プログラミング基礎演習A

関数の定義

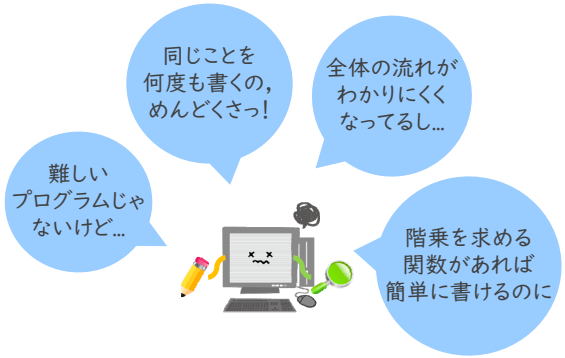
順列${}_nP_r$を求めるプログラム

$${}_nP_r = \frac{n!}{(n-r)!} \qquad a! = 1 \times 2 \times \cdots \times (a-1) \times a$$

```
#include <stdio.h>

int main()
{
   int n, r;
   int i, x, y, npr;

   printf("n:");
   scanf("%d", &n);
   printf("r:");
   scanf("%d", &r);
```

```
   x = 1;
   for(i = 2; i <= n; i++) {
      x *= i;
   }
   y = 1;
   for(i = 2; i <= n - r; i++) {
      y *= i;
   }
   npr = x / y;
   printf("nPr = %d¥n", npr);
}
```


ぱそ子のつぶやき…



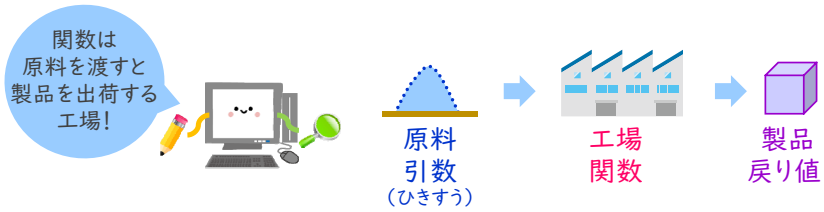
ぱそ子のひらめき！

プログラムにおける関数

- ある処理を実行するための命令のあつまり
- 括弧の中に記述した値を使って，記述された命令通りに実行し，処理後の値を返す

```
x = sqrt(11.25);
printf("Hello, world!¥n");
```

※ 引数と戻り値はないときもある



関数を作る手順

1. 関数の名前を決める
   例）階乗だからfactorialという名前にする

2. 何を渡したら何を返す関数なのかを決める
   引数　　戻り値
   例）factorialは自然数aを渡すと階乗a!を返す関数にする

3. 関数の定義をプログラムとして書く
   例）1からaまでの自然数の積を計算して，得られた積を返すようなプログラムを書く

関数の定義

```
戻り値の型  関数名(型名 引数名, 型名 引数名, …)
{                    第1引数     第2引数
    関数が呼び出されたときに実行される命令
    return 戻り値;
}
```

- 引数がないとき　：丸括弧の中はvoid
- 戻り値がないとき：戻り値の型はvoid
  returnの後には何も書かないか，returnの行自体を省略

関数factorialの定義例と定義を書く位置

```
int factorial(int a)
{
    int fact, i;

    fact = 1;
    for(i = 2; i <= a; i++) {
        fact *= i;
    }
    return fact;
}
```

ここのヘッダファイル読込みの後に書く

メイン関数の実行に必要な情報の記述

```
int main()
{
    実行される命令
}
```

メイン関数

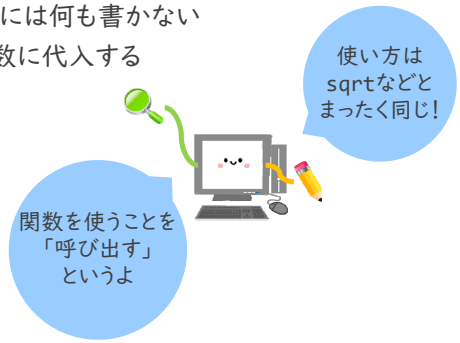
関数の呼出しと戻り値の受取り

関数名(引数, 引数, …)　← 戻り値がないとき/使わないとき

変数 = 関数名(引数, 引数, …)　← 戻り値を使うとき

- 引数がないとき　：　丸括弧の中には何も書かない
- 戻り値を使うとき ：　戻り値を変数に代入する

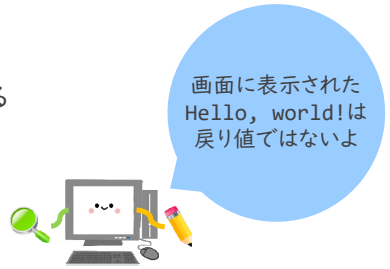
```
x = factorial(n);
y = factorial(n-r);
```



