プログラミング演習 第5回
関数

目 次

1 目標 **1**

2 御託宣 **1**

3 前回までの復習と補足 **1**

4 関数の宣言と記述 **2**

4.1 補足: main が返す値は？ 2

5 実習 **3**

5.1 例 1. ユークリッドの互除法 3

5.2 例 2: ヘロンの公式を使った 3 角形の面積 6

5.2.1 再び数学関数ライブラリの使い方 7

5.2.2 補足 1: 変数のスコープ 7

5.2.3 補足 2: エラー処理 8

5.3 例 3: 再帰 (recursion) 9

6 練習問題 **10**

1 目標

- while 文の復習
- C 言語における関数の宣言の仕方, 書き方を知る.
- 再帰と言う言葉を知る.

2 御託宣

今回は関数[1] の書き方をやります. 例えば, main() と言うのは関数ですし, これまでに, printf(), gets(), sscanf() も関数です. 関数と言うのは, 通常は変数 (プログラミングでは引数と言う) を受け取ってその値から別の値を計算して出力するものですが, C 言語では printf の様に計算機に何らかの操作を与えるものも関数と言います. printf(), gets(), sscanf() 等を使う時に ( ) の括弧の中にあるものは, 引数と言われます. printf 等は実は値を返しています. これまでのプログラムでは, その値を使っていないだけです. どのような値を返すかは, man コマンド で調べる事ができます.

なお, 関数は数学のように何らかの処理をして値を返すという行為を実現するためだけにあるのではありません. 実際のプログラミングでは, この事よりもプログラムの処理を適当な大きさの単位に分割するという目的で主に使われます. 人間が一度に把握できる内容は, 概ね画面に表示可能な量の情報であると経験上知られており, それ以下の単位に処理を分割するというのは, 大きなプログラムを間違いなく書くための基本的なテクニークです. 今日書くプログラムも, 関数を使わずに実現する事が可能ですが, 実際には今日のようにプログラムする方が良いわけです. これは, 構造化プログラミングのための仕組みの一つです.

3 前回までの復習と補足

これまでに, 変数を宣言する事を学びました. さて, 変数が宣言された時点で, その値はいくらなのでしょうか ? 実は, C 言語には, それを決めるための仕様はありません. 従って, 変数は代入をしない限りどんな値を持っているかはわからないのです. ちなみに次のプログラムを書いてコンパイルして実行して下さい.

```
/* Variable testing program */

#include <stdio.h>

main()
{
     int r;
     printf("The value of r is %d.\n", r);
}
```

私が実験したところ r の値は -2146890064 でした. 例えば, その後 r の値が 0 か否かを判定する while ループがあるとその while ループは実行されません.

[1] 本来は関数ではなく, 函数です. function の音読みが中国語になってそのまま日本に輸入された言葉です. function の本来の意味は, 辞書で索いて下さい. その意味を知っておいた方が, より理解がしやすいです. 例えば, キーボード の最上部に F1 から F15 までの刻印があるキーがありますが, これはファンクションキーと呼ばれています. これを「関数鍵」と訳すと変ですよね.

4　関数の宣言と記述

関数は, 使う前に宣言しなければなりません. その意味で変数と同じ扱いを受けます. コンパイラは, その宣言を見て機械語を作り出すのです.

printf 等の関数をこれまで使って来ましたが, これらも実は宣言してから使っています. これらの宣言は今まで記述して来た,

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

で自動的になされた事になっているのです. この文は, stdio.h というファイルを取り込むと言う意味で, stdio.h には, printf が宣言されています. 不等号 $<, >$ で囲まれている場合/usr/include ディレクトリにあるファイルを読み込みます.

```
% less /usr/include/stdio.h
```

を実行して, printf が宣言されている事を確かめてください.

関数の宣言は, 次の形になります.

関数の返り値の型　関数の名前 (引数の型 引数名, …)

関数の宣言では, 次の決まり (言語としての仕様) があります.

- 関数の返り値の型および引数の型として許されるのは, 基本データ型, ポインタ, 構造体, 共有体で, 配列型を使う事はできない.
- 関数の返り値の型が指定されていない時には, 自動的に整数型を返す関数となる.

従って, 今まで何も気にしていませんでしたが, main() は整数型の値を返す関数です.

また, 関数そのもの記述は次のようになります.

```
関数の返り値の型　関数の名前 (引数の型 引数名, … )
{
        関数内で用いる変数 (局所変数) の宣言 ;

        関数の処理 ;

        return 式 ;
}
```

最後の return の部分で, この名前の関数は, そこの式の値を返します.

