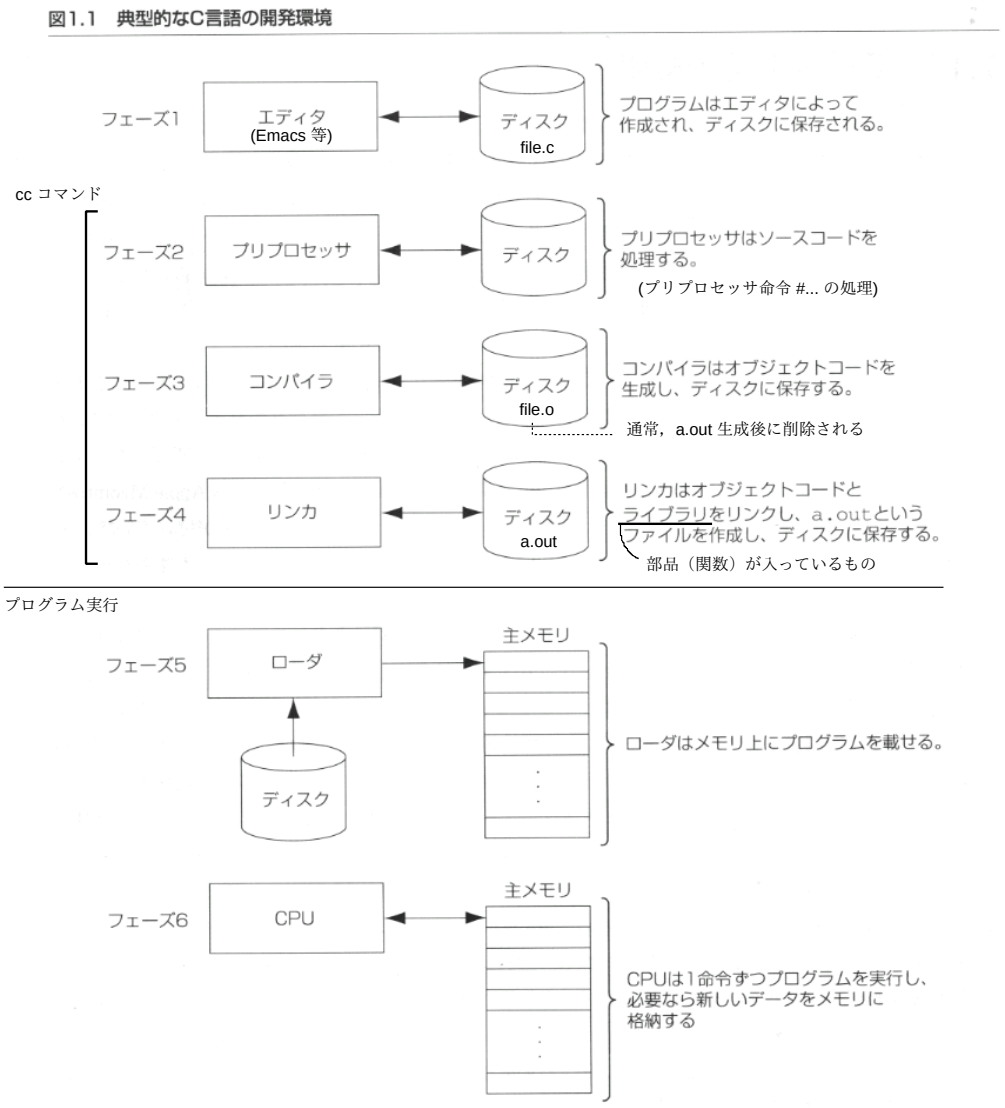
6 関数の概要

6.1 C のプログラムモジュール

- 小さな部品 (モジュール) からプログラムを構築すると管理が容易になる
- 関数: C のプログラムモジュール
  - C ではユーザー定義関数とライブラリ関数を組み合わせてプログラムを作る
  - printf や scanf, getchar, putchar 等は C 言語の処理系に予め用意されている関数であり，これらを “標準ライブラリ関数” という。

6.2 プログラムの開発・実行過程とエラーメッセージ



- gcc コマンドの引数エラー (指定したソースファイルが存在しない)

```
[konno@as ~/src_err_example]$ gcc nofile.c
gcc: nofile.c: そのようなファイルやディレクトリはありません
gcc: no input files
[konno@as ~/src_err_example]$ ls
compile_error.c  include_error.c  link_error.c
```

