# これまでの復習 (後期中間試験に向けて)

山本昌志*

2006 年 12 月 1 日

**概 要**

後期中間試験に向けて，これまで学習した内容をまとめる．このプリントは試験対策用である．

## 1 前期末試験の傾向と対策

試験の範囲は，以下の通り．

- 第 17 回の講義から本日の第 24 回の講義に配布したプリント
- 教科書は，p.153–202 が範囲となる．2〜3 回位は読み直した方がよい．教科書の範囲で講義で触れなかった部分は試験には出さない．先週配布したプリントでは，教科書の p.216 までが試験範囲と記述しているが，そこまで進まなかった．
- このプリントの内容を十分理解して，試験に臨むこと．分からなければ，私を含めた他の人に聞くこと．

## 2 制御の流れ

### 2.1 ループ処理 (break，continue，goto)

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

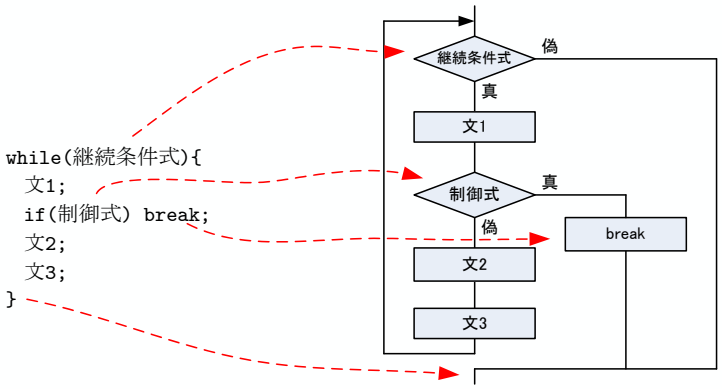


図 1: break 文を使って，ループからの脱出する構文

**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．

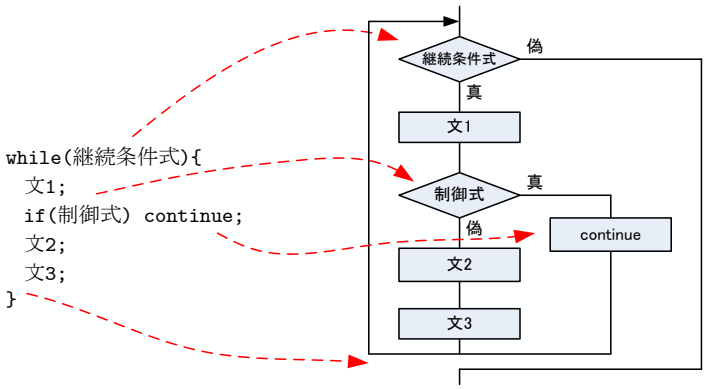


図 2: contine を使って，ループブロックをスキップする構文

**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．

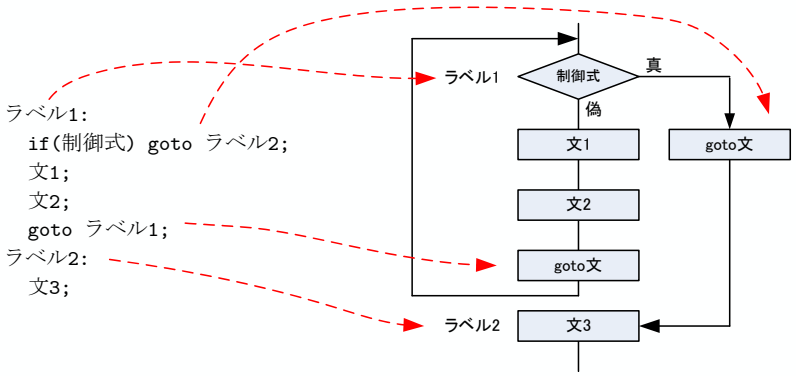


図 3: goto 文とラベルによる制御

# 3 関数

## 3.1 関数の役割と作成方法

関数とは，処理の手続きをまとめたものである．関数を使うことにより，C 言語のプログラムは

- 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．

とすることができる．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」ということが実現できる．

関数をつくり，使うためには，ソースプログラムを図 4 のように記述する．必要な記述は，以下の 3 つである．

- **プロトタイプ宣言**．関数の入出力 (引数や戻り値) の仕様 (個数と型) 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．