標準関数の引数と戻り値

- sqrt(x)
  - 引数　：　x
  - 戻り値　：　xの平方根
    - sqrt(x)そのものがxの平方根になる
    - ↓
    - printf("%f", sqrt(2.0))と書ける
- printf("Hello, world!¥n");
  - 引数　：　"Hello, world!¥n"
  - 戻り値　：　書き出された文字数

この例では14
この戻り値は普通は使わない



順列${}_nP_r$を求めるプログラムふたたび

$${}_nP_r = \frac{n!}{(n-r)!} \qquad a! = 1 \times 2 \times \cdots \times (a-1) \times a$$

```
#include <stdio.h>


int factorial(int a)
{
   int fact;

   fact = 1;
   for(i = 2; i <= a; i++) {
      fact *= i;
   }
   return fact;
}
```

```
int main()
{
   int n, r;
   int i, npr;

   printf("n:");
   scanf("%d", &n);
   printf("r:");
   scanf("%d", &r);
   npr = factorial(n) / factorial(n-r);
   printf("nPr = %d¥n", npr);
}
```


数学の関数との比較

- 関数$f(x)$の定義
  $f(x) = x^2 + 2x - 5$
- $x = 3$のときの$f(x)$は?
  1. $f(3)$を計算する
     $f(x)$に$x = 3$を代入して計算すると10になる
  2. $f(3)$が10になる

- 関数factorialの定義

```
int factorial(int a)
{
   ...
   return fact;
}
```

- 3の階乗は?
  1. factorial(3)を呼び出す
     factorial(a)をa=3として実行するとfactが6になる
  2. 6を返す
     (factorial(3)が6になる)

関数の実行

関数が呼び出されたら，それまでの実行を中断し，関数の中を実行して，関数の実行が終わったら，中断したところに戻って実行を再開する

```
#include <stdio.h>

void functionX(void)
{
    処理A
}

int main
{
    処理B
    functionX();
    処理C
}
```

実行はここから始まる

処理A～Cの実行順序

処理B → 処理A → 処理C

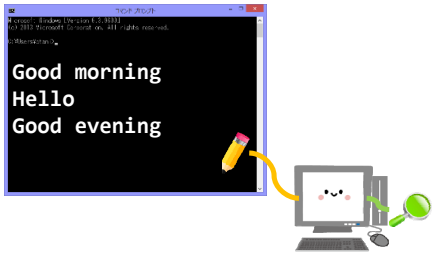
プログラム例(1)

```
#include <stdio.h>

void print_hello(void)
{
    printf("Hello¥n");
}

int main()
{
    printf("Good morning¥n");
    print_hello();
    printf("Good evening¥n");
}
```

戻り値なし

引数なし

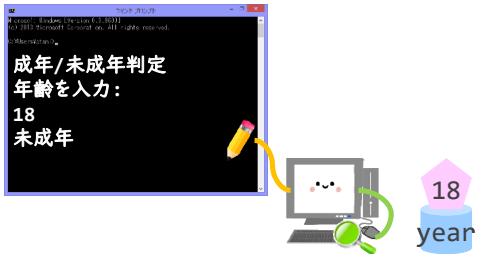
実行はここから始まる



プログラム例(2)

```
#include <stdio.h>

void print_error(void)
{
    printf("入力値がマイナスです¥n");
}
```

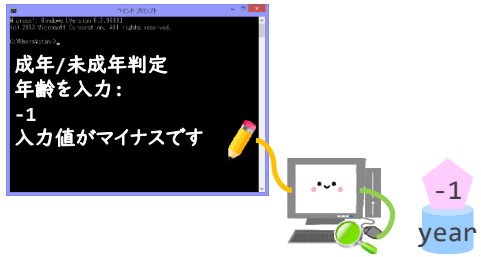
```
int main()
{
    int year;

    printf("成年/未成年判定¥n");
    printf("年齢を入力:¥n");
    scanf("%d", &year);
    if(year >= 20) {
        printf("成年¥n");
    } else if(year >= 0) {
        printf("未成年¥n");
    } else {
        print_error();
    }
}
```



プログラム例(2)

```
#include <stdio.h>

void print_error(void)
{
    printf("入力値がマイナスです¥n");
}
```

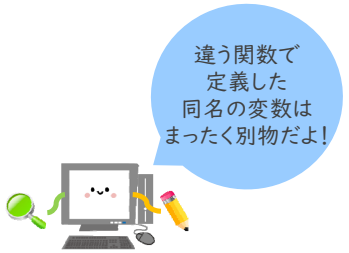
```
int main()
{
    int year;

    printf("成年/未成年判定¥n");
    printf("年齢を入力:¥n");
    scanf("%d", &year);
    if(year >= 20) {
        printf("成年¥n");
    } else if(year >= 0) {
        printf("未成年¥n");
    } else {
        print_error();
    }
}
```



変数のスコープ

変数が参照できる範囲

- 関数の引数
  - 関数の開始から終了まで
- 関数の中で定義された変数
  - 定義された場所から{}で構成されるブロックを抜けるまで



プログラム例(3)

```
#include <stdio.h>

void draw_bar(int width)
{
    int i;

    for(i = 0; i < width; i++) {
        printf("#");
    }
    printf("¥n");
}
```

