情報処理演習
(5)関数　その1

知能システム学　准教授
万 偉偉（ワン ウェイウェイ）

今までの内容

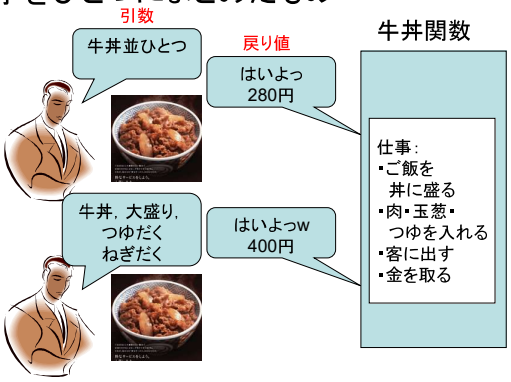
- 変数、基本の書き方
- 条件分岐
- 繰り返し
- 条件分岐と繰り返しの組合せ

- プログラムが大きくなった
- 重複が多くなった
- 関数を活用しましょう

関数とは

- 一連の仕事をひとつにまとめたもの

- 簡単な指示で複雑な仕事をさせることができる
- 指示を出すほうは作業方法を知らなくてもよい
- あちこちから何度でも同じ仕事を依頼できる
- 少し違った要求ならこたえることが出来る



関数の例

- 整数を自乗する関数

```
#include <stdio.h>

int square (int x);                    関数の宣言（関数の概要の記述）
                                       プロトタイプ宣言　必ず必要
int main(void) {
        int y;

        y = square(5);
        printf("result is %d¥n", y);
        return 0;
}

int square (int x) {                   関数の定義
        return x * x;                  （関数そのもの記述）
}
```

関数の「いいところ」

- 関数の中身を知らなくていい
  - 「どういう結果を生じるか」のみ知っていればOK
  - プログラムが見やすく，分かりやすくなる
- 同じ処理を何度も書かなくて良い
  - 同じ処理をするたびに，呼び出せばよい
- 処理を「一人立ち」させることが出来る
  - 関数を独立に作成・デバッグし，完成したものとして提供できる
  - 他人にも簡単に使ってもらえる

関数の「精神」

- ブラックボックス化
  - 関数の中身は必ずしも知らなくてよい
  - 処理は複雑でも，提供する機能は分かりやすい
    - 例えば，sin( ) とか sqrt( ) など
  - 確実に動作する
    - バグ（不具合）がよく除かれている
  - 明示的な結果以外の作用（副作用）がない
    - 他の変数の値が勝手に変わるなど

関数の「仕様」

- どんな「引数」を取るのか
  - 引数の数，型 (int, double, ..)
- どんな「戻り値」を返すのか
  - 戻り値の有無，戻り値の型
  - 戻り値無しの場合は void を使う
- 関数の機能
  - どのような引数を与えるとどのように動作し，どのような結果（画面入出力，戻り値など）をもたらすのか

関数の使い方

- どこからでも呼び出すことが出来る

```
y = sq (i);                          普通の式
printf("val = %d¥n", sq(j) );        関数の引数で
z = sq( sq (2) );                    （同上）
```

- 戻り値を利用することも，使わないのもOK
- 引数の型に注意 (int に double を入れる等)

関数の中身

```
int max (int a, int b);          関数の宣言（関数の概要の記述）

int max (int a, int b) {         ← 引数の名前と型
        int result;              ← 関数内部で変数を定義できる（有効範囲はこの関数内だけ）
        if(a > b) {
                result = a;
        }                        関数の定義（関数そのもの記述）
        else {
                result = b;
        }
        return result;           ← return で関数を抜けることが出来る その際に戻り値を指定できる
}
```

戻り値の型

関数の注意事項

- 関数内で変数を定義したり値を変更しても，呼び出し側には影響しない
- 関数内に宣言した変数は，関数の内部{ }に囲まれた部分でのみ有効である．他の関数内（main()も含む）に同名の変数があっても完全に別のものである
- 呼び出し側に影響を与える3つの方法
  - 戻り値で情報を戻す
  - 「配列」を使う（次回，次々回）
  - ポインタを使う（年末）

呼び出した側の値

```
#include <stdio.h>
void swap( int x, int y);
int main( void )
{
        int x=5;
        int y=3;
        printf( "関数を呼び出す前x=%d, y=%d¥n", x, y );      (5, 3)
        swap( x, y );
        printf( "関数の結果x=%d, y=%d¥n", x, y );            (5, 3)
}

void swap( int x, int y )
{
        int tmp;
        printf( "関数にはいってきた時x=%d, y=%d¥n", x, y );  (5, 3)
        tmp = y; y = x; x = tmp;
        printf( "関数を出る時x=%d, y=%d¥n", x, y );          (3, 5)
}
```

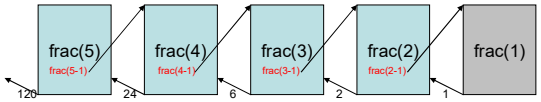
再帰呼び出し

```
int frac(int n) {
  int r;

  if(n == 1)
    r = 1;
  else
    r = n * frac(n-1);

  return r;
}
```

- 関数は自分自身を呼び出すことが出来る

再帰呼び出し

```
int frac(int n) {
  int r;
  if(n == 1)
    r = 1;
  else
    r = n * frac(n-1);
  return r;
}
```



- 呼び出した関数の状態は保持されている
- どこかで再起呼び出しをやめて return しなければならない（終止条件）

標準ライブラリの関数

- printfやscanfも関数→プロトタイプ宣言が必要
- ヘッダーファイルで用意
  - stdio.hやmath.h
  - #includeで読み込み，C言語のソースファイル中に展開し，埋め込む

本日学んだ内容

- 関数
  - 戻り値，引き数，変数型
- プロトタイプ宣言
- 再帰呼び出し