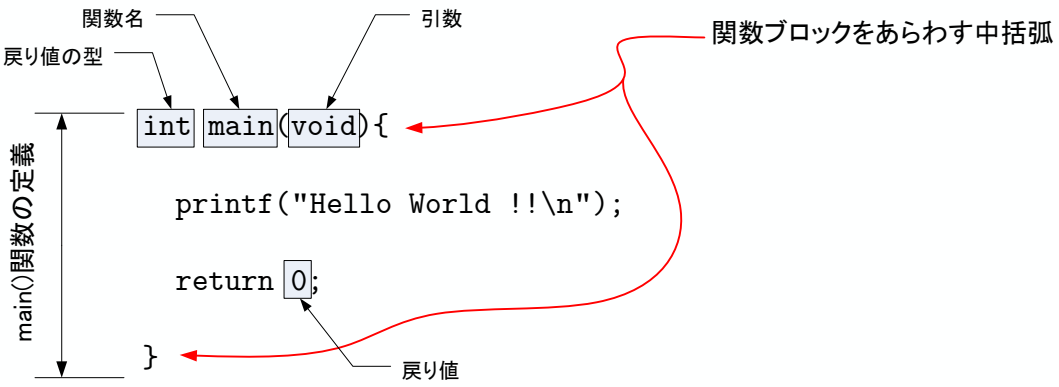


図 1: main() 関数の構造

## 2.2　関数の例

実際の C 言語のプログラムは，main 関数のみからできていることは希で，リスト 1 のように複数の関数からできている．C 言語のプログラムは関数の集合から構成され，それらが順番に呼び出されて処理を行う．リスト 1 を例に関数の動作を考えよう．

このプログラムでは，main 関数と bigger と言う関数が使われている．main 関数の構造については，先ほど述べたとおりである．bigger の方は，次のようになっている．

- **戻り値の型**　int と整数を返す関数である．33 行目を見ると，return 文で整数の big を返している．
- **関数名**　bigger となっている．ここでは，main 関数で，関数名 (bigger) を引数を伴って，呼び出すことにより，処理がはじまる．
- **引数**　main 関数から処理に必要な値—x と y— がわたされ，それが変数 a と b に格納される．変数の型は倍精度実数型である．
- **返り値**　変数 big に格納されている値を返す．

[1] 後で学習する参照渡し

このリスト 1 の動作の内容を，図 2 のフローチャートに示す．動作は単純なので，すぐに理解できるであろう．関数を使って，大きい方の値を調べ，表示している．

数学の関数とほとんど同じような動作をしていることが分かるだろう．数学では変数を与えて，関数値が求まる．C 言語では，変数に代わり引数が与えられ，戻り値がも求まるわけである．

リスト 1: 関数を使ったプログラム例．大きい方を調べる．

```
1  #include <stdio.h>
2
3  double bigger(double a, double b);    /* プロトタイプ宣言 */
4
5  /*======================================================================*/
6  /*      メイン関数                                                      */
7  /*======================================================================*/
8  int main(void){
9    double x, y, z;
10
11   x=2.5;
12   y=3.1415;
13
14   z=bigger(x, y);
15
16   printf("大きい方は、%fです。\n", z);
17
18   return 0;
19 }
20
21 /*======================================================================*/
22 /*      大きい方を探す関数                                              */
23 /*======================================================================*/
24 double bigger(double a, double b){
25   double big;
26
27   if(a<b){
28     big = b;
29   }else{
30     big = a;
31   }
32
33   return big;
34
35 }
```

