# Cプログラミング1(再)
# 第6回

講義では、Cプログラミングの基本を学び
演習では、やや実践的なプログラミングを通して学ぶ

# コンピュータは計算機

Cプログラミングと数学の関係

普通の数学とは異なる　　「情報数学」

Cは演算子を使って計算する

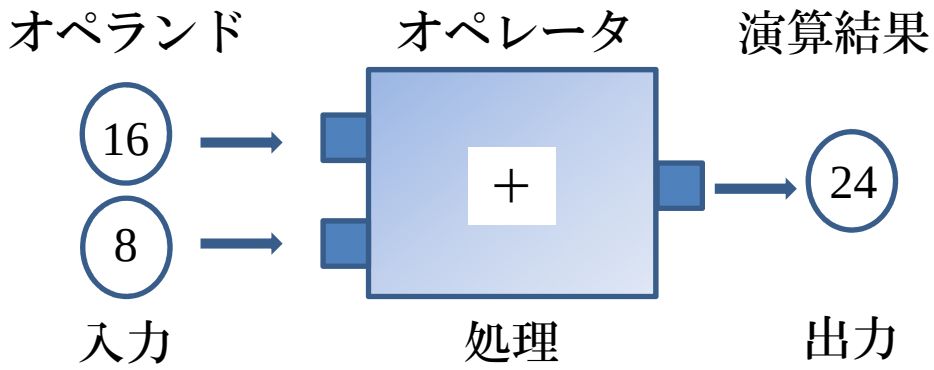
オペレータ(演算子) と  オペランド(非演算数)

例:  100 ÷ 20

式は「オペレータ」と「オペランド」から作られる

# 演算子を使った計算

「オペレータ」に「オペランド」が入力され、演算結果が得られる




Cの大原則
　演算結果は数になる

# 演算子には優先順位がある

演算子には優先順位と結合規則がある

それは数学における数式と同様

例: 7 + 3 × 2　　は　(7+3)×2　とは計算されない

×の方が＋よりも「優先順位が高い」

⇒ 先に計算される

# 定数と変数

定数：「値が固定された数」

例: 3.1415926536

100L　　6.02e+23　　0123　　0x80　　‘A’

意味のある定数は #define を用いて名前をつける

変数：「値が変わる可能性のある数を記憶する」道具

# イコール(=)は代入

数学では＝は等式

だから $T\cos\theta = S+Mg$ と書いても良い

Cでは=は代入

```
x = 10      // 10という値をxに代入する(記憶させる）
x = x+1   //  xの値に1を加えた値をxに代入
```

だから Tcosθ = S+Mg と書いてはいけない

コンピュータは式の変形を行わない
式の変形は人間の仕事
　　その式に値を代入して計算結果を求めるのが
　　プログラムの基本

最適化(optimize):
　処理速度を高めるために式の変形をコンパイラが行う
　ただし万能ではない---効率が悪い計算方法ではダメ

# 2進数の計算

Cでは２進数の基本演算ができる
（入門コースでは普通はやらないテーマだが）

| | |
|---|---|
| 論理積 | & |
| 論理和 | \| |
| 排他的論理和 | ^ |
| 否定 | ~ |
| シフト演算 (左シフト、右シフト) | << >> |

bitop.c参照

# 演算子

四則演算と剰余

＋　－　＊　／　％

演算結果は「型」に依存する

int 同士の演算結果は int になることに注意

# カンマ演算子

「左から順番に計算する」ことを表すのがカンマ演算子

例:

```
a = 3,   b = a * 2
a = (1,  2*3,  4+5)
```

# 増分演算子、減分演算子

増分演算子: ++　increment(インクリメント)
減分演算子: －－　decrement(デクリメント)
例:

$x = 10$ ,　y = $x$++
$x = 10$,　y = ++$x$
$x = 10$ ,　y = $x$−−
$x = 10$,　y = −−$x$

# 代入演算子

= は代入であり、 Cでは等号ではない

例：

```
a = 5;
a += 10;
a –= 3;
a /= 2;
a *= 5;
a %= 3;
```

比較：

```
data->people[number].score =
                    data->people[number].score +10;

data->people[number].score += 10;
```

わかりやすいのはどちら？

# 比較演算子

大小比較

＜　＞　＜＝　＞＝

等値比較

＝＝　！＝

C言語のポイント:

0 はfalse (偽、不成立）

それ以外はtrue(真、成立）を表す

# 論理演算子

論理積　　&&

論理和　　||

否定　　！

例：a とbとc が等しい　(a == b)　&& (b == c)

　　aかbがcと等しい　(a == c)　|| (b == c)

# 条件演算子

つう
通の人がよく書く書き方

```
x = (a %2 == 0) ? a : a+1;
```