Cプログラミング1(再)
第2回

講義では、Cプログラミングの基本を学び
演習では、やや実践的なプログラミングを通して学ぶ

前回のレポートから

- 前回の宿題「数あてゲーム」の説明において、次のように書いていたものがいた：

  これはコンピュータがランダムに設定した数字を人間が当てるゲームである

この説明でどこかおかしなところはないだろうか？

コンピュータの用語と日常的な用語の違い

物理において「力」というのは日常語の「力」とは違うものだ、
という話を「物理学」でしたのは覚えているだろうか？？？

ここでは「数」と「数字」がそれにあたる

2 は数だろうか、数字だろうか？

それでは 20 は数だろうか、数字だろうか？

似た例では、「文字」と「文字列」がある

a は文字だろうか、文字列だろうか（単語）

apple は文字だろうか、文字列だろうか

実はいずれも文字列ではなく、Cの意味の「文字」でもない

1. Cプログラムを観察しよう

おなじみのプログラム：これは何をするものか？

```
# include<stdio.h>

int main(void)
{
        printf("Hello, world!¥n");

        return 0;
}
```

2. Cの流儀を理解しよう

実行開始の場所と終了の場所

　mainはCプログラムのエントリーポイント、かつ終了の場所

Cプログラムは関数の集まり

　関数

　　我々の知っている関数の例: sin　cos　log

　　数学的には *f*(x)　とか　*g*(x,y) のように書かれる

引数（ひきすう, argument）

基本的に関数とは、引数をとり、何らかの計算をして、値を返す(return)もの

3. Cのプログラムを見てみよう

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4: int factorial(int n);
 5:
 6: int main(int argc, char *argv[])
 7: {
 8:     int x, n;
 9:
10:     if (argc < 2) {
11:         fprintf(stderr, “Usage: %s number¥n”, argv[0]);
12:         return 1;
13:     }
14:     n = atoi(argv[1]);
15:     x = factorial(n);
16:     printf(“%d! = %d¥n”, n, x);
17:     return 0;
18: }
```

ヘッダファイルの取込

factorial関数の宣言

main関数の定義

変数の宣言

条件文

factorial関数の呼出

値の戻し(return)

```
19:
20: int factorial(int n)
21: {
22:     int  i,  x = 1;
23:
24:     for (i = 2; i <= n; i++)
25:         x *= i;
26:     return x;
27: }
```

factorial関数の定義

変数の宣言

繰り返し文

値の戻し(return)

註：factorial　階乗

関数を使うには、事前に宣言・定義されていないといけない

４．Cプログラムの構造解析

(1) ヘッダファイル

(2) 関数宣言

(3) main関数定義

(4) main関数以外の関数(ここではfactorial)定義

```
1:  #include <stdio.h>
2:  #include <stdlib.h>
3:
4:  int factorial(int n);
5:
6:  int main(int argc, char *argv[])
7:  {
8:      int x, n;
9:
10:     if (argc < 2) {
11:         fprintf(stderr, “Usage: %s number¥n”, argv[0]);
12:         return 1;
13:     }
14:     n = atoi(argv[1]);
15:     x = factorial(n);
16:     printf(“%d! = %d¥n”, n, x);
17:     return 0;
18: }
19:
20: int factorial(int n)
21: {
22:     int  i,  x = 1;
23:
24:     for (i = 2; i <= n; i++)
25:         x *= i;
26:     return x;
27:  }
```

ちょっと複雑なCプログラム

