Programming II 0x02

関数の復習（2011.04.21）

塩澤秀和 http://vilab.org

“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>

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

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

関数

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

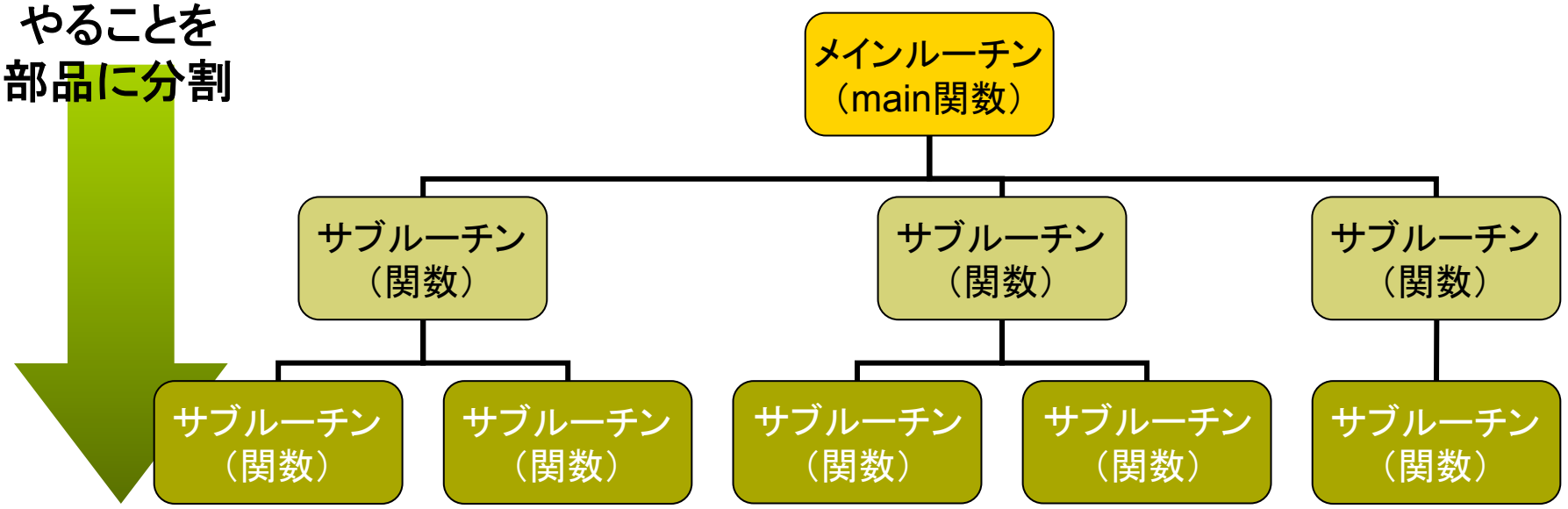
```
int main(void)
{
    ...

    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

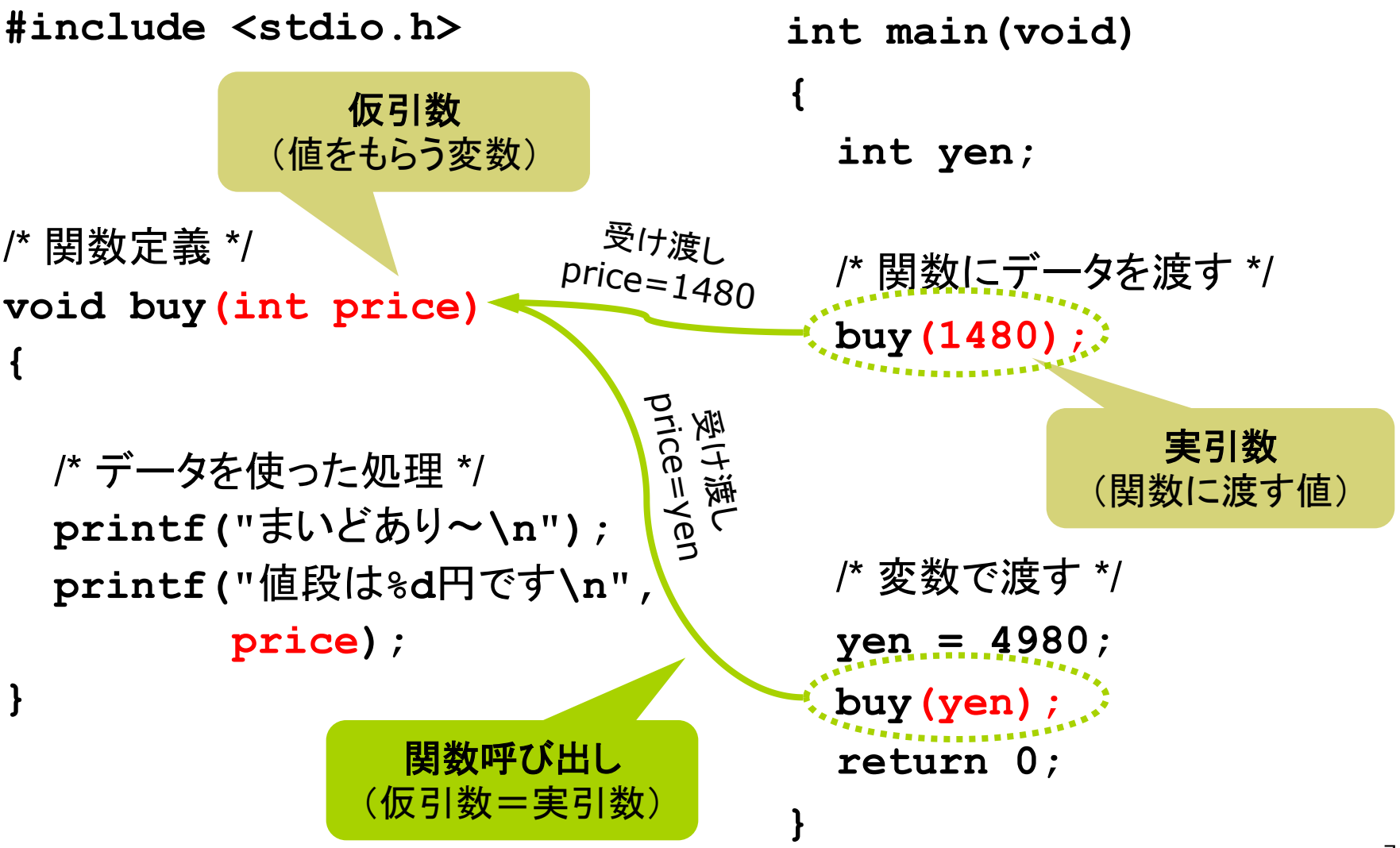
```
void 関数名 (データ型 変数, …)
{
    変数を使ったプログラム
}
```

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>
仮引数
（値をもらう変数）
/* 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);
}
受け渡し
price=1480
受け渡し
price=yen
関数呼び出し
（仮引数＝実引数）
int main(void)
{
int yen;
/* 関数にデータを渡す */
buy(1480);
実引数
（関数に渡す値）
/* 変数で渡す */
yen = 4980;
buy(yen);
return 0;
}

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

関数の最初の使用

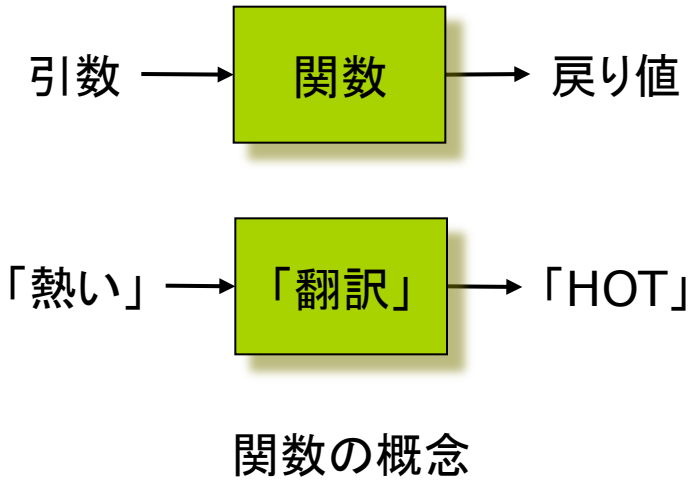
関数本体の定義

関数の戻り値(p.74)

- 関数からデータを渡す方法
  - 計算結果などの値を、関数から呼び出し元(main)に渡す
  - “引数”とは逆方向 ＝ “戻り値”、“返り値”、“返却値”

