Programming I 0x0b

関数の基礎（2010.06.14）

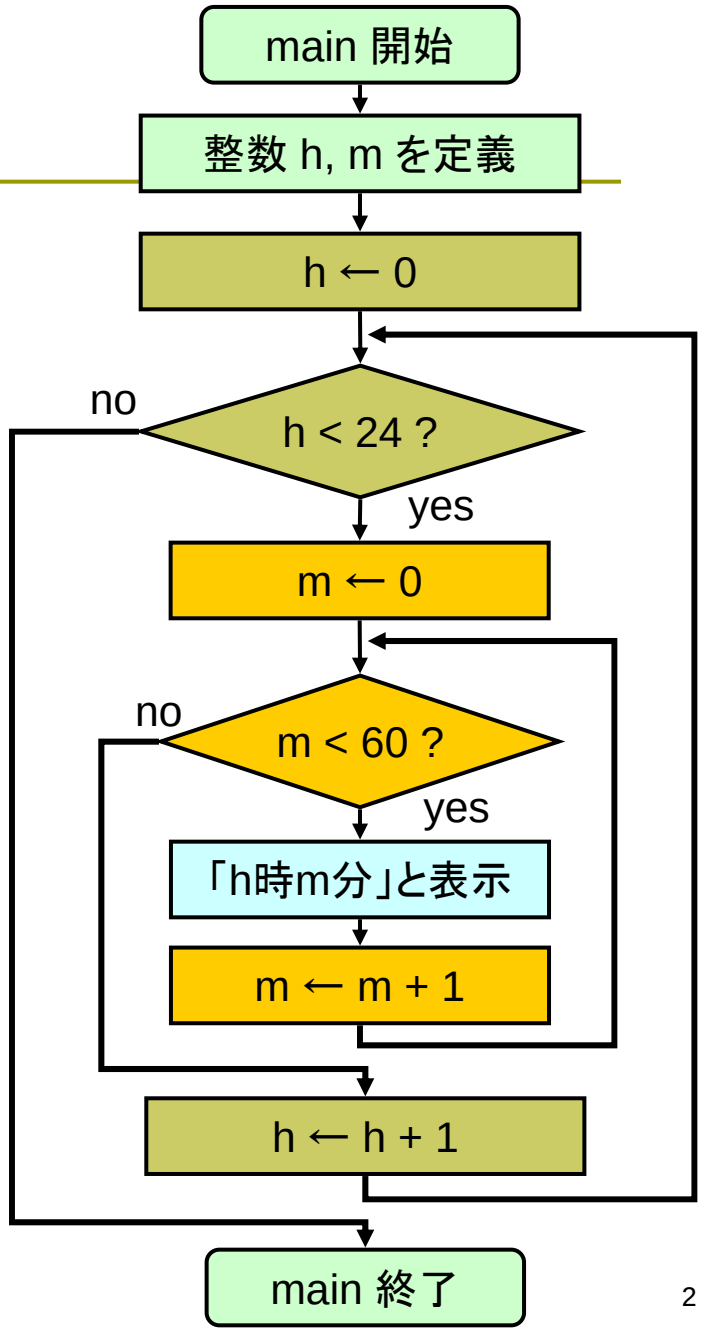
塩澤秀和 http://vilab.org

2重ループ（復習）

```
#include <stdio.h>

int main(void)
{
  int h;   /* 時 0～23 */
  int m;   /* 分 0～59 */

  for (h = 0; h < 24; h++) {
    for (m = 0; m < 60; m++) {
      printf("%d時%d分¥n", h, m);
    }
    getchar(); /* [Enter]キーを待つ */
  }
  return 0;
}
```