```
1:  #include <stdio.h>
2:  #include <stdlib.h>
3:
4:  int factorial(int n);
5:
6:  int main(int argc, char *argv[])
7:  {
8:      int x, n;
9:
10:     if (argc < 2) {
11:         fprintf(stderr, “Usage: %s number¥n”, argv[0]);
12:         return 1;
13:     }
14:     n = atoi(argv[1]);
15:     x = factorial(n);
16:     printf(“%d! = %d¥n”, n, x);
17:     return 0;
18: }
19:
20: int factorial(int n)
21: {
22:     int  i,  x = 1;
23:
24:     for (i = 2; i <= n; i++)
25:         x *= i;
26:     return x;
27: }
```

参考：

stdio.h で宣言されているもの:

stderr, fprintf, printf

stdlib.h で宣言されているもの:

atoi

関数を使うには、事前に宣言・定義されていないといけない

このプログラムの構造

main関数

- fprintf → stdio
- atoi → stdlib
- factorial → factorial
- printf → stdio

factorial関数

外部関数: fprintf, atoi, printf
ユーザー定義の関数： factorial

注: =は関数ではない。return, if, for, whileもそう

```
1:  #include <stdio.h>
2:  #include <stdlib.h>
3:
4:  int factorial(int n);
5:
6:  int main(int argc, char *argv[])
7:  {
8:      int x, n;
9:
10:     if (argc < 2) {
11:         fprintf(stderr, “Usage: %s number¥n”, argv[0]);
12:         return 1;
13:     }
14:     n = atoi(argv[1]);
15:     x = factorial(n);
16:     printf(“%d! = %d¥n”, n, x);
17:     return 0;
18: }
19:
20: int factorial(int n)
21: {
22:     int  i,  x = 1;
23:
24:     for (i = 2; i <= n; i++)
25:         x *= i;
26:     return x;
27: }
```

5. 関数の構造

```
 6: int main(int argc, char *argv[])
 7: {
 8:     int x, n;
 9:
10:     if (argc < 2) {
11:         fprintf(stderr, “Usage: %s number¥n”,argv[0]);
12:         return 1;
13:     }
14:     n = atoi(argv[1]);
15:     x = factorial(n);
16:     printf(“%d! = %d¥n”, n, x);
17:     return 0;
18: }
```

関数の定義・宣言

- 関数の型
- 関数名
- 引数の宣言

関数本体

全体を波括弧{}でくくる

- 変数宣言
- ブロックまたは実行文
  文の終わりは；
- return 文で値を返す

上から順に文を実行

6. Cプログラムは文字で作る

プログラミングには、キーボードから文字を入力

テキストエディタを用いる: TeraPad, 秀丸、Emacs, Vi, …

プログラム開発環境: Visual Studio, Eclipse, …

プログラムを素早く正確にタイピングする能力

はとても大事

7. Cは英語とは違う

でも、英語の省略形をよく使っているので英語が分かっていると、プログラムもわかりやすい

¥n --- nは newline(改行)の頭文字

¥t --- tは tab (タブ)

h --- header (ヘッダ)

io --- input/output (入力・出力)

lib --- library(ライブラリ)

std --- standard (標準)

int --- integer (整数)

float --- floating point number (浮動小数点数)

stdio.h = standard input/output header
stdlib.h = standard library header

8. プログラムの音読のすすめ

実際に読んでみましょう（注:カッコ、コッカは「開きカッコ」「閉じカッコ」の略）

```
# include<stdio.h>

int main(void)
{
        printf("Hello, world!¥n");