```
戻り値の型　関数名 (引数並び)
{
    計算などの手順
    ...
    return　戻り値;
}
```

戻り値の式



関数の概念

- return文(p.76)
  - 関数を終了して、呼び出し元にすぐ戻る
  - 戻り値がある関数の場合、値を“戻す”ために絶対に必要

引数と戻り値

```
#include <stdio.h>

int plus(int x, int y)
{
   int z;

   z = x + y;

   return z;
}

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

   printf("a b -> ");
   scanf("%d %d", &a, &b);

   c = plus (a, b);

   printf("c = %d\n", c);

   return 0;
}
```

戻り値のデータ型

引数（関数に渡す値）

戻り値（関数から返す値）

計算を関数に分ける

```c
#include <stdio.h>

int main(void)
{
  double pi = 3.1416;
  double r, s;

  r = 5.0;
  s = pi * r * r;
  printf("%.4f\n", s);
  return 0;
}
```

本体

部品化

```c
#include <stdio.h>
double menseki(double r);

int main(void)
{
  double r, s;

  r = 5.0;
  s = menseki(r);
  printf("%.4f\n", s);
  return 0;
}

double menseki(double r)
{
  double pi = 3.1416;
  double s;

  s = pi * r * r;
  return s;
}
```

プロトタイプ宣言

ここではprintfしない!!

- 数学のような“関数”
  - 「値を返す」「値を戻す」

⇒ 結果は**表示(printf)せず**に“戻り値”として返す(return)

演習問題

- ソースファイルの名前は「課題番号.c」（例：2a.c）とすること

2a. 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。

2b.「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。

2c.「計算を関数に分ける」のプログラムを三角形の面積を計算するように改造しなさい。menseki は底辺と高さという2つの引数をとる関数に改造し、mainも修正する必要がある。

2d. 数学の表現で f(x) = 2x + 1 となる関数をC言語で作成し、mainで呼び出して f(0), f(1), ..., f(10) の値を表示させなさい。

課題提出（アップロード）

- 提出用Webページ
  - http://vilab.org/upload/c2-upload.html
  - フォームにクラス・出席番号等を入力
  - 参照でソースファイルを選択
    - （マイ）ドキュメント → Visual Studio 2008 → Projects → プロジェクト名 → プロジェクト名 → ファイル名.c
  - [送信]ボタンを押して提出
    - プログラムのファイル（ソースファイル）が画面に表示されれば、ちゃんと提出されたはず

- 次回までの課題
  - 教科書の配列のところ（p.101～p.105）を予習しておく
  - 例示されているプログラムを入力して動作を確認しておく