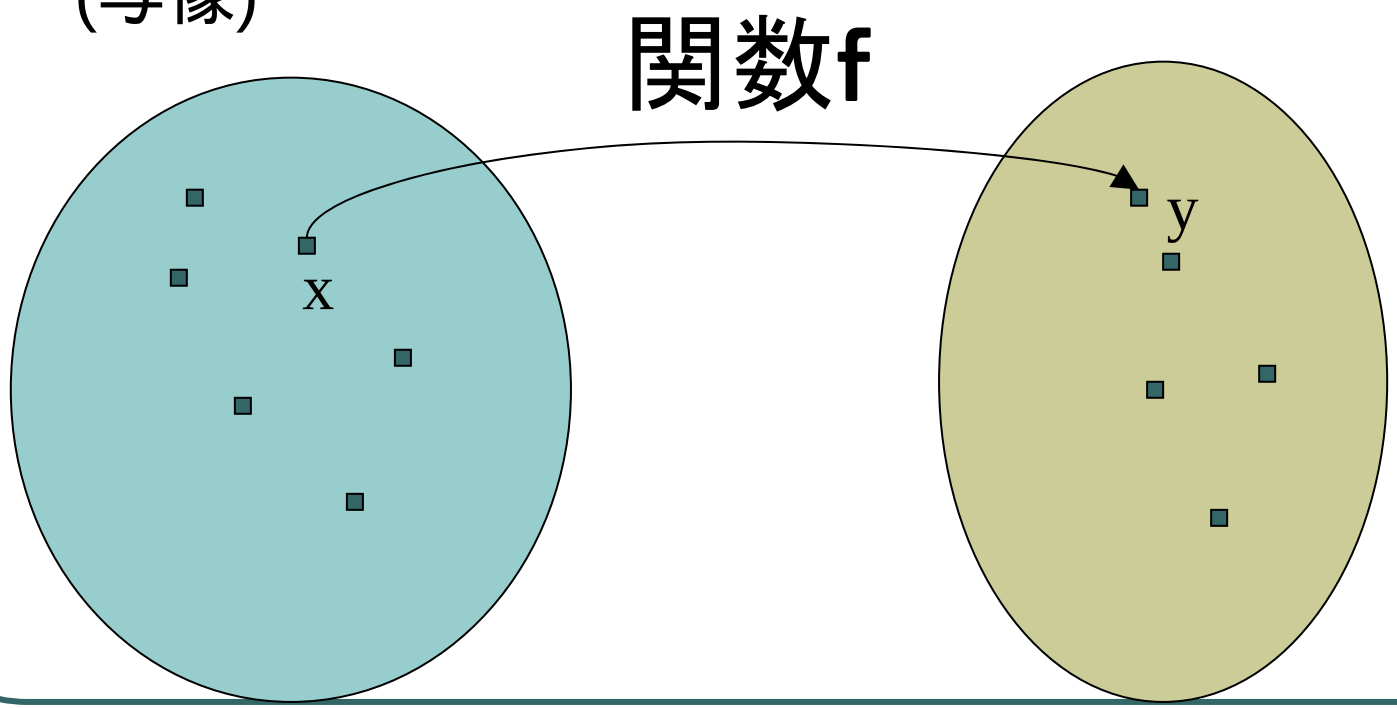
コンピュータ概論　第12回

関数

関数

- 関数(集合論的定義)
  - ある集合の要素を別の集合の要素と対応付ける関係(写像)