        return 0;
}
```

シャープ　インクルード　かぎカッコ
スタンダード・アイ・オー　ピリオド　エイチ　かぎコッカ

イント　メイン　カッコ　ボイド　コッカ

波カッコ（中カッコ）

プリントエフ　カッコ　ダブルクォーテーション
ヘロー　コンマ　ワールド　エクスクラメーション
円マーク　エヌ　ダブルクォーテーション
コッカ　セミコロン

リターン　ゼロ　セミコロン

波コッカ（中コッカ）

9. Cの単語の読み方

読んでみましょう -----　元の単語を書いてみましょう

stdio.h

stdlib.h

int

argc

argv

fprintf

atoi

printf

ヒント: argument counter vector

ヒント: ascii to integer

記号に慣れよう

一つで意味を持つもの

＃　，　＊　；＜　￥　＝　＋＋　--　*＝　＋＝　-＝　など

２つで意味を持つもの(いろいろな括弧に意味あり)

＜…＞　　（…）　　［…］　　｛…｝

括弧以外にも:　　"…"　　'…'

読めるかな？

!=

10. 英字の省略形に慣れよう

printf --- fprintf sprintf

print formatted

f: file

s: string

scanf --- fscanf sscanf

scan formatted

stdio --- stdin

standard input output

stdout

stderr

stdlib

11. インデント(字下げ)を理解しよう

1）カッコで囲まれた範囲が一目で分かるようにする
2) 制御文の範囲が一目で分かるようにする

インデントのルール
1) { が始まると、1段下げる
2) } で閉じると、1段戻る
3) for, while, if, else で制御される範囲は { } がなくても1段下げる
　　その次の行は1段戻る

インデントにはいくつかの流儀があり、どれが一番とは言えません。
自分にあった流儀を身につけてください (でも必要!)

練習：このプログラムを読んでみよう。読みにくいところはあるか？

```
 1:   #include <stdio.h>
 2:   #include <stdlib.h>
 3:
 4:  int factorial(int n);
 5:
 6:  int main(int argc, char *argv[])
 7: {
 8:       int x, n;
 9:
10:       if (argc < 2) {
11:           fprintf(stderr, “Usage: %s number¥n”, argv[0]);
12:           return 1;
13:       }
14:       n = atoi(argv[1]);
15:       x = factorial(n);
16:       printf(“%d! = %d¥n”, n, x);
17:       return 0;
18: }
19:
20:  int factorial(int n)
21:  {
22:       int  i,  x = 1;
23:
24:       for (i = 2; i <= n; i++)
25:           x *= i;
26:      return x;
27:   }
```

練習：このプログラムの「意味」をとって読んでみよう

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  int factorial(int n);
 5:
 6:  int main(int argc, char *argv[])
 7: {
 8:      int x, n;
 9:
10:      if (argc < 2) {
11:          fprintf(stderr, “Usage: %s number¥n”, argv[0]);
12:          return 1;
13:      }
14:      n = atoi(argv[1]);
15:      x = factorial(n);
16:      printf(“%d! = %d¥n”, n, x);
17:      return 0;
18: }
19:
20:  int factorial(int n)
21:  {
22:      int  i,  x = 1;
23:
24:      for (i = 2; i <= n; i++)
25:          x *= i;
26:      return x;
27:   }
```

標準入力と標準ライブラリのヘッダファイルを取り込む

整数の引数を取り、値として整数を返す関数factorialを宣言

整数の引数argcと文字配列argvを引数とするmain関数を定義

整数型変数xとnを宣言

argcの値が2未満なら

標準エラー出力(stderr)に引数の個数が足りないとして使用法を表示し, エラー番号1を返す

mainの引数を数に変換して変数nに代入
関数factorialにこの値を与えて計算、その結果をxに代入

nの階乗の値が x という表示を行い、正常終了

整数の引数を取り、値として整数を返す関数factorialを定義

整数型の変数iとxを宣言、xには初期値として1を代入

iが2からnまで繰り返しxにiを掛け算した結果をxに代入

xの値を関数factorialの戻り値とする

演習問題1.

プログラミングに使われる以下の単語の『英語訳』(本当は、英語の単語が先にあって、日本語に訳されたのですが…)を答えよ

1. 引数
2. 戻り値(返却値)
3. 変数
4. 整数
5. 文字
6. 文字列
7. 配列

演習問題2.

以下の単語の読み方とCプログラミングにおける意味を書け

1. text　　解答例：テキスト　（文字で書かれたデータ）
2. editor
3. standard
4. input
5. output
6. print
7. format
8. comment
9. function

練習問題3. 次のプログラムを「読んでみよ」

```
#include <stdio.h>

int main(void) {
    int i,j;
    printf("input two numbers :");
    scanf("%d %d", &i, &j);
    printf("%d * %d = %d¥n", i, j, i*j);
    return 0;
}
```