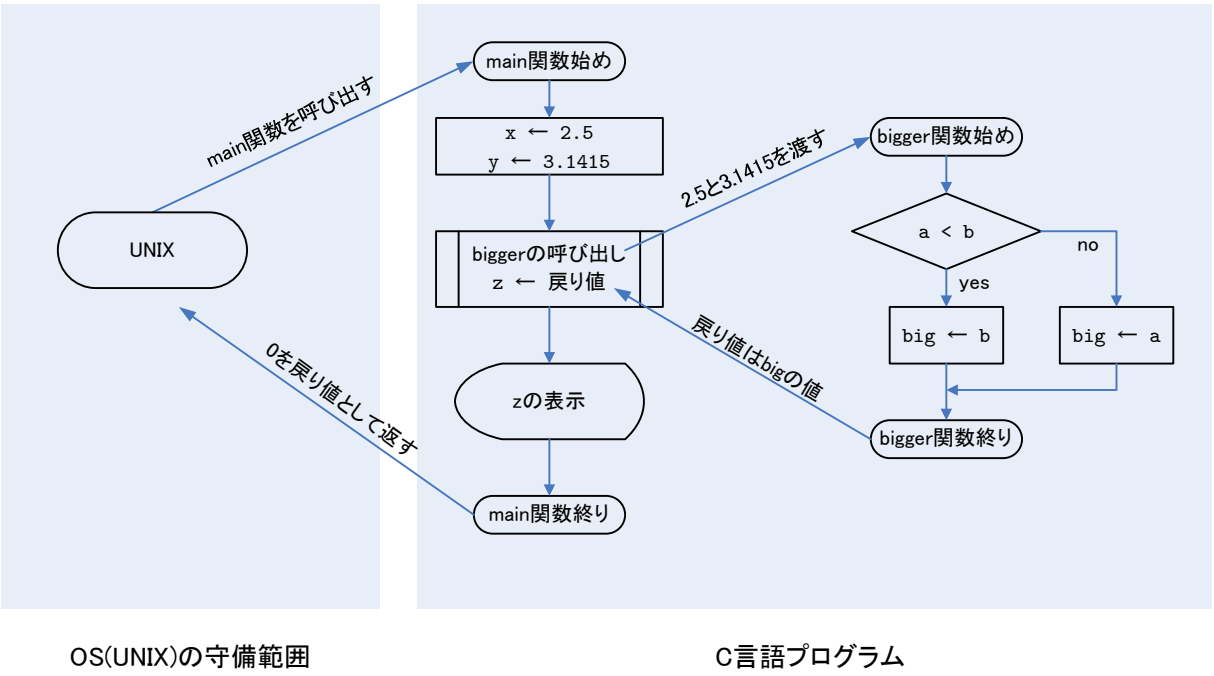


図 2: リスト 1 の動作フローチャート

## 2.3 関数の役割

関数は，長いプログラムを効率よく記述するために必要である．そのために，この関数には 2 つの役割がある．一つは，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，一つにまとめ，必要なときに呼び出すのである．

もう一つ，長いプログラムの問題は，処理が分かりにくい点である．例えば，windows2000 だとソースプログラムは大体 4000 万行だと言われている．この場合，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[2]に分解することである．そうすると，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，機能単位 (モジュール) に分け，分かりやすくすることが重要である．C 言語では関数を使い，機能単位にプログラムを分割する．

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，同じような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．

[2]module:単位．機器の場合，ある機能を果たす部品単位を表す

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．

### 2.3.1 同じ処理をまとめる

関数の役割の一つの「同じ処理をまとめる」ことの例を示す．そのために，大きさ 10 の配列 a[] と大きさ 100 の配列 b[] に整数の乱数を格納し，その最値を求めるプログラムを考える．リスト 2 のようなプログラムである．

このプログラムでは，乱数を使っている．乱数の発生方法については，このプリントの付録 (18 以降) に書いてある．又教科書の p.449 の srand 関数の説明の部分にも書いてある．

リスト 2: 関数を使わない最大値検索プログラム

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  int main(void){
6    int a[10], b[100], max_a, max_b, i;
7
8    srand((unsigned int)time(NULL));          /* 起動毎に異なる乱数発生のため */
9
10   for(i=0; i<10;   i++) a[i]=rand();      /* 配列 a[] の値設定 */
11   for(i=0; i<100;  i++) b[i]=rand();      /* 配列 b[] の値設定 */
12
13   /* ---- 配列 a[] の最大値検索 ----- */
14   max_a=a[0];
15   for(i=1; i<10; i++){
16     if(max_a<a[i]) max_a=a[i];
17   }
18
19   /* ---- 配列 b[] の最大値検索 ----- */
20   max_b=b[0];
21   for(i=1; i<100; i++){
22     if(max_b<b[i]) max_b=b[i];
23   }
24
25   printf("max a=%d\n",max_a);              /* 最大値印刷 */
26   printf("max b=%d\n",max_b);
27
28   return 0;
29 }
```

リスト 2 をよく見ると，最大値を求める部分はほとんど同じである．そこで，それを一つの関数にまとめることを考える．リスト 3 のようにすれば良い．これで，プログラムがすっきりした．こうすると，最大値を求めるアルゴリズムを変えたい場合でもプログラムの変更が容易である．

リスト 3: 関数を使用した最大値検索プログラム

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  int find_max(int n, int ix[]);             /* プロトタイプ宣言 */
6
```

```
7  /*=========================================================================*/
8  /*    main 関数                                                            */
9  /*=========================================================================*/
10 int main(void){
11   int a[10], b[100], max_a, max_b, i;
12
13   srand((unsigned int)time(NULL));          /* 起動毎に異なる乱数発生のため */
14
15   for(i=0; i<10;   i++) a[i]=rand();      /* 配列 a[] の値設定 */
16   for(i=0; i<100;  i++) b[i]=rand();      /* 配列 b[] の値設定 */
17
18   max_a=find_max(10, a);
19   max_b=find_max(100,b);
20
21   printf("max a=%d\n",max_a);              /* 最大値印刷 */
22   printf("max b=%d\n",max_b);
23
24   return 0;
25 }
26
27 /*=========================================================================*/
28 /*    最大値探索の関数                                                     */
29 /*=========================================================================*/
30 int find_max(int n, int ix[]){
31   int i, max;
32
33   /* ---- 配列 a[] の最大値検索 ----- */
34   max=ix[0];
35   for(i=1; i<n; i++){
36     if(max<ix[i]) max=ix[i];
37   }
38
39   return max;
40 }
```