引数1個

```
int main()
{
    int a;

    printf("taro: ");
    draw_bar(10);
    a = 8;
    printf("goro: ");
    draw_bar(a);
    printf("kuro: ");
    draw_bar(a / 2);
}
```

プログラム例(4)

```
#include <stdio.h>

void draw_mon_bar(int month, int width)
{
    int i;

    printf("%d月: ", month);
    for(i = 0; i < width; i++) {
        printf("#");
    }
    printf("¥n");
}
```

引数2個

```
int main()
{
    draw_mon_bar(1, 5);
    draw_mon_bar(2, 7);
    draw_mon_bar(3, 8);
}
```

プログラム例(5)

```
#include <stdio.h>

int calcFee(int width, int height, int depth)
{
    int fee;

    if(width + height + depth > 100) {
        fee = 1000;
    } else {
        fee = 600;
    }
    return fee;
}
```

引数3個

整数の戻り値あり

戻り値を返す

```
int main()
{
    int w, h, d, f;

    printf("幅: ");
    scanf("%d", &w);
    printf("高さ: ");
    scanf("%d", &h);
    printf("奥行き: ");
    scanf("%d", &d);
    f = calcFee(w, h, d);
    printf("%d円¥n", f);
}
```

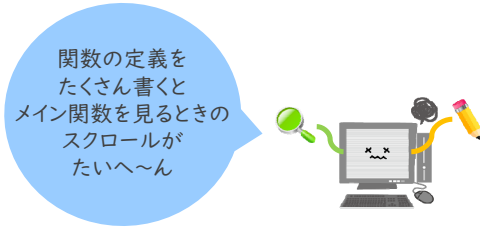
戻り値を受け取る

プログラムの構造の復習

メイン関数の実行に必要な情報の記述

```
int main()
{
    実行される命令
}
```

メイン関数



プロトタイプ宣言

戻り値の型 関数名(型名 引数名, 型名 引数名, …);

◆ メイン関数の前で関数の仕様を宣言すると，関数の定義をメイン関数の後に書ける

```
#include <stdio.h>

void print_hello(void);

int main()
{
    printf("Good morning¥n");
    print_hello();
    printf("Good evening¥n");
}

void print_hello(void)
{
    printf("Hello¥n");
}
```

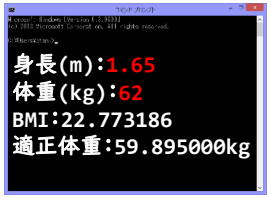
引数名は
省略可

演習

身長(m)と体重(kg)を入力すると，BMIと適正体重が表示されるプログラムを作成せよ．ただし，BMIは関数calcBMI，適正体重は関数calcAppWeightを使って算出すること．関数はプロトタイプ宣言をして，メイン関数の後に定義を書くこと．

関数calcBMI
　引数　：身長(m) <double型>
　　体重(kg) <double型>
　戻り値 ：BMI <double型>
関数calcAppWeight
　引数　：身長(m) <double型>
　戻り値 ：適正体重(kg) <double型>

<実行例>



赤字は実行時にキーボードから入力する部分

プログラム名はe10とすること．

演習　～手順～

1. 中身が空のメイン関数，必要なヘッダファイルを書く
2. 作る関数の仕様は？
   - → 関数名はcalcBMIとcalcAppWeight
   - → 引数と戻り値はすべて実数
   - → プロトタイプ宣言を書く
   - → 中身が空の関数を書く

```
#include <stdio.h>

double calcBMI(double height, double weight);
double calcAppWeight(double height);

int main()
{
}

double calcBMI(double height, double weight)
{
}

double calcAppWeight(double height)
{
}
```

演習 ～手順～

3. メイン関数で必要な変数は？
   - → 身長, 体重, BMI, 適正体重を代入する変数
   - → 型と変数名を決めて変数を宣言する
4. メイン関数の処理の順番は？
   - → 「身長(m):」を表示する（printf）
   - → 変数に値を読み込む（scanf）
   - → 「体重(kg):」を表示する（printf）
   - → 変数に値を読み込む（scanf）
   - → BMIを計算する（関数呼出し）
   - → 適正体重を計算する（関数呼出し）
   - → BMIと適正体重を表示する（printf）

演習 ～手順～

5. calcBMI関数で必要な変数は？
   - → BMIを代入する変数
   - → 型と変数名を決めて変数を宣言する
6. calcBMI関数の処理の順番は？
   - → BMIを求める
   - → 求めたBMIの値を返す（return）
7. calcAppWeight関数で必要な変数は？
   - → 適正体重を代入する変数
   - → 型と変数名を決めて変数を宣言する
8. calcAppWeight関数の処理の順番は？
   - → 適正体重を求める
   - → 求めた適正体重の値を返す（return）

BMI ＝ 体重kg ÷ (身長m)^2
適正体重 ＝ (身長m)^2×22