## 3.2 データの受渡し

呼出元から関数へ処理に必要なデータは，引数として渡す．呼出元の引数を実引数，呼び出された関数側の引数を仮引数 (図 5) と言う．関数を呼び出したとき，実引数の値が仮引数の変数にコピーされる．そ

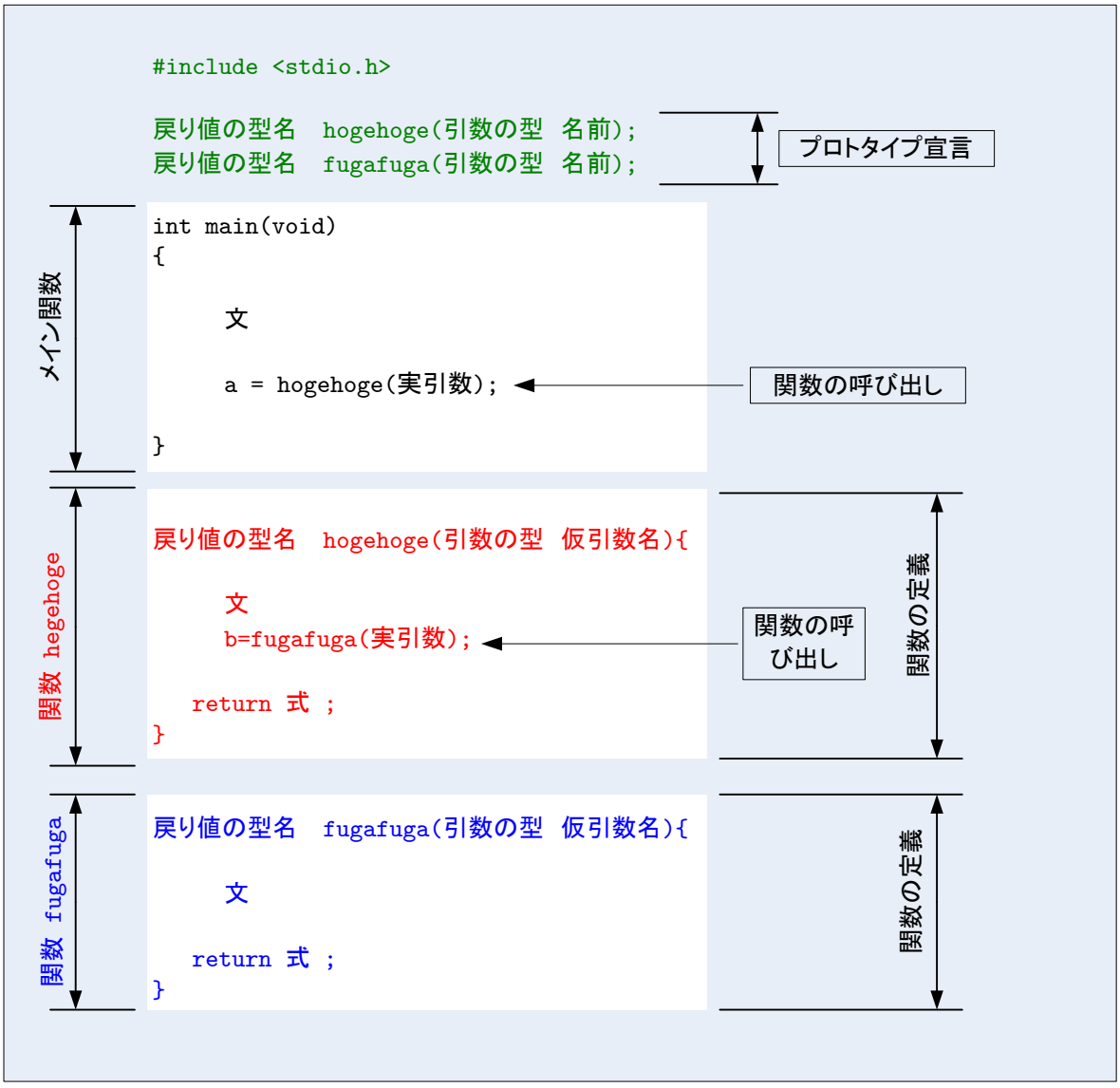


図 4: ユーザー定義関数を含んだソースプログラムの書き方

れを使って，関数は処理を行う．一方，関数が処理したデータを呼出元へ返すときには，戻り値をつかう．これは，return 文により返すことができる．

これらの引数と戻り値により，呼出元と関数の間でのデータの受渡しを行う．受渡しは，一方通行であることを忘れてはならない．

- 引数は呼び出し元から関数へ複数のデータを渡すことはできる (呼出元 → 関数)．しかし，関数から呼出元へデータを渡すことは不可能．
- 戻り値は，関数からデータを渡すことができる (呼出元 ← 関数)．そして，呼び出し元へ返せる値は一つに限られる．一方，呼出元から関数へデータを渡すことはできない．