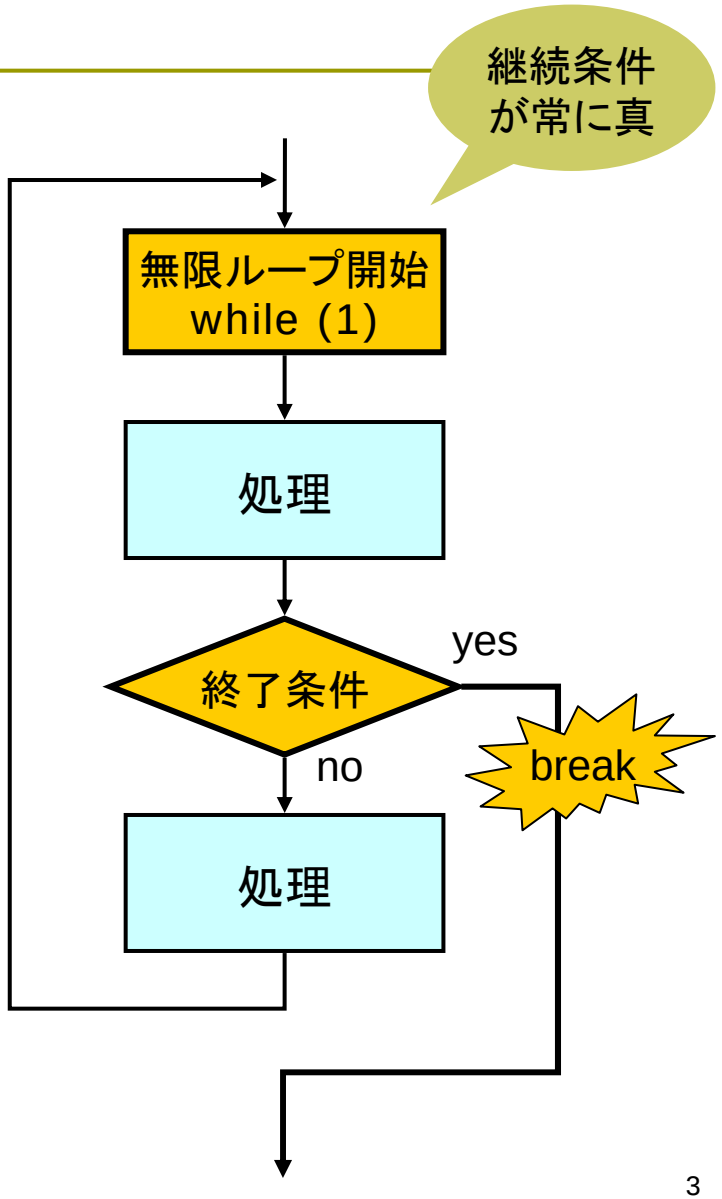
無限ループ（復習）

- 無限（永久）ループ
  - わざと無限ループを作る方法

```
while (1) {
   ...
}
```

```
for (;;) {
   ...
}
```

  - 本当に終わらないループを作ってしまうとまずいので…
- break文（p.58）
  - ループを中止して外に抜ける
    if (終了条件) break;
  - ループの途中で終了（継続）を判定したいときに使う



その他の制御構造（復習）

- do-while文（p.59）
  - ループの処理の最後に、継続条件を判定する
  - 判定が最後なので、最低1回はループの中を通る
  - うまく使えば分かりやすいが、必要ないときには使わない
- switch-case文（p.65）
  - 整数値での多重分岐を“見やすく”書くための構文
  - if ～ else if ～...～ else で書いても本質的な違いはない
- goto文（p.68）
  - **使わない！**（適切な理由がある場合を除く）
  - 1968年の有名な論文『GO TO文は有害だと考えられる』

エドガー・ダイクストラ

“関数”を使ったプログラム

- 関数＝プログラムの部品
  - よく使う手順を部品にする
  - 部品（関数）をmainで利用

```
#include <stdio.h>

int main(void)
{
  printf("痛恨の一撃！¥n");
  printf("痛恨の一撃！¥n");
  printf("痛恨の一撃！¥n");
  return 0;
}
```

関数を使う前のプログラム

部品化

本体

関数を使ったプログラム

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

関数

```
void tsukon(void)
{
  printf("痛恨の一撃！¥n");
}
```

```
int main(void)
{
  tsukon();
  tsukon();
  tsukon();
  return 0;
}
```

回数を変えてみよう

簡単な関数の作りかた

- 関数の定義
  - 定義はmainの外に書く

```
void 関数名(void)
{
    処理手順
    ...
}
```

※voidの意味は後で説明する

- 関数の“呼び出し”
  - mainの中で関数を使う

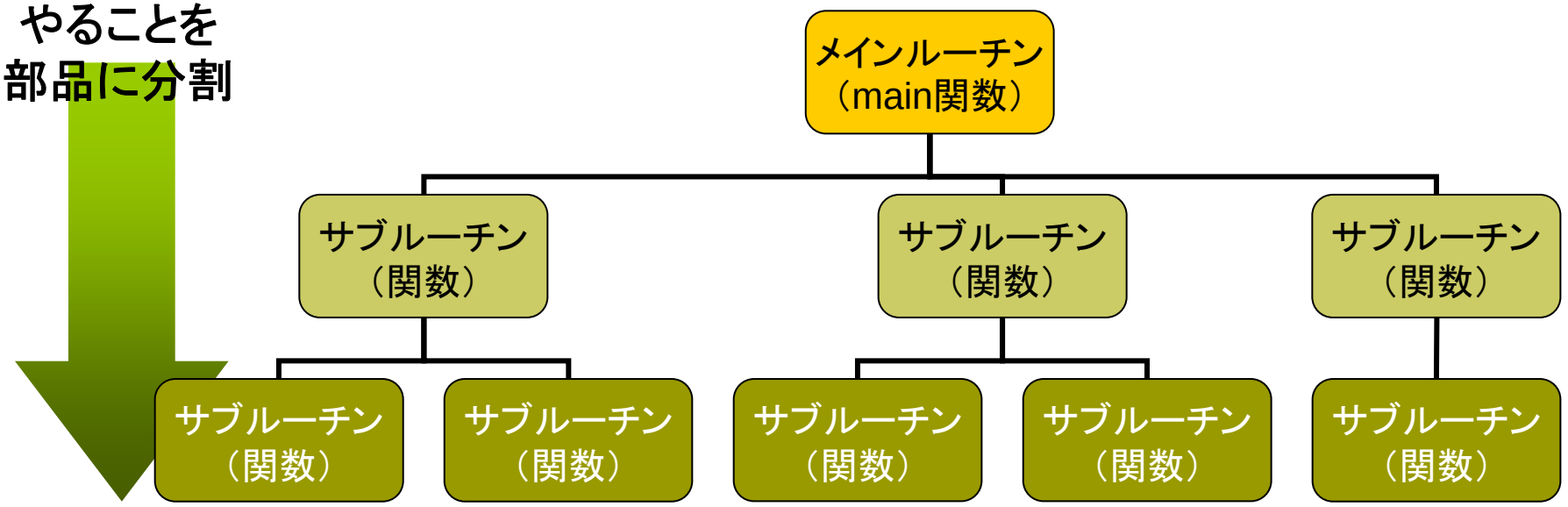
```
int main(void)
{
    ...

    関数名();
    ...
}
```