4.1　補足: main が返す値は ?

上でも述べた通り, main() は整数値を返す関数です. もし, main() が値を返すようにプログラムを書いたら, その値はどうなるのでしょうか ? 実は, これは OS によって扱いが異なります. Macintosh OS では, この値は何も意味を持ちません. しかし, 皆さんがプログラムを作っている UNIX では, その値は Shell 変数 status に格納されます. この値によってコマンドの実行が成功したか失敗したかを知る事ができます. したがって UNIX では, コマンド実行が成功すれば 0, 失敗すればそれ以外の値を返すようにするのが, 望ましいプログラミングです. また, より実践的には, その値によってプログラムがどの段階で失敗しているかを知る事ができるように, プログラムを組みます.

5 実習

5.1 例 1. ユークリッドの互除法

前回のレポート問題 4 の 3 にあげていた問題でした. プログラムは以下のようになります.

```
#include <stdio.h>

main()
{
  int a, b, r;        /* 変数 a,b,r が整数型であることを宣言*/
  int m;

   r = 1 ;

  printf("二つの整数を入力して下さい >>");
  scanf("%d %d", &a, &b);  /* & を忘れずに */

  if(b > a)
    {
      m = a;
      a = b;
      b = m;
     }

         while(r > 0)
    {       /* 余り r が 0 でない限り while 文の中身を繰り返す*/
       r = a%b;
       a =b;
       b = r;
     }
    printf("最大公約数は%d です\n", a);

}
```

While 文の中味が最大公約数 (GCD) を求めるアルゴリズムです.

この部分を GCD(a,b) 関数として定義し, プログラムを書き直すと以下のように, main 内部がすっきりし処理の内容が理解できやすくなります.

ファイル名を 5-1.c とし書いてみましょう.

```
#include<stdio.h>

int GCD(int, int);

main()
{

  int a, b;

  printf("二つの整数を入力して下さい >>");
  scanf("%d %d", &a, &b);
  printf("二つの整数%d %d の最大公約数は%d です\n", a, b, GCD(a,b));
}



int GCD(int a, int b)
{
     int m, r;
     r = 1 ;

     if(b > a)
    {
      m = a;
      a = b;
      b = m;
     }

         while(r > 0)
     {       /* 余り r が 0 でない限り while 文の中身を繰り返す*/
        r = a%b;
        a =b;
        b = r;
      }
      return a;
}
```

負の整数も意識したプログラム

```
#include<stdio.h>

int GCD(int, int);
int abs(int);

main()
{
  int a, b;

  printf("二つの整数を入力して下さい >>");
  scanf("%d %d", &a, &b);
  printf("二つの整数%d %d の最大公約数は%d です\n", a, b, GCD(a,b));
}



int GCD(int a, int b)
{
     int m, r;
     r = 1;

     if(abs(b) > abs(a))
    {
      m = a;
      a = b;
      b = m;
     }

         while(r > 0)
     {       /* 余り r が 0 でない限り while 文の中身を繰り返す*/
        r = a%b;
        a =b;
        b = r;
      }
      return abs(a);
}

int abs(int x)
 {
    if(x >= 0) x = x;
    else x = -x;
    return x;
}
```

5.2 例 2: ヘロンの公式を使った 3 角形の面積

次は, 3 角形の 3 辺からその面積を求めるプログラムをヘロンの公式を使って書いて見ます.

この例では, 以前一度使いました数学関数ライブラリを使うのでそこで使う関数を宣言するために, 最初に math.h というファイルを include しています. 使う関数は, sqrt で 平方根をとる関数です. 数学関数ライブラリでどのような関数が利用できるかは, ファイル /usr/include/math.h を読めばわかります[2] .

```
/* File name 5-2.c */
/* Computes the area of triangle */

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

double heron(double a, double b, double c);

main()
{
     double a, b, c, area;  /* a, b, c are edges.  */
     char nyuryoku[80];

     printf("3 辺の長さを空白で区切って入力してください>>");
     gets(nyuryoku);
     sscanf(nyuryoku, "%lf %lf %lf", &a, &b, &c);

     area = heron(a, b, c);
     printf("与えられた 3 角形の面積は %f\n", area);
}

double heron(double a, double b, double c)
{
     double s;

     s = (a + b + c)/2.0 ;

     return sqrt(s*(s-a)*(s-b)*(s-c));
}
```

[2] これらの include ファイルを一度は読む事

5.2.1 再び数学関数ライブラリの使い方

上のプログラムは, 通常の cc コマンドでは, コンパイルできません.

```
% cc 5-2.c
```

を実行して, 出て来るメッセージを確認してください. 私の環境では, 次のようになりました.

```
Undefined                       first referenced
 symbol                             in file
sqrt                                5-2.o
ld: fatal: Symbol referencing errors. No output written to a.out
```

(意味は第 2 回のレジュメを参照)

```
% cc 5-2.c -lm
```

とコンパイルをすればいいですね.