- プリプロセッサのエラーメッセージ (#include するファイル名の誤り)

```
[konno@as ~/src_err_example]$ gcc include_error.c
include_error.c:2:20: error: studio.h: そのようなファイルやディレクトリはありません
include_error.c: In function ‘main’:
include_error.c:6: 警告: incompatible implicit declaration of built-in function ‘printf’

[konno@as ~/src_err_example]$ cat include_error.c
#include <studio.h>

main()
{
     printf("Hello World\n");

     return 0;
}
```

- コンパイラのエラーメッセージ (構文エラー)

```
[konno@as ~/src_err_example]$ gcc compile_error.c
compile_error.c: In function ‘main’:
compile_error.c:8: error: expected ‘;’ before ‘}’ token
[konno@as ~/src_err_example]$ cat compile_error.c
#include <stdio.h>

main()
{
     printf("Hello World\n");

     return 0
}
```

- リンカのエラーメッセージ (関数名の誤り)

```
[konno@as ~/src_err_example]$ gcc link_error.c
```

```
/tmp/ccQwiqdO.o: In function ‘main’:
link_error.c:(.text+0xf): undefined reference to ‘print’
collect2: ld はステータス 1 で終了しました
[konno@as ~/src_err_example]$ cat link_error.c
#include <stdio.h>

main()
{
     print("Hello World\n");

     return 0;
}
```

6.3 標準ライブラリ関数の利用

6.3.1 数学ライブラリ関数 — 数学計算を行うための標準ライブラリ関数群

■問題 年利 5 % の普通預金口座に 1000 ドル入っている。今後 10 年間にわたる毎年の残高を次式で計算し表示せよ。

$a = p\,(1 + r)^n$, p : 元金, r : 年利率, n : 年数

■リスト 6.1 **標準ライブラリ関数** pow **を使った複利計算**

```
 /* 複利計算 */
#include <stdio.h>
#include <math.h>  /* 数学ライブラリ関数を使うときに必要 */

int main()
{
  int year;
  double amount;               /* 預金残高（元利合計） */
  double principal = 1000.0;   /* 元金 */
  double rate = .05;           /* 年利率 */

  printf("%4s%21s\n", "Year", "Amount of deposit");

  for (year = 1; year <= 10; year++) {
    amount = principal * pow(1.0 + rate, year);
    printf("%4d%21.2f\n", year, amount);
  }
```

```
  return 0;
}
```

実行結果

```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

- `#include <math.h>`
  - 数学ライブラリ関数を使うときに必要
- `printf("%4s%21s\n", "Year", "Amount of deposit");`
  - %21s は，文字列を (半角) 21 文字分の幅で右寄せに表示する
- `amount = principal * pow(1.0 + rate, year);`
  - pow(x,y) は x の y 乗を求める数学標準ライブラリ関数
  - 数学ライブラリ関数では，それらの引数（関数名の右の括弧内の値）と戻り値（計算結果）は必ず double 型
  - UNIX 系 OS では，数学ライブラリ関数を含むソースファイルをコンパイルする時に，通常，コマンド行オプション `-lm` が必要。すなわち，

    `cc file -lm` または `gcc file -lm`

    としなければ実行可能ファイル (a.out) はできあがらない。
- `printf("%4d%21.2f\n", year, amount);`
  - %21.2f は，浮動小数点型 (実数) の数値の小数点以下第 2 位までを，21 文字分の幅で右寄せに表示する。

6.3.2 ライブラリ関数の利用

- 関数呼び出し: 関数を使う（起動する）こと
  - 書式： **関数名（実引数）**
    - 実引数 (argument) が複数ある場合には，コンマで区切る
  - 例）`printf("%.2f", sqrt(900.0));`
    - 関数 `sqrt` の呼び出し
    - sqrt は実引数に与えれらた数の平方根を返す数学ライブラリ関数

  * 関数が返す値を，関数の戻り値 (return value) という。
  * 関数の実引数には，定数，変数，式，が使える
    - 例えば `printf("%.2f", sqrt(c1 + d * f));` も可能
- 関数の実引数の個数や戻り値と有無，および，それらのデータ型は関数毎に定められている。
- UNIX の man コマンドで C の標準ライブラリ関数を調べることができる
  * UNIX のコマンド行から `man pow` を実行すると

    ```
    double pow(double x, double y);
    ```

    と記述がある。これより関数 pow の呼び出しには double 型 の実引数 2 つが必要であり，戻り値（計算結果）は double 型であることがわかる。なお，x と y に特に意味はない。「x の y 乗を計算する。」のような関数 pow の機能説明のために書いてあるだけ。
  * man getchar の

    ```
    int getchar(void);
    ```

    からは, getchar には実引数が無く (void)，戻り値が int 型であること，および

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

    が必要であることもわかる。

6.4 関数

■リスト 6.2 **ユーザー定義関数を含むプログラム例（動作説明用）**