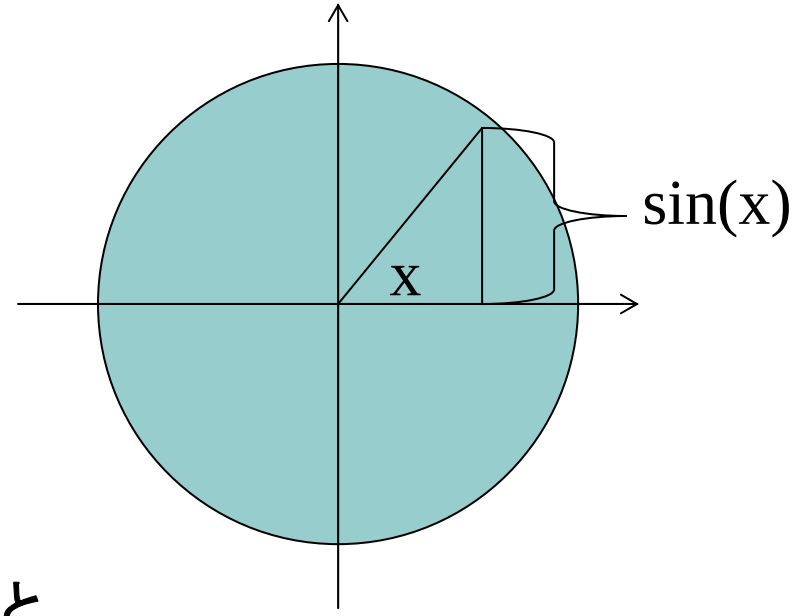


関数

- 一般的な関数
  - １変数関数　y=f(x)
  - ２変数関数　z=f(x,y)
  - 三角関数　　y=sin(x)

    f,sin:関数の名前

    (x),(x,y):関数の項
- 関数の定義
  - 具体的な関数の中身を示すこと
    - 1次関数　　f(x)=5x+3
    - 三角関数　　sin(x)={半径1の円の内角xにおける正弦値}