必要な引数を伴って関数名を書けばその関数のプログラムを実行することができる．戻り値で返された値は，次のように変数に格納する．

```
y=f(x);
```

これで，変数 x の値を独立変数として $-x^2 + 10x + 8 + 10\sin^2 x$ の計算を行い，その結果を変数 y に格納する．注意:呼出元の x が実引数．

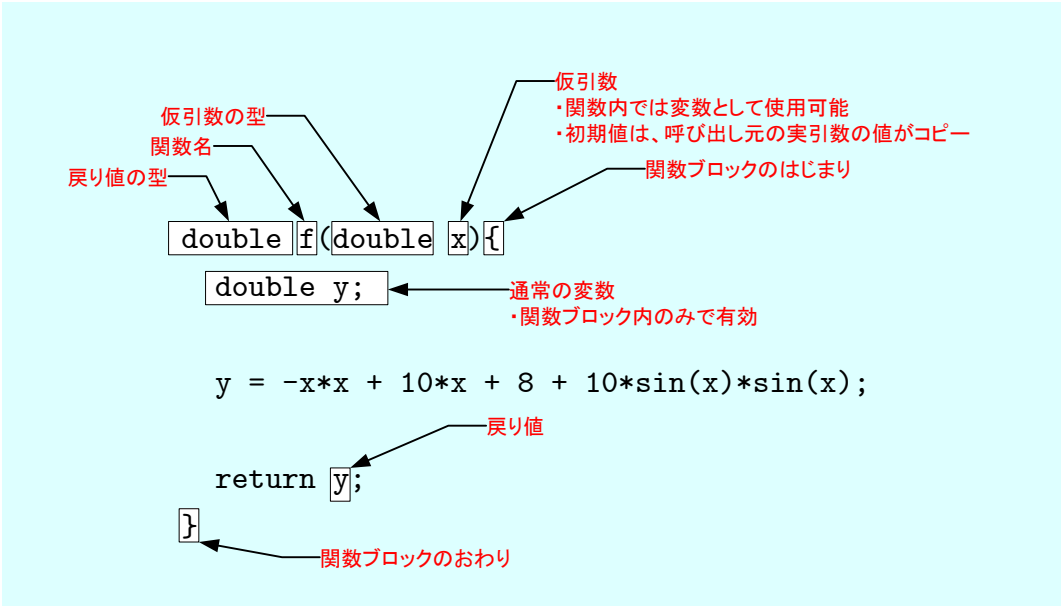


図 5: 数学関数 $f(x) = -x^2 + 10x + 8 + 10\sin^2 x$ を計算する C 言語のユーザー定義関数．この関数は p.8 のリスト 1 で使っている．

## 3.3 変数のスコープ

変数の有効範囲のことを**変数のスコープ**と言う．それは宣言する場所で決まる．変数のスコープは 3 種類あり，それぞれについて以下の箇条書きと図 6 に示す．

- 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[1]—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．

スコープが異なれば，同じ名前の変数名を使うことができる．同じ名前がある場合，優先度の高い変数が使われる．優先度の順序は，ブロック内宣言の変数 → ローカル変数 → グローバル変数となる．スコープが狭い程，優先順位が高い．