```
/* パラメータが無く，戻り値を使わないプログラム例 */
/* 関数のプロトタイプ宣言を書くのが望ましいが省略した */
#include <stdio.h>

int main()
{
    int i;

    for (i = 1; i <= 2; i++)
       printline();

    return 0;
}

int printline()
{
    int i;

    for (i = 1; i <= 5; i++)
       printf("*");
    printf("\n");
```

```
    return 0;
}
```

実行結果

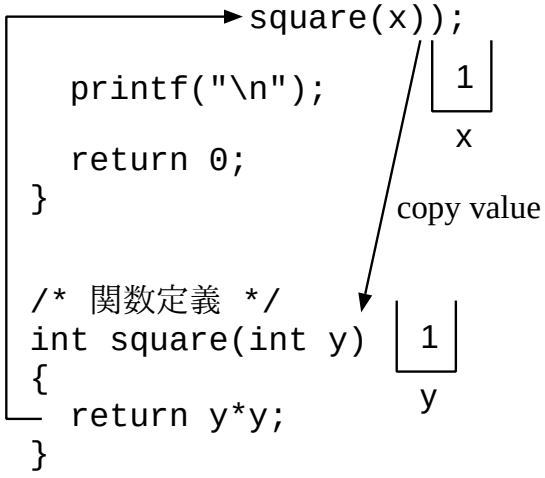
```
*****
*****
```

- 関数内で宣言された変数は局所変数である
  - 宣言が行われた関数内でしか使えない
- 関数を使ってプログラムを作ることの意義
  - 関数はプログラムをモジュール化する
    - プログラムの管理を容易にする
  - ソフトウェアの再利用性
    - 新しいプログラムを書くときに，作成済みの関数を部品として使える
    - 抽象化 — プログラム内部での処理の詳細を隠す
  - 同じコードの繰り返しを避ける
- 「C を学ぶ」= 次の二つを学ぶこと
  1. C 言語自体
  2. C 標準ライブラリにある様々な関数の使い方
- プログラムを作成するときには
  - C の標準ライブラリ関数を使う
    - プログラムの構成部品として利用
  - プログラマは自分自身で関数を作ることができる
    - 利点: 関数の動作を完全に把握できる
    - 欠点: 時間がかかる
  - 利用可能な既存の関数があれば，それを使うべき
    - ライブラリ関数は効率良く動くように，注意深く書かれている
- main はユーザー定義関数である
  - C プログラムは一つ以上の関数から成り，そのうちの一つは main でなければならない
  - C プログラムの実行は関数 main から始まる

6.5 関数定義

■リスト 6.3　1 **から** 10 **までの整数の** 2 **乗を表示するプログラム**

```
/* プログラマ定義関数 square */
#include <stdio.h>

int square(int);  /* 関数プロトタイプ */

main()
{
  int x;

  for (x = 1; x <= 10; x++)
    printf("%d  ",
           square(x));

  printf("\n");

  return 0;
}


/* 関数定義 */
int square(int y)
{
  return y*y;
}
```



実行結果

```
1 4 9 16 25 36 49 64 81 100
```

- 関数定義の書式

  戻り値の型　関数名（**パラメータリスト**）
  {
  　宣言文

  　実行文
  }

  – 関数名: 任意の識別子 (変数名と同じルールでつけて良い)
  – 戻り値 (リターン値) の型: 関数が呼び出し側に返すデータの型
  　＊ デフォルトは (何も書かなければ) int 型
  　＊ `void` — 関数が何も返さないことを表す
  – パラメータリスト: コンマで区切ったパラメータ（仮引数）のリスト

    * パラメータ毎に型を明示する (例えば `float x, y` ではなく, `float x, float y` とする)
    * `void` — 関数がパラメータ（引数）を通じてデータを受け取らないことを表す
  - 宣言文と実行文: 関数の本体 (ブロック)
    * ブロック内では変数を宣言することができる
      (ブロックを入れ子にすることも可能)
    * 関数内部で他の関数を定義することはできない
  - 呼び出し側に制御を戻す (関数の実行を終了する) 方法
    * 戻り値を返さない場合
      · `return;` の実行
      · 関数本体の閉じブレース (}) に到達
    * 戻り値を返す場合
      · `return` **式**`;` — 式の値が呼び出し側に返される

■リスト 6.4 **複数の引数を持つ関数定義の例**