### 2.3.2 処理をまとめてプログラムを分かりやすく

関数のもう一つの機能「処理をまとめてプログラムを分かりやすく」の例である．リスト 2 のメイン関数の部分の処理を分かり易くするために，関数を用いた例をリスト 4 に示す．

リスト 4: 処理を関数毎に分けて，わかり易くした例

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  /* ----  プロトタイプ宣言 -----  */
6  void make_data(int nx, int ix[], int ny, int iy[]);
7  int find_max(int n, int ix[]);
8  void print_results(int a, int b);
9
10 /*=========================================================================*/
11 /*    main 関数                                                            */
12 /*=========================================================================*/
13 int main(void){
14   int a[10], b[100], max_a, max_b;
15
16
```

```
17    make_data(10, a, 100, b);                /* 乱数データ作成 */
18
19    max_a=find_max(10, a);                   /* 最大値検索 */
20    max_b=find_max(100,b);
21
22    print_results(max_a, max_b);             /* 最大値印刷 */
23
24    return 0;
25  }
26
27
28  /*=========================================================================*/
29  /*     データ作成                                                          */
30  /*=========================================================================*/
31  void make_data(int nx, int ix[], int ny, int iy[]){
32    int i;
33
34    srand((unsigned int)time(NULL));          /* 起動毎に異なる乱数発生のため */
35
36    for(i=0; i<nx; i++) ix[i]=rand();
37    for(i=0; i<ny; i++) iy[i]=rand();
38
39  }
40
41
42  /*=========================================================================*/
43  /*     最大値探索の関数                                                    */
44  /*=========================================================================*/
45  int find_max(int n, int ix[]){
46    int i, max;
47
48    /* ---- 配列 a[] の最大値検索 ----- */