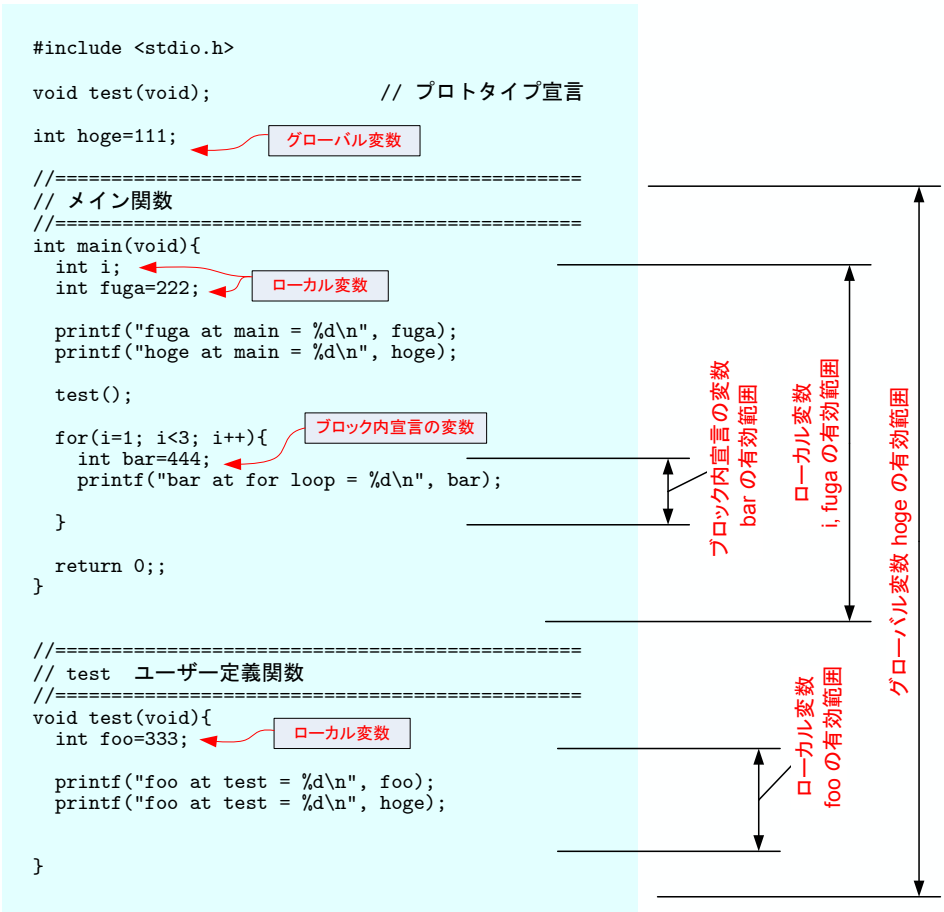


図 6: 変数のスコープ (有効範囲)

[1] if 文やループの時使った．

## 3.4 記憶クラス

メモリーにデータを格納する方法を示したものが **C 記憶クラス**である．C 言語には，4 個の記憶クラス指定子 (1)auto，(2)extern，(3)register，(4)static がある．これらは，変数宣言の先頭に

```
auto int hogehoge;
extern int fugafuga;
register int foo;
static int bar;
```

と書く．

記憶クラスは 4 種類あるが，この中で static はローカル変数[2]とグローバル変数では，動作が異なる．したがって，5 種類の動作があると考える．

記憶クラスを指定しない場合，自動変数 (auto) となる．デフォルトは自動変数．そのため，通常，自動変数を使う場合，記憶クラス指定子 auto は書かない．

表 1: 記憶クラスの特徴

| 記憶クラス | 指定子 | 特徴 |
|---|---|---|
| 自動変数 | auto | ローカル変数の場合は，関数が呼び出される毎に記憶領域の確保と初期化 (初期値の指定がある場合) が行われ，関数の処理が終わったときにその変数は消滅する．グローバル変数の場合は，プログラム実行中，変数の記憶領域は保持される． |
| ローカル変数の静的変数 | static | プログラムの実行に先立って，記憶領域の確保と初期が行われる．プログラムの実行中，記憶領域は保持される． |
| グローバル変数の静的変数 | static | 分割コンパイルの場合，そのファイルのみで使えるグローバル変数ということを示している． |
| 外部変数 | extern | この記憶クラスはグローバル変数のみに使う．これが指定されると，記憶領域の確保は行われず，どこか他のファイルにこの変数があることを示す．記憶領域は，どこか他のファイルにある変数を使う． |
| レジスター変数 | register | レジスターを記憶領域として使う．レジスターは CPU の記憶領域で，メインメモリーよりも高速のアクセスが可能である．したがって，処理の高速になる． |

[2]ブロック内宣言の変数もローカル変数とする．

## 3.5 関数の例

### 3.5.1 関数を使った基本プログラム

関数

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

の最大値を求めるプログラムである．これは，10 月 20 日配布のプリントに書かれている例.