- 関数が動くしくみ
  - プログラムがスタートすると、**必ずmainから実行を始める**
  - mainの中で関数が出てくると、その関数に実行が移る
  - 関数の処理が終わると、mainに戻ってきて続きをやる

C言語の関数の役割

- 機能分割によるプログラム開発
  - 複雑なプログラムを、小さな部品（機能）に分けて開発する
    - 英語ではfunction（関数）に、「機能」という意味もある
  - 本体＝「メインルーチン」「主プログラム」
  - 部品＝「サブルーチン」「副プログラム」「関数」「手続き」



変数の通用範囲

```
#include <stdio.h>

int main(void)
{
  int dp; /* 変数定義 */

  printf("ダメージ ->");
  scanf("%d", &dp);

  printf("会心の一撃！¥n");
  printf("%dのダメージ¥n", dp);

  return 0;
}
```

ここの部分を関数にしても…

```
#include <stdio.h>

void kaishin(void)
{
  printf("会心の一撃！¥n");
  printf("%dのダメージ¥n", dp);
}

int main(void)
{
  int dp; /* 変数定義 */

  printf("ダメージ ->");
  scanf("%d", &dp);

  kaishin();
  return 0;
}
```

コンパイルエラー!!

- 局所変数（ローカル変数）
  - 通常の変数は、定義されている { } の外には通用しない

関数の引数（ひきすう）

- 関数にデータを渡す方法
  - 関数呼び出しの( )のなかに、データ（式）を並べる
    関数名(式); ← 2つ以上なら「関数名(式, 式, …);」
  - 関数に受け渡しするデータを“引数（ひきすう）”という

- 引数をとる関数の定義
  - 関数定義の( )のなかに、データを受け取る変数を定義

```
void 関数名(データ型 変数, …)
{
    変数を使ったプログラム
}
```

double x, int a
などいくつでも

※ voidは空（なし）という意味

  - 引数の変数は関数内でのみ通用する（ローカル変数）

関数にデータを渡す

```
#include <stdio.h>

/* 関数定義 */
void buy(int price)
{

   /* データを使った処理 */
   printf("まいどあり～¥n");
   printf("値段は%d円です¥n",
          price);
}
```

```
int main(void)
{
   int yen;

   /* 関数にデータを渡す */
   buy(1480);

   /* 変数で渡す */
   yen = 4980;
   buy(yen);
   return 0;
}
```

仮引数
（値をもらう変数）

受け渡し
price=1480

受け渡し
price=yen

実引数
（関数に渡す値）

関数呼び出し
（仮引数＝実引数）

関数プロトタイプ宣言

```
#include <stdio.h>

/* 関数プロトタイプ宣言 */
void ask(int hira);
void starbar(int n);

int main(void)
{
  int n;

  ask(0);
  scanf("%d", &n);
  starbar(n);
  return 0;
}

void ask(int hira)
{
  if (hira)
    printf("ほしのかずは？ ");
  else
    printf("星の数は？ ");
}

void starbar(int n)
{
  int i;

  for (i = 0; i < n; i++)
    printf("★");
  printf("¥n");
}
```

プロトタイプ宣言の作りかた
「関数定義の1行目」+「;」

関数を使う場所より後ろに
関数の定義を書くとき必要

関数本体
の定義

関数の最初
の使用

演習問題

11a. 自分の名前を表示する myname という関数と、出身地を表示する hometown という関数を別々に定義し、mainのなかで順に呼び出して表示するプログラムを作成しなさい。

11b.「関数を使ったプログラム」で、tsukon関数をmainの後ろに移動し、関数プロトタイプ宣言を使うように変更しなさい。

11c.「変数の通用範囲」のプログラムを、kaishin関数がint型の引数をとるように修正して、うまく動くようにしなさい。

11d. 月と日を表す整数を引数にとり、標準出力に「○月×日」と表示する関数 print_date を作成しなさい。

- mainのなかで print_date(6, 14) などとして、動作を確認しなさい。
- 関数のプロトタイプ宣言は void print_date(int month, int day);

- 次回までの課題：**リスト5-1～5-6**を入力して教科書を予習