```
/* 3 つの整数の中から最大値を見つける */
#include <stdio.h>

int maximum(int, int, int);  /* 関数プロトタイプ宣言 */

int main()
{
  int a, b, c;

  printf("Enter three integers: ");
  scanf("%d%d%d", &a, &b, &c);
  printf("Maximum is: %d\n", maximum(a, b, c));

  return 0;
}


/* 関数 maximum の定義 */
int maximum(int x, int y, int z)
{
  int max = x;

  if (y > max)
    max = y;
```

```
  if (z > max)
    max = z;

  return max;
}
```

実行結果例 1

```
Enter three integers: 5 15 10
Maximum is: 15
```

実行結果例 2

```
Enter three integers: 10 5 15
Maximum is: 15
```

実行結果例 3

```
Enter three integers: 15 10 5
Maximum is: 15
```

6.6 プロトタイプ宣言

- 関数のプロトタイプ宣言
  - 構成要素
    - 関数名
    - パラメータ — どの型のデータを何個受け取るか
    - 戻り値の型 — 関数が返すデータの型 (デフォルトは int 型)
  - 関数呼び出しの際に，関数に与えるパラメータの型と個数および戻り値の型は，すべて関数定義と同じであることが原則。
  - 関数呼び出しが正しく行われているかをチェックするために使われる — 関数定義とプロトタイプ宣言の型が一致しなければ構文エラー
  - 例:

    ```
    int maximum(int, int, int);
    ```

    - 3 つの int 型引数を受け取る
    - int 型のデータを返す (戻り値は int 型)
  - `int maximum(int x, int y, int z);` と書いてもよい
  - 関数呼び出しよりも後ろで関数定義が行われている場合や，別のソースファイルまたはライブラリに関数が存在する場合に必要— 関数呼び出しの前にプロトタイプ宣言するべき。ただし，ライブラリ関数の場合には `#include` によってプロトタイプ宣言するので，プロトタイプ宣言を自分のソースには書かない。
- データ型の自動変換
  - 関数のプロトタイプ宣言があれば，コンパイラは，関数呼び出し時に実引数の型をプロトタイプ宣

言に記述した型に変換して，関数に渡す
  * 例:
    ```
    printf("%.3f\n", sqrt(4));
    ```
    sqrt の整数型の引数 4 は，double 型の 4.0 に変換されて sqrt に渡される
- 注意) 整数型のパラメータをもつ関数に実数を渡すと，小数点以下が切り捨てられる（数値の誤差が生じる）
  * list 6.3 で定義された関数 square (パラメータは int 型) に対して
    ```
    square(4.5)
    ```
    とすると，square は 4 を受け取る

6.7 ヘッダファイル

- ヘッダファイル (header file)
  - 標準ライブラリ関数の関数プロトタイプが書いてある
  - stdio.h や math.h はヘッダファイル
  - 標準ライブラリ関数を利用するソースファイルでは，関数を呼び出す部分より前に
    ```
    #include <headerfile>
    ```
    を書いて *headerfile* を取り込むことで，プロトタイプ宣言を行わなければならない。
- 自作のヘッダファイルを作って使うことができる
  - ユーザー定義関数の関数プロトタイプが入ったヘッダファイルを *filename*.h という名前で作る
  - その関数を使うソースファイルで
    ```
    #include "filename.h"
    ```
    と書く

7 補足： 標準入出力を用いた C プログラムによるファイル入出力

- scanf や printf は，データの入出力を標準入力 (stdin) や標準出力 (stdout) に対して行う。
- 標準入出力の接続先は，キーボードや画面から別のデバイス（ファイル等）に切り替えできる。
- 例えば，`./a.out > tmp.out` を実行すれば，a.out の出力先が画面からファイル tmp.out に切り替わり，出力は tmp.out に入る（シェルによる標準出力のリダイレクト）。

参考文献

[1] ハーベイ・M・ダイテル，ポール・J・ダイテル, C 言語プログラミング. ピアソン・エデュケーション, 2004.

[2] H. M. Deitel, P. J. Deitel, C How to Program Third Ed. Prentice Hall, 2001.

[3] 柴田 望洋，新版明解 C 言語入門編. ソフトバンククリエイティブ, 2004.

[4] 財団法人日本情報処理開発協会 監修，情報処理技術者テキスト プログラミング入門 C 言語. 実教出版, 2005.