関数とは

- 項の値が具体的に決定されると、その値を算出できるように算出方法を定義したもの

↓

- 関数値を算出するための手続きを定義したもの

↓

- 関数値を算出するためのアルゴリズム

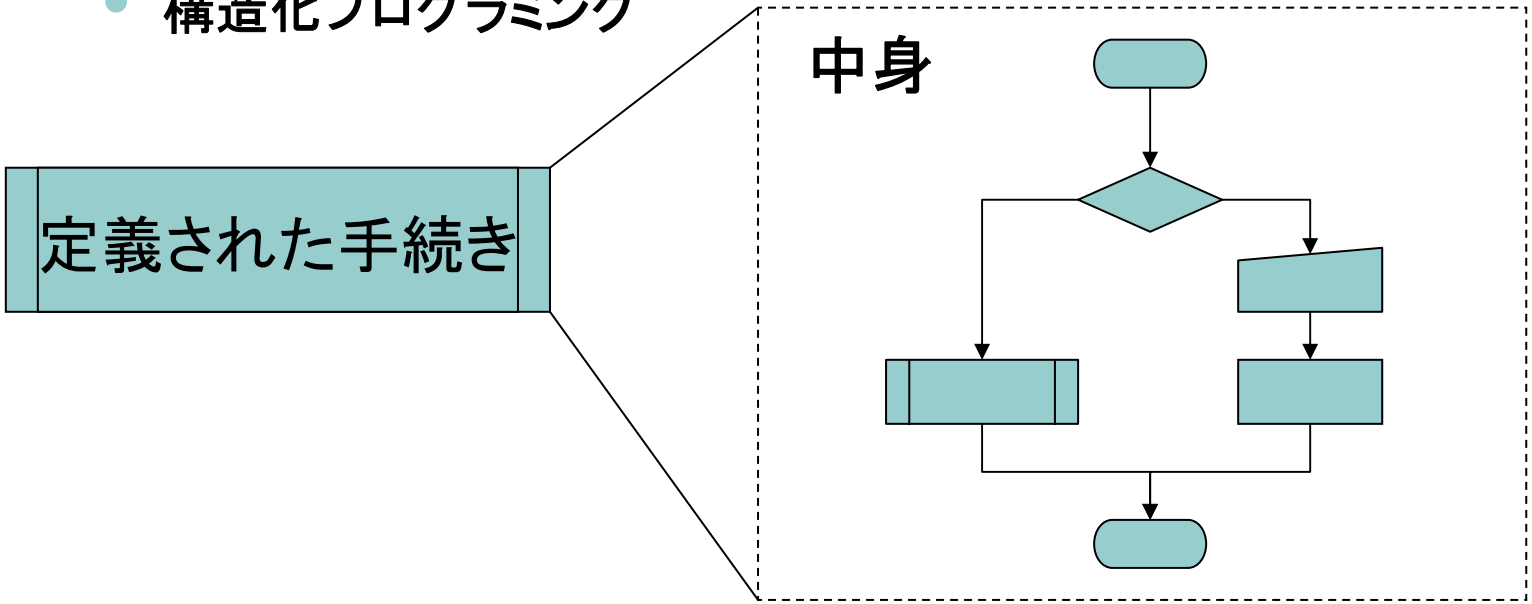
プログラミング言語における関数

- 数学的関数（function）
  - 何らかのアルゴリズムで値を計算するもの
- 手続き的関数（procedure）
  - 何らかのアルゴリズムを実行するもの

- 引数
  - 数学における関数の項と同じ

関数とサブルーチン

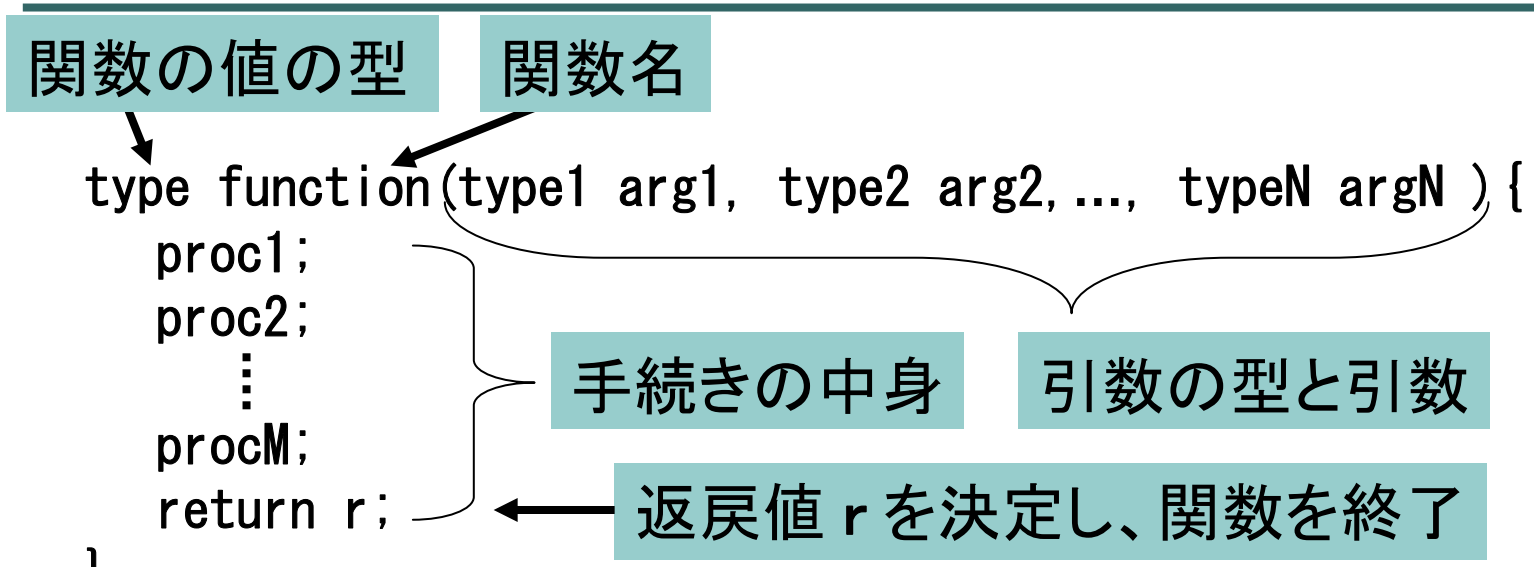
- 関数は独立したプログラムと同じ
  - 1つの関数には1つのフローチャート（アルゴリズム）
- 定義された手続き（サブルーチン）として利用
  - アルゴリズム（プログラム）をスマートにできる
  - 構造化プログラミング