5.2.2 補足 1: 変数のスコープ

上のプログラムでは, heron という関数の中で s という double 型の変数を宣言しています. しかし, この形の宣言では, この変数を参照できるのはこの関数だけで, この値を他で利用する事はできません. 同様に, main で宣言されている nyuryoku も, heron に何らかの形で引数として渡さない限り, 参照する事はできません.

すなわち, 関数内で宣言された変数は, 通常はその関数の中だけで有効なのです. これは, プログラミングにとってとても重要な事です. プログラミングは, つまるところデータの変形方法の指定です. 一つの変数の値が, 至る所で変化するとプログラムの流れを追うのがとても大変になります. だから, そのような事にならないように, 言語の仕様の方で工夫してあると考えてください.

5.2.3 補足 2: エラー処理

上のプログラムで, 例えば 3 辺の値として 1, 2, 5 を代入するとエラーが発生します (実験せよ). これは, 1, 2, 5 を 3 辺とする 3 角形が存在しないからです. ということで, このエラー処理まで考慮に入れたプログラムの例を以下に書いておきます. [3]

三角形ができない場合は,「三角形になりません」と出力し, プログラムを終了させます.

ここでは, main() は正常終了したときに, shell に 0 を返すようにしました. また, fprintf は標準エラー出力にメッセージを出すために使われており, exit() は shell に括弧内の値を返した後プログラムを終了する関数です.

```
/* File name 5-2-1.c */
/* Computes the area of triangle */

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

double heron(double a, double b, double c);

int main()
{
     double a, b, c, area;
     char nyuryoku[80];

     printf("3 辺の長さを空白で区切って入力してください>>");
     gets(nyuryoku);
     sscanf(nyuryoku, "%lf %lf %lf", &a, &b, &c);

     area = heron(a, b, c);
     printf("与えられた 3 角形の面積は %f\n", area);
     return 0;
}

double heron(double a, double b, double c)
{
     double s, s2;

     s = (a + b + c)/2.0 ;
     s2= s*(s-a)*(s-b)*(s-c);
     if(s2 <= 0){
     fprintf(stderr,"%lf, %lf, %lf は三角形になりません\n",a,b,c);
      exit(1);
     }
     return sqrt(s2);
}
```

[3] 実は, センター試験のプログラムも数字以外の文字を入れるとエラーを発生させます.

5.3 例 3: 再帰 (recursion)

再帰と言うのは, 感覚的に漸化式をそのままプログラムする様なものです. 例えば, $S(n) = \sum_{k=1}^{n} k$ は漸化式 $S(n) = S(n-1) + n,\ S(1) = 1$ を満たします. ほとんどの高級言語では, これをそのままプログラムで表現する事ができます. C 言語では, 次のようになります. 次の例の sum がまさに上の漸化式の定義通りになっています.

```
/* filename 5-3.c */

#include <stdio.h>

int sum(long n);

main()
{
       int n;

       printf("正の整数を入力してください >> ");
       scanf("%d", &n);
       printf("1 から %d までの和は, %d です. \n", n, sum(n));
}

int sum(long n)
{
      if ( n==1 )
           return 1;
      else
           return sum(n-1)+n;
}
```

6 練習問題

1. 再帰を用いて Fibonacci 数列 $f_n = f_{n-1} + f_{n-2}$, $f_0 = f_1 = 1$ の第 n 項を出力するプログラムを書け.
2. 入力された整数 x の n 乗を求めるプログラムを関数`power(x,n)` を定義して書け.
3. 階乗 $n!$ を求めるプログラムを, 再帰を用いた関数`kaijo(n)` を定義して書け.
4. (レポート問題) さらに拡張して 2 項係数 $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ を求めるプログラムを関数`comb(n,k)` を定義して作れ.

注意

1. 今回のレポート問題の Subject 名は, creportno.5-4 です. 間違わないようにして下さい.
2. 締切は 5 月 23 日 (火) です.