49    max=ix[0];
50    for(i=1; i<n; i++){
51      if(max<ix[i]) max=ix[i];
52    }
53
54    return max;
55  }
56
57
58  /*=========================================================================*/
59  /*     結果の印刷                                                          */
60  /*=========================================================================*/
61  void print_results(int a, int b){
62
63    printf("max a=%d\n",a);
64    printf("max b=%d\n",b);
65
66  }
```

# 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 3 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．

## 3.1　関数の作り方

### 3.1.1　プロトタイプ宣言

プロトタイプ宣言はコンパイラー[3]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．

プロトタイプ宣言は，図 3 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 [?] 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．

### 3.1.2　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]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

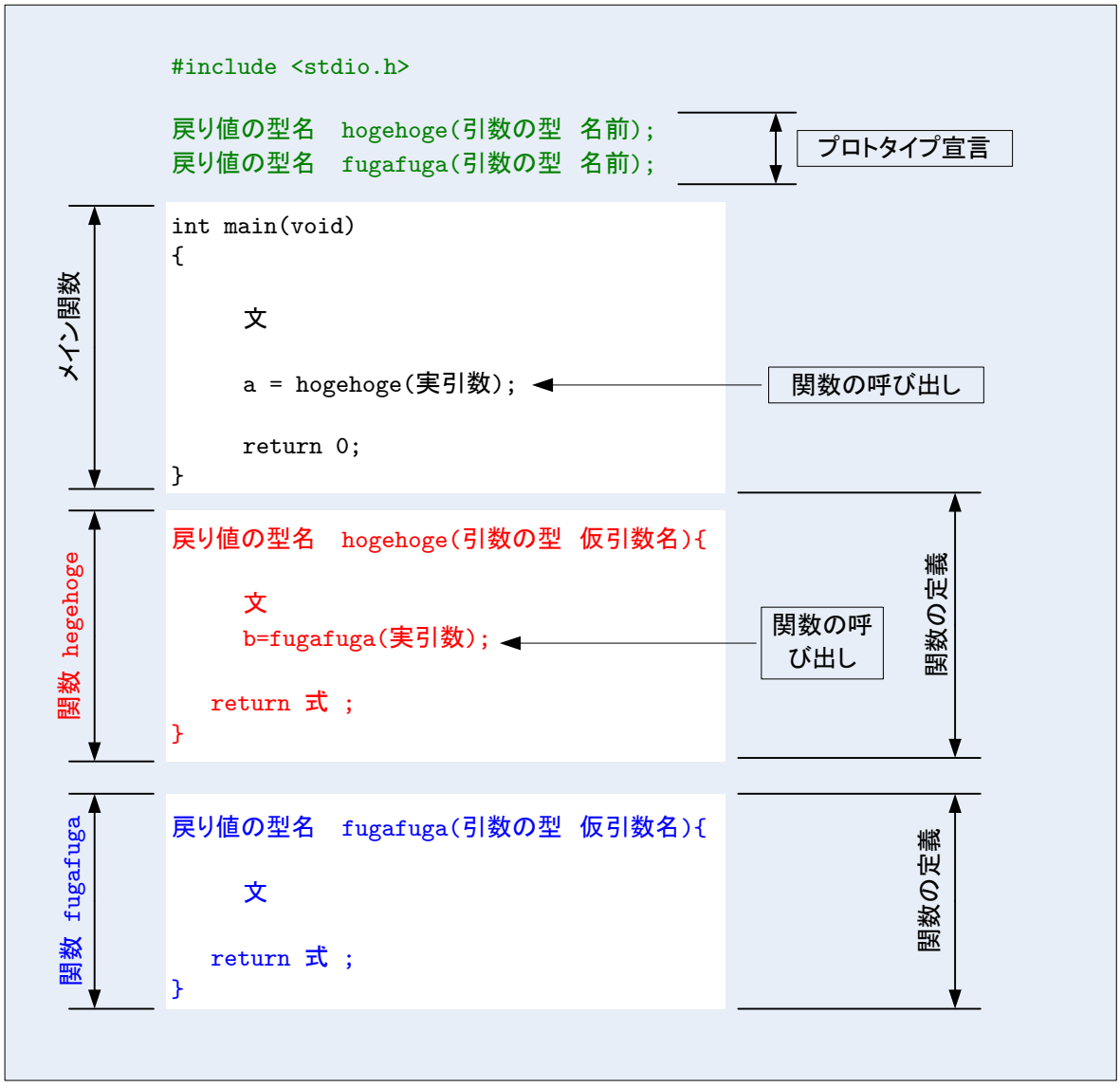


図 3: ユーザー定義関数を含んだソースプログラムの書き方

### 3.2 関数の使い方

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 関数と変数のスコープの関係

### 4.1 宣言の場所と有効範囲

変数はデータを記憶するためのもの—記憶領域に名前をつけたもの—である．その変数は，宣言する場所によって有効範囲が異なる．この有効範囲のことをこれを変数のスコープ (scope:範囲) という．有効範囲というのは，プログラム中で変数が使える場所のことである．具体的には，その変数を使って計算したり，表示することができる範囲で

```
printf("%f", hogehoge);
```

のように書いてもエラーにならない範囲である．

変数を宣言する場所，次の 3 箇所と考えてよい．

- 全ての関数の外側，プログラムの先頭付近で宣言した変数は全ての関数で有効である．これを**グローバル**変数と呼ぶ．
- 関数の先頭で宣言した変数は，その関数内のみで有効である．これを，**ローカル変数**と呼ぶ．また，関数の仮引数もローカル変数である．
- コードブロック—{　} で囲まれた部分[4]—の先頭で宣言した変数は，そのブロック内のみで有効になる．これもローカル変数のひとつであるが，ここでは**ブロック内宣言の変数**と呼ぶことにする．

この 3 つの変数宣言の場所とスコープの関係をを図 4 に示す．これをしっかり理解せよ．

これが理解できたならば，図 4 のプログラムの実行結果が，以下のようになることが分かるであろう．自分でこのプログラムの動作を追ってみよ．

実行結果