リスト 1: 数学関数の最大値を求めるプログラム

```
1  #include <stdio.h>
2  #include <math.h>
3
4  double f(double x);              //プロトタイプ宣言
5
6  //=========================================================
7  // メイン関数
8  //=========================================================
9  int main(void){
10   double x, dx, xmin, xmax, y;
11   double max_y, max_x;
12   int i, ncal;
13
14   //--- 計算条件設定 ---
15   xmax = 10;
16   xmin = -10;
17   dx = 0.0001;
18   ncal = (xmax-xmin)/dx;
19
20   //--- 暫定最大値 ---
21   max_x = xmin;
22   max_y = f(xmin);
23
24   //--- 最大値検索 ---
25   for(i=1; i<=ncal; i++){
26     x = xmin + i*dx;
27     y = f(x);
28     if(max_y <= y){          //最大値が見つかった場合
29       max_x = x;
30       max_y = y;
31     }
32   }
33
34   printf("%fのとき，最大%fとなる．\n",max_x, max_y);
35
36   return 0;
37
38 }
39
40 //=========================================================
41 // ユーザー定義関数
42 //=========================================================
43 double f(double x){
44   double y;
45
46   y = -x*x + 10*x + 8 + 10*sin(x)*sin(x);          // 関数の計算
47
48   return y;
```

```
49 }
```

### 3.5.2 戻り値の工夫

リスト 2 は，ヘロン公式

$$s = \frac{a+b+c}{2} \tag{2}$$

$$S = \sqrt{s(s-a)(s-b)(s-c)} \tag{3}$$

を使って三角形の面積 $S$ を計算するプログラムである．これは，12 月 1 日配布のプリントに書かれている例．

もし，仮引数で渡された三辺では三角形ができなかった場合，戻り値を-999 とする．三角形ができた場合の面積は必ず正である．もし戻り値が負の値となった場合，呼び出し元は渡した実引数が不適で，三角形が成立しなかったことが分かる．-999 のようなヘンテコリンな数字を戻り値とするのはプログラミングの定石である．そして，倍精度実数の比較には等価演算子 (==) を使わないこと．

リスト 2: 三角形の面積を計算するプログラム．

```
1  #include <stdio.h>
2  #include <math.h>
3
4  double helon(double a, double b, double c);  //プロトタイプ宣言
5
6  //=============================================================
7  // メイン関数
8  //=============================================================
9  int main(void)
10 {
11   double hen1, hen2, hen3;
12   double menseki;
13
14   printf("辺1の長さ?\t");
15   scanf("%lf",&hen1);
16   printf("辺2の長さ?\t");
17   scanf("%lf",&hen2);
18   printf("辺3の長さ?\t");
19   scanf("%lf",&hen3);
20
21   menseki = helon(hen1, hen2, hen3);
22
23   if(menseki < -990){
24     printf("入力した辺では，三角形はできません!!!!!\n");
25   }else{
26     printf("面積は，%fです．\n", menseki);
27   }
28
29   return 0;
30 }
31
32 //=============================================================
33 // ユーザー定義関数
34 //=============================================================
35 double helon(double a, double b, double c)
```

```
36 {
37   double s, S, test;
38
39   s=(a+b+c)/2;
40   test=s*(s-a)*(s-b)*(s-c);
41
42   printf("%f\t%f\t%f\n",a,b,c);
43   if(test <= 0){
44     S = -999.0;
45   }else{
46     S = sqrt(test);
47   }
48
49   return S;
50 }
```

### 3.5.3 複数の戻り値

三角形の面積と周長のように2つの値を計算する関数の場合，戻り値でその値を返すことができない．引数を使って，値を返すこともできない．

- 戻り値で呼び出し元へ返せる値は，一つに限られる．
- 引数は呼び出し元から関数へデータを渡すことはできるが，その逆は不可能．ようするに引数を使ったデータの受け渡しは一方通行である．