Cにおける関数

- **プログラムの基本単位**
  - プログラムはいくつかの関数を組み合わせて作られる
  - プログラムを実行するとmain関数が実行される
- **関数の定義**
  - 何らかの手続きをまとめていれば、まとめ方は自由
- **関数の中と外は別世界**
  - 同じ名前の変数でも別のものとして扱われる
- **一度定義すれば、再利用できる**
  - よく使う処理を関数としておくと楽

Cにおける関数の定義



- 返戻値を返さない関数の型はvoid
  - returnで終了
- main関数の型はint（通常は省略）
  - 本来はreturnが必要（通常は省略）
  - システムによってはvoid

関数定義の例

Ex) f(x,y)=4x+5y

```
float func1(float x, float y){
  float r;
  r = 4*x+5*y;
  return r;
}
```

関数の利用

- 利用する場所の前で定義する

```
#include<stdio.h>
float func1(float x, float y){
  float r;
  r = 4*x+5*y;
  return r;
}
main(void){
  float a, b, c;
  scanf("%f", &a);
  scanf("%f", &b);
  c = func1(a, b);
  printf("Answer is %f¥n", c);
}
```

関数の利用

- 利用する場所の後で定義したい場合
  - 関数の型宣言を行う

```
#include<stdio.h>
float func1(float, float);
main(void){
  float a, b, c;
  scanf("%f", &a);
  scanf("%f", &b);
  c = func1(a, b);
  printf("Answer is %f¥n", c);
}
float func1(float x, float y){
  float r;
  r = 4*x+5*y;
  return r;
}
```

関数定義の例

Ex)文字（エラーメッセージ)を表示する

```
void dataErr(void){
  printf("入力データにミスがあります¥n");
  return;
}
```

- 引数が不要の場合は、引数をvoidとする

関数の利用

- 引数無しの例

```
#include<stdio.h>
void dataErr(void){
  printf("入力データにミスがあります¥n");
  return;
}
main(void){
  int a,b;
  scanf("%d", &a);
  if(a<=0) dataErr();
  scanf("%d", &b);
  if(b<=0) dataErr();
  printf("底辺%d、高さ%dの長方形の面積は%dです¥n",a,b,a*b);
}
```

変数の範囲(スコープ)

- 同じ変数も関数が違えば別

```
#include<stdio.h>
float func1(float s, float t){
  float u;
  u = 4*s+5*t;
  return u;
}
main(void){
  float x, y, s;
  scanf("%f", &x);
  scanf("%f", &y);
  s = func1(x, y);
  printf("Answer is %f¥n", s);
}
```

ライブラリ関数

- プログラミングする上で、有用な関数をまとめたもの
  - 三角関数
  - ソート関数
  - 統計処理関数
  - 入出力関数

    printf, scanfもライブラリ関数（標準関数とも言う）
  - ネットワーク通信関数

ライブラリ関数の利用

- #include<……>部で関数の型を宣言
  - ……はヘッダファイル名
    - ヘッダファイルはライブラリ関数などの型宣言用ファイル
    - scanf, printfはstdio.hファイルで型宣言
    - sin, cos, log, expなどは、math.hファイルで宣言
- ライブラリ関数を使うにはどのヘッダファイルで型宣言されているかを知っている必要がある。
  - 勉強が必要
  - いったん覚えれば、無駄なプログラムを書かなくても良くなる。(使えるものは利用する)

```c
/*ライブラリ関数を使わないプログラム*/
#include<stdio.h>
double sin(double);
double power(double, int);
int fact(int);

main(void)
{
  double t, s;
  printf("正弦値を求める。¥n入力単位はラジアン
  scanf("%lf",&t);
  s = sin(t);
  printf("正弦値はsin(%lf) = %lfです。¥n", t, s);
}
/*級数展開したsin関数*/
double sin(double th)
{
  double s = 0;
  int i;
  for (i = 1; i < 5; i++)
    s += power(-1.0, i-1) * power(th, 2*i-1)/(doub
  return s;
}
/*指数(xのm乗, m > 0)*/
double power(double x, int m)
{
  double p = 1;
  int i;
  if ( m < 0 ) return 0.0;
  for( i = 0; i < m; i++) p *= x;
  return p;
}
/*階乗(n!, n > 0)*/
int fact(int n)
{
  int f = 1;
  while(n > 1){
    f *= n;
    n--;
  }
  return f;
}
```