```
fuga at main = 222
hoge at main = 111
foo at test = 333
foo at test = 111
bar at for loop = 444
bar at for loop = 444
```

[4] if 文やループの時使った．

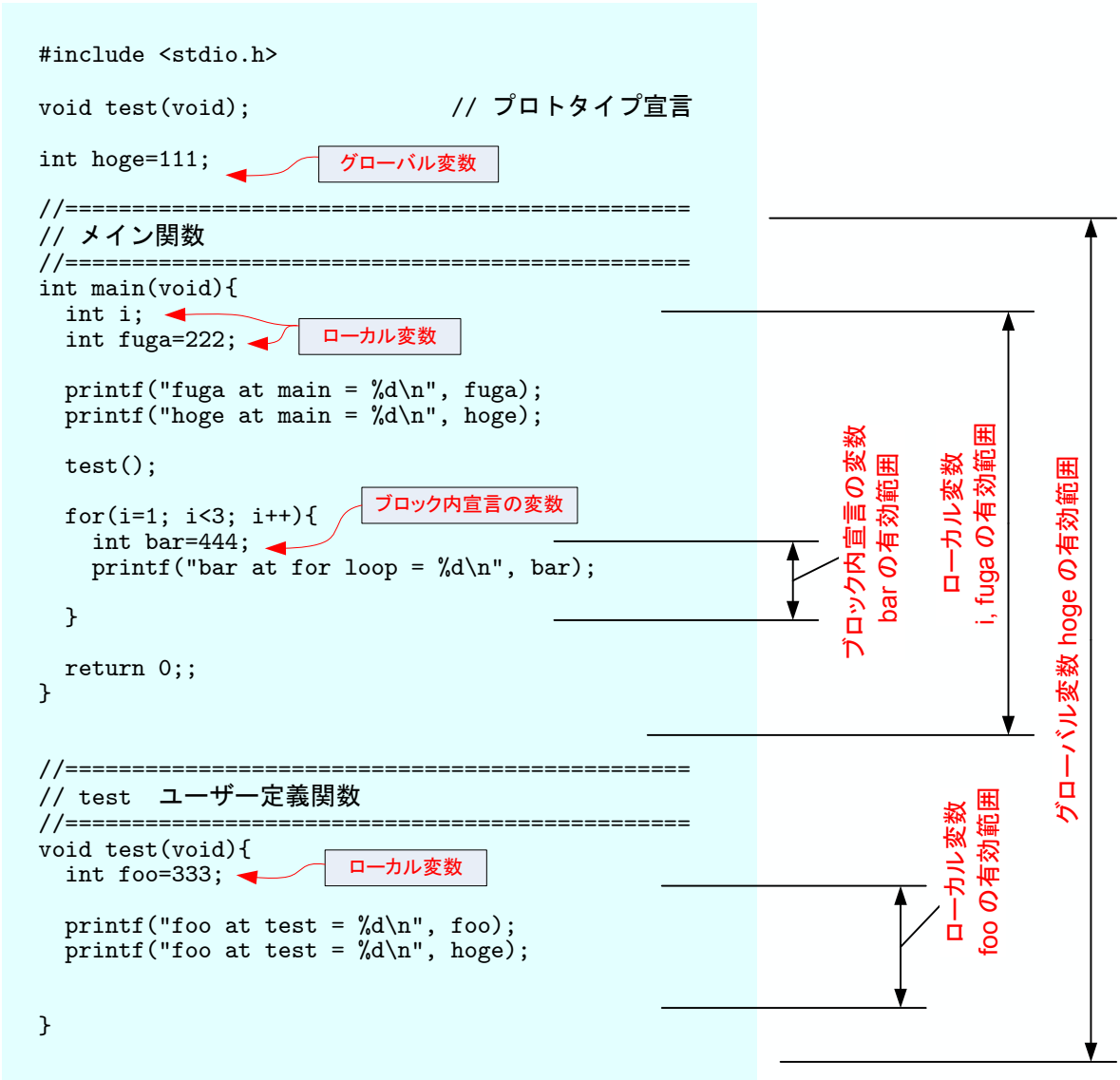


図 4: 変数のスコープ (有効範囲)

## 4.2 優先順位

どのような理由で，変数は宣言する場所でスコープが異なるのであろうか?．ちょっと考えると，グローバル変数だけでよいように思える．小さなプログラムであれば，グローバル変数だけでよい．実際，グローバル変数しかないプログラミング言語もある．このようなプログラミング言語は，大きなプログラムを作

ることは不可能で，ちょっとした小さなプログラム専用である．なぜならば，グローバル変数だけだと，同じ変数名を使うことができず，変数名の命名に大変苦労する．C 言語のように宣言する場所で，変数のスコープが異なると，同じ変数名を使うことができる．メイン関数で「hogehoge」と言う変数を使っていても，他のユーザー定義関数などで「hogehoge」を使っても，異なった物として取り扱ってくれる．これは，便利な機能で複数のプログラマーがそれぞれ関数を作成してひとつのプログラムを作る場合，各人が勝手な変数名を使うことができる．最初の疑問「変数宣言の場所でスコープが異なる理由?」の答えは，同じ変数名を使えることにするためで，それにより大規模なプログラムが開発できるようにしている．

それでは,「グローバル変数とローカル変数で同じ変数名を使った，どちらが実際使われるのか?」ということになる．そのため，同じ変数名には優先順位がある．優先度の高い順に並べると，(1) ブロック内宣言の変数，(2) ローカル変数，(3) グローバル変数となる．