このような場合，グローバル変数を使って，値を返す[3]．

リスト 3: 三角形の面積と周長を計算するプログラム．

```
1  #include <stdio.h>
2  #include <math.h>
3
4  void info_tri(double a, double b, double c);     // プロトタイプ宣言
5  double S, total_len;                               // グローバル変数
6
7  //============================================================
8  // メイン関数
9  //============================================================
10 int main(void)
11 {
12   double hen1, hen2, hen3;
13   double menseki;
14
15   printf("辺1の長さ?\t");
16   scanf("%lf",&hen1);
17   printf("辺2の長さ?\t");
18   scanf("%lf",&hen2);
19   printf("辺3の長さ?\t");
20   scanf("%lf",&hen3);
21
22   info_tri(hen1, hen2, hen3);
23
24   if(S < -990){
25     printf("入力した辺では，三角形はできません!!!!!\n");
```

[3]ポインターを使う方法が最も良い方法であるが，諸君はまだ学習していない．

```
26    }else{
27      printf("面積は, %fです. \n", S);
28      printf("周長は, %fです. \n", total_len);
29    }
30
31    return 0;
32  }
33
34  //=============================================================
35  // ユーザー定義関数
36  //=============================================================
37  void info_tri(double a, double b, double c)
38  {
39    double s, test;
40
41    s=(a+b+c)/2;
42    test=s*(s-a)*(s-b)*(s-c);
43
44    if(test<=0){
45      S = -999.0;
46    }else{
47      S = sqrt(test);
48      total_len = a+b+c;
49    }
50
51  }
```

### 3.5.4 void 型を使った関数

引数や戻り値の無い関数を宣言するときには，void と型宣言をする．

リスト 4: void 型の関数の例.

```
1   #include <stdio.h>
2
3   void write_file(void);                          // プロトタイプ宣言
4
5   //=============================================================================
6   // メイン関数
7   //=============================================================================
8   int main(void){
9
10    write_file();                                 // 関数の呼出
11
12    return 0;
13  }
14
15  //=============================================================================
16  // void型の関数
17  //=============================================================================
18  void write_file(void){
19    FILE *fuga;
20
21    fuga = fopen("hello.txt", "w");               // ファイルのオープン
22
23    fprintf(fuga, "Hello World !!!\n");           // ファイルへの書き出し
24
25    fclose(fuga);                                 // ファイルのクローズ
26  }
```

### 3.5.5 ファイルへの書き込み

先週は，さまざまなユーザー定義関数を学習した．その内容は，以下のようなものであった．

- **FILE *out;**　ファイルの情報を格納する変数宣言．*out がファイルの情報を格納する変数である．out は fugafuga のようにどんな名前でも良い．しかし，先頭のアスタリスク (*) は必須である．
- **out=fopen(”data.txt”,”w”);**　書き込み用にファイルを開く．data.txt がファイル名で，w が書き込み用にファイルを開くことを示している．ファイル名は適当に変えてもよい．fopen() 関数の戻り値は，ファイル情報を格納する変数に代入する．
- **fprintf(out, ”%f\t%f\n”, x, y);**　ファイルにデータを書き込む．ファイルに書き込む fprintf() 関数は，ディスプレイに表示する printf() 関数とそっくりである．書き込むファイルを指定するファイル情報の変数を最初に書く—ことが異なる．
- **fclose(out);**　ファイルを閉じる．使い終わったファイルは閉じなくてはならない．

リスト 5: 数学関数の $x$ と $y$ を計算し，ファイルに保存するプログラム．

```
1  #include <stdio.h>
2
3  double hogehoge(double x);                          // プロトタイプ宣言
4
5  //=========================================================================
6  // メイン関数
7  //=========================================================================
8  int main(void){
9
10   FILE *out;                                        // ファイルの情報を格納
11   int i, np;
12   double x, dx, xmax, xmin;
13
14   printf("xの最小値\t");                             // キーボード入力
15   scanf("%lf", &xmin);
16   printf("xの最大値\t");
17   scanf("%lf", &xmax);
18   printf("プロット点の数\t");
19   scanf("%d", &np);
20
21   dx = (xmax - xmin)/np;                            //データ間隔計算
22
23   out=fopen("data.txt","w");                        //ファイルをオープン
24
25   for(i=0; i<=np; i++){
26     x = xmin + i*dx;
27     fprintf(out, "%f\t%f\n", x, hogehoge(x));       //データの書き込み
28   }
29
30   fclose(out);                                      //ファイルをクローズ
31
32   return 0;
33 }
```

```
34
35 //=============================================================================
36 //  プロットする数学関数
37 //=============================================================================
38 double hogehoge(double x){
39   double y;
40
41   y=(3*x+2)/(x+2);
42
43   return y;
44 }
```