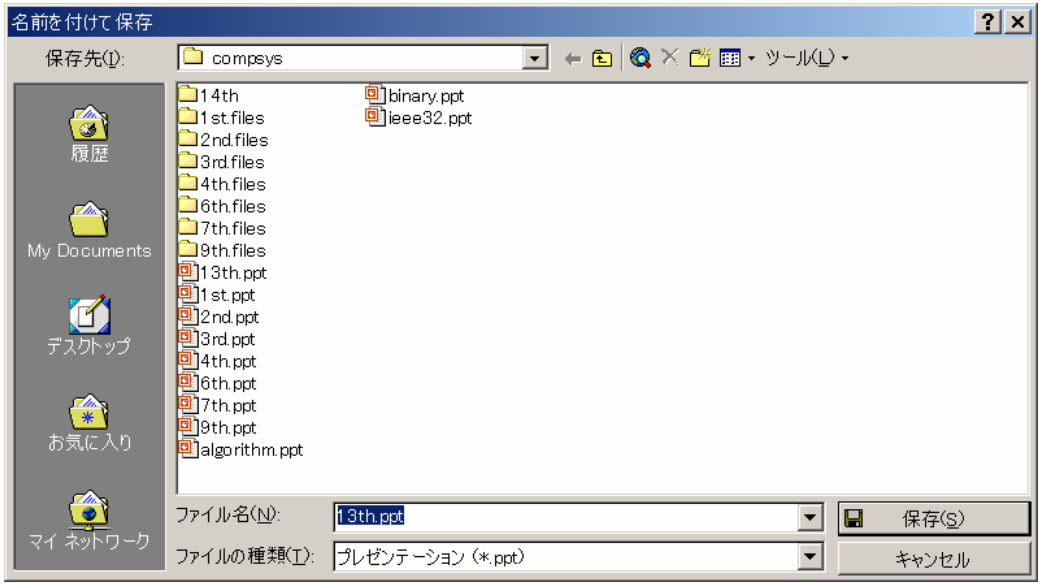
```c
/*ライブラリ関数を使ったプログラム*/
#include<stdio.h>
#include<math.h>

main(void)
{
  double t, s;
  printf("正弦値を求める。¥n入力単位はラジアン。¥n");
  scanf("%lf",&t);
  s = sin(t);
  printf("正弦値はsin(%lf) = %lfです。¥n", t, s);
}
```

これだけの差が生じる

API

- Application Program Interface
  - OSやその他のアプリケーションの機能を利用するための
    ライブラリ関数群
  - Windowsの「ファイルを保存」ダイアログのようなものを簡単
    に利用できる。
  - ウィンドウを使う
    ような高度なプロ
    グラムを簡単に
    作成することが
    できる。



おさらい

- 関数
  - 数学における関数との違い
  - サブルーチン
- Cにおける関数
  - 記述方法
  - 使用方法
  - ライブラリ関数

課題

- 引数に整数をとり、その整数が偶数か奇数かを判定した結果を表示する関数を作って、9回目の授業の課題のプログラムを書き換えよ。
  - 偶奇判定の関数の型はvoidとする。
  - 関数名はeoro。
  - 整数の入力はmain関数の中で行うこと。
  - 判定から表示までをeoro関数の中で行うこと。
    （整数の入力はmain関数の中で）
  - 9回目の課題と同様に、1以下の整数が入力されるまで繰り返し判定を行なえるようにする。