# 5　データを渡す方法 (p.214)

関数を使って処理を行う場合，呼び出す側の関数とと呼び出される側の関数とでデータの受け渡しが必要である．呼び出す側は値 (あるいは変数) を用意し，呼び出される側は変数を用意する．呼び出す側の値を呼び出される側の変数にコピーすることで，データの受け渡しが行われる．呼び出す側の関数の値 (変数) を実引数，呼び出される側の関数の変数を仮引数と言う．図 3 に示しているので，実引数と仮引数の違いを理解せよ．

関数へのデータの渡し方に，2 つの方法がある (通常は値渡し)．

**値渡し**　呼び出す側と呼ばれる側の関数が各々変数を用意する．仮引数側の変数の値は，実引数の側の変数の値のコピーとなる．呼ばれた関数が変数の値を変えても，呼び出した側の変数値には，影響がない．

**アドレス渡し**　呼び出す側の実引数は，アドレスです．呼ばれる側は，ポ インターを用意して，実引数のアドレスを受け取ります．呼ばれた関 数が処理をすると，呼び出した側の実引数の変数にも影響があります．

それでは，データの受け渡しについて，教科書を見ながら，練習せよ．

## 5.1　値による呼び出し (p.214)

以下の練習問題を実施せよ．

**[練習 1]**　main 関数から，角度 [度] の値を関数に渡して，プログラマー作成の関数で，sin と cos と tan を計算して，印刷する．このプログラムを作成せよ．ただし，main 関数で繰り返し文を使い，0〜360[度] まで，1 度間隔でデータを送ること．ヒントは以下の通り．

- `#include <math.h>`を書く．三角関数のような数学関数を使う場合，math.h というヘッダーファイルが必要である．
- 三角関数は，`sin(a)`,`cos(a)`,`tan(a)` と書けば計算できる．単位は，[ラジアン] である．

– コンパイルには，-lm オプションが必要である．このオプションは数学関数を使うときに必要である．実際には次のようにする．C 言語のソースファイルが hogehoge.c で，機械語の実行ファイルが fugafuga である．

gcc -lm -o fugafuga hogehoge.c

[**練習 2**] リスト 5 の変数 i,j の値が入れ替わらない理由を考えよ．

リスト 5: 値渡しのため，値が交換できない

```
1  #include <stdio.h>
2
3  void swap(int i, int j);                /* プロトタイプ宣言 */
4
5  /*=========================================================================*/
6  /*      メイン関数                                                         */
7  /*=========================================================================*/
8  int main(void){
9    int a=2, b=3;
10
11   printf("a=%d   b=%d\n", a, b);
12
13   swap(a, b);
14
15   printf("a=%d   b=%d\n", a, b);
16
17   return 0;
18 }
19
20 /*=========================================================================*/
21 /*      swap関数                                                           */
22 /*=========================================================================*/
23 void swap(int i, int j){
24   int temp;
25
26   temp = i;
27   i=j;
28   j=temp;
29
30 }
```

## 5.2 アドレスによる呼び出し (p.216)

以下の練習問題を実施せよ．

[**練習 3**] リスト 6 の変数 i,j の値が入れ替わる理由を考えよ．

リスト 6: アドレス渡しを用いたため，値が交換さる例

```
1  #include <stdio.h>
2
3  void swap(int *i, int *j);              /* プロトタイプ宣言 */
4
5  /*=========================================================================*/
6  /*      メイン関数                                                         */
7  /*=========================================================================*/
```

```
 8 int main(void){
 9   int a=2, b=3;
10
11   printf("a=%d    b=%d\n", a, b);
12
13   swap(&a, &b);
14
15   printf("a=%d    b=%d\n", a, b);
16
17   return 0;
18 }
19
20 /*====================================================================*/
21 /*      swap関数                                                      */
22 /*====================================================================*/
23 void swap(int *i, int *j){
24   int temp;
25
26   temp = *i;
27   *i=*j;
28   *j=temp;
29 }
```

### 5.3 一次元配列を渡す (p.218)

以下の練習問題を実施せよ．

**[練習 4]** 要素数が 10000 個の一次元配列に，整数の乱数を格納して，その最大値を求めるプログラムを作成せよ．

- 最大値を求めるルーチンは，独立の関数とせよ．
- その関数では，最大値の出力も行うこと．

### 5.4 多次元配列を渡す (p.220)

以下の練習問題を実施せよ．

**[練習 5]** 要素数が 1000×1000 の二次元配列に，整数の乱数を格納して，その最大値を求めるプログラムを作成せよ．

- 最大値を求めるルーチンは，独立の関数とせよ．
- その関数では，最大値の出力も行うこと．

## 6 データを返す方法 (p.222)

関数から見るとデータを返す方法であるが，呼出元からみると，データを受け取る方法について説明する．返すデータのことを戻り値と言う．

### 6.1　1 個の戻り値を返す (p.222)

以下の練習問題を実施せよ．

[**練習 6**]　教科書の例を参考にして，

$$f(x) = x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040}$$

を計算する関数を作成せよ．そして，メインルーチンで，$x = 0$〜$x = 3.14$ まで，180 等分した値を求めよ．この値は何か?．注意事項は以下の通り．

- n 乗は，pow と言う関数をつかう．使い方は，教科書の P.437 を見よ．
- pow を使うためには，`#include <math.h>`が必要である．さらにコンパイルするときには，-lm オプションが必要である．

### 6.2　複数の戻り値 (p.223)

以下の練習問題を実施せよ．

[**練習 7**]　$\sin 2\theta$ と $2\sin\theta\cos\theta$ の値を 0〜360 で 1 [度] 間隔で，を計算するプログラムを作成せよ．

- ただし，これらの三角関数の値は，一つの関数で計算すること．
- プロトタイプの宣言は，以下のようにすること．

```
void keisan(double theta, double *s1, double *s2);
```

## 7　グローバル変数によるデータの受け渡し (p.228)

グローバル変数を使うと関数の独立性が失われ，再利用のする場合，支障をきたす．独立性の高い関数はコピーすると，そのまま他のプログラムでも使える．グローバル変数を使うと，関数のみならず，グローバル変数もコピーする必要が生じる．さらに，グローバル変数は全ての関数で使えるため，名前の衝突を考えなくてはならない．大きなプログラムになると，これは大変な問題を生じる．非常に分かりにくいバグの原因となるので，気を付けなくてはならい．

この講義で諸君が作る程度の短いプログラムならば，グローバル変数を使っても良いだろう．むしろポインターが分からなくて悩むよりは，グローバル変数を使った方がプログラムを楽しめて良いだろう．

[**練習 8**]　リスト 6 のプログラムをポインターを使わないでグローバル変数を使ったプログラムに書き換えよ．

## 8　main 関数への引数渡し (p.228)

教科書に書かれているこのテクニックは，よく使われる．しかし，本講義ではあまり使わないので説明しない．興味のある者は，自分で調べよ．

# 9 課題

## 9.1 内容

以下の課題を実施し，レポートとして提出すること．

**[問 1]** (復) 教科書 [1] の第 11 章 (pp.198–236) を 3 回読め．レポートには「3 回読んだ」と書け．

**[問 2]** (復) 本日配布したプリントを 2 回読め．レポートには「2 回読んだ」と書け．さらに，誤字・脱字，表現の悪いところ，間違いを指摘せよ．

**[問 3]** (復) キーボードから 3 辺の長さを読み取り，ヘロンの公式

$$s = \frac{a+b+c}{2}$$
$$S = \sqrt{s(s-a)(s-b)(s-c)}$$

を用いて面積を計算し，値を表示するプログラムを作成せよ．ここで，$S$ は三角形の面積，$(a, b, c)$ は辺の長さである．与えられた，辺の長さで三角形が構成できないときは，負の面積—例えば-1 など—を表示せするようにせよ．ここで，面積の計算にはユーザー定義関数を使うこと．

**[問 4]** (復) 次の数学関数

$$f(x) = \frac{x+3}{x^2+4} - x^2 - 10x + x\sin(x) + \sqrt{x^2+1} \qquad -10 \leqq x \leqq 10 \tag{1}$$

の最大値とその時の $x$ の値を求めよ．計算精度は，$10^{-4}$ とする．関数の計算には，ユーザー定義関数を使うこと．**ヒント**:以下のように考える．

- -10 から 10 まで $x$ の値を 0.0001 刻で変化させて，関数の値を計算する．これは繰り返し文を使う．
- 繰り返し文内で，最大値の判定を行う．それまでの最大値とその時関数の値を比較する．もし，この時の関数の値の方が大きかったら，最大値と $x$ の値を保存する変数—例えば `max_f` や `max_x`—に格納する．

**[問 5]** (復) 次の行列の転置行列を計算し，その結果を表示するプログラムを作成せよ．転置行列の計算は，ユーザー定義関数内で行うこと．

$$\boldsymbol{A} = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix}$$

**[問 6]** (復) 次の関数を計算する C 言語の関数を作成しなさい．

$$\begin{aligned} f(x) &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^4}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + \cdots \\ &= \sum_{k=0}^{N} \frac{(-1)^k}{(2k)!} x^{2k} \end{aligned}$$

そして，$N = 1, 3, 5, 7$ と変化させた場合，$f(x)$ と $\cos(x)$ の値を比較せよ．計算範囲は $[-\pi, \pi]$ とする．

[**問 7**] (復)[練習 2] と [練習 3] について，答えよ．

[**問 8**] (予)(復) 教科書 [1] の第 15 章と第 16 章を 2 回読め．レポートには「2 回読んだ」と書け．

[**問 9**] ここでの学習内容でわからないところがあれば，具体的に記述せよ．

## 9.2 レポート 提出要領

期限　6 月 19 日 (水) AM 8:45
用紙　A4 のレポート用紙．左上をホッチキスで綴じて，提出のこと．
提出場所　山本研究室の入口のポスト
表紙　表紙を 1 枚つけて，以下の項目を分かりやすく記述すること．

授業科目名「計算機応用」
課題名「関数」
提出日
5E　学籍番号　氏名

内容　2 ページ以降に問いに対する答えを分かりやすく記述すること．

# 付録 A　乱数

乱数とは，バラバラな数列のことを言う．とくに，自然数がめちゃくちゃに現れるようなものを自然乱数という．0 以上無限大までの全ての自然数を用いた自然乱数が考えられるが，実際上は最大の自然数を決め，その範囲で考えることが多い．我々が使用しているシステムの C 言語の場合，0〜2147483647 の範囲[5]の乱数を発生させることができる．

C 言語では，rand() 関数を使って乱数を発生させる．例えば，次のようにすると，rand() 関数が呼び出される度にその関数が乱数を返し，配列 a[i] に格納される．

```
for(i=0; i<ndata; i++){
  a[i]=rand();
}
```

コンピューターは，正確にプログラムのとおりに計算を行う．そのため，めちゃくちゃな順序で数が並んでいる乱数を発生させることは苦手である．先ほどの rand() 関数は，ある初期値[6]を使って，計算により乱数を決めている．同じ初期値をつかうと，同一の数列が発生するこのになる．これでは，乱数とは言い難いので，初期値を毎回変更するのが普通である．そのため，実行毎に異なる初期値を決める必要がある．現在の暦時刻を返す time() 関数を用いるのが一般的である．初期値の設定は，srand() 関数に引数 (符号無し整数) を渡すことにより可能である．次のようにすれば，毎回異なる初期値を決めることができる．

```
srand((unsigned int)time(NULL));
```

ただし，1 秒以内であれば time は同じ値となり，同一の数列となる．(unsigned int) は，キャストと呼ばれる強制型変換で，引き続く値の型を変換している．time() 関数の引数は暦時刻で，暦時刻がポインターで格納される．暦時刻を格納する必要がないときには，NULL と空ポインターを指定する．

乱数を発生させるためには，rand() と srand()，time() 関数が必要である．これらの関数を使うためには，関数の宣言が書かれているヘッダーファイルをインクルードしなくてはならない．rand() と srand() には stdlib.h，time() には time.h である.. したがって，配列 a[i] に 1024 個の乱数を格納するプログラムは次のようにする．

```
#include <stdlib.h>
#include <time.h>

int main(void){
  int a[1024], i;

  srand((unsigned int)time(NULL));

  for(i=0; i<1024; i++){
    a[i]=rand();
  }

  return 0;
}
```

[5] $0〜2^{31}$-1 の範囲である．
[6] 正確には seed(種) と言うらしい．

## 参考文献

[1] 林春比古. 新訂 C 言語入門 シニア編. ソフトバンク パブリッシング, 2004.