プログラミング入門
第2回講義

Cプログラムの初歩
― はじめてのCプログラミング ―
（その1）

マークのあるサンプルプログラムは
`/home/course/prog0/public_html/2006/lec/source/`
下に置いてありますから、各自自分のディレクトリに
コピーして、コンパイル・実行してみてください

日本語のテキストファイルを作成する際に気をつけないといけない文字コードの話がLec11-23,24にありますので、読んでください

Cプログラムの初歩（1）

- Cプログラムのスタイル
- 1+2=3を計算させてみる
- 整数型のいろいろな演算

Helloと表示するプログラム(p.26)

kterm の画面上に **Hello** と表示させるプログラム(ファイル lec02-1.c に格納したとする)

```
#include <stdio.h>

main()
{
  printf("Hello\n");
}
```

/home/course/prog0/public_html/2006/lec/source/lec02-1.c

実行例

同じディレクトリ内にあるlec02-1.cをコンパイルする

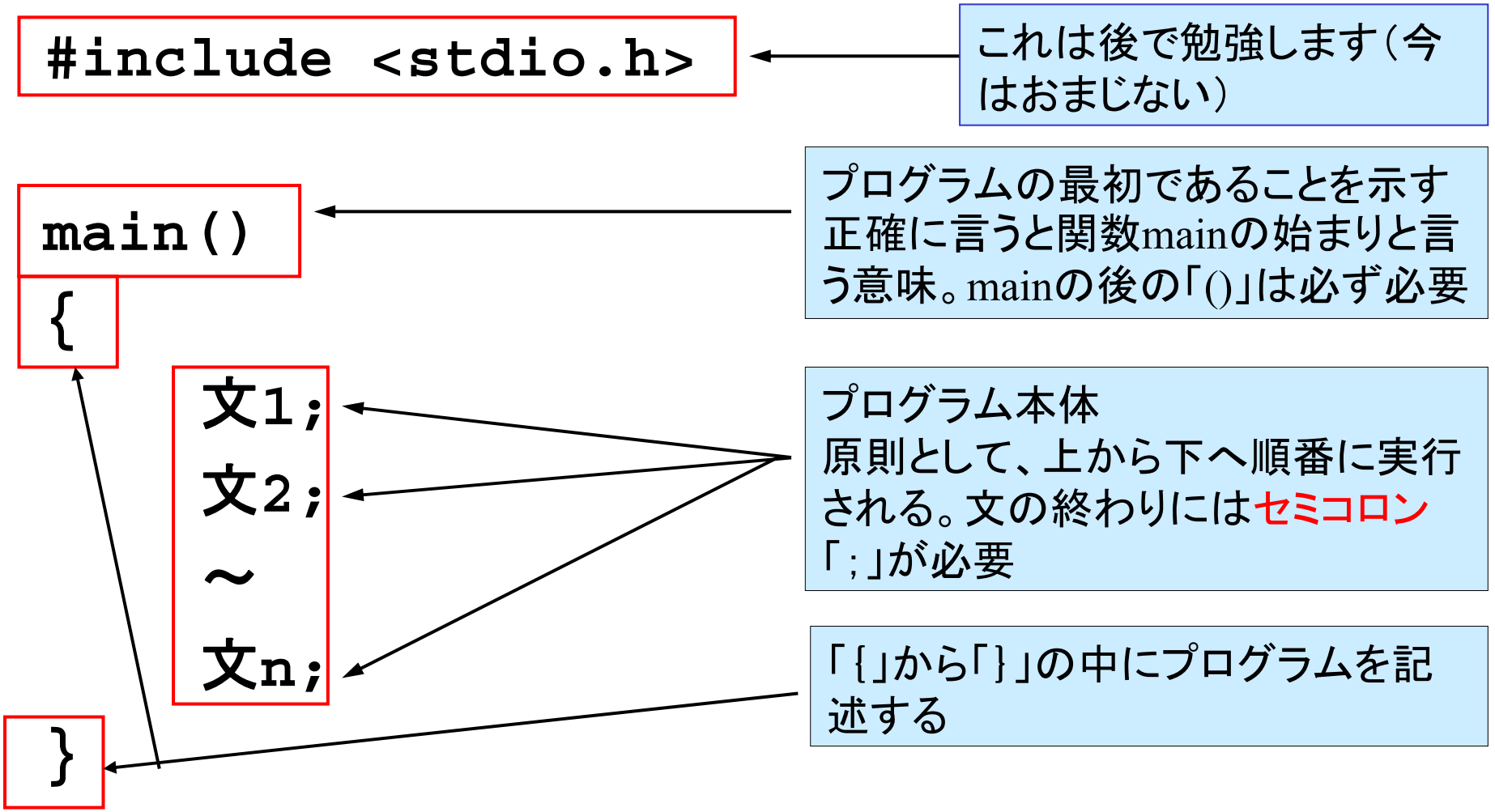
```
std1dc1{s1000000}1: gcc lec02-1.c
std1dc1{s1000000}2: ./a.out
Hello
std1dc1{s1000000}3:
```

実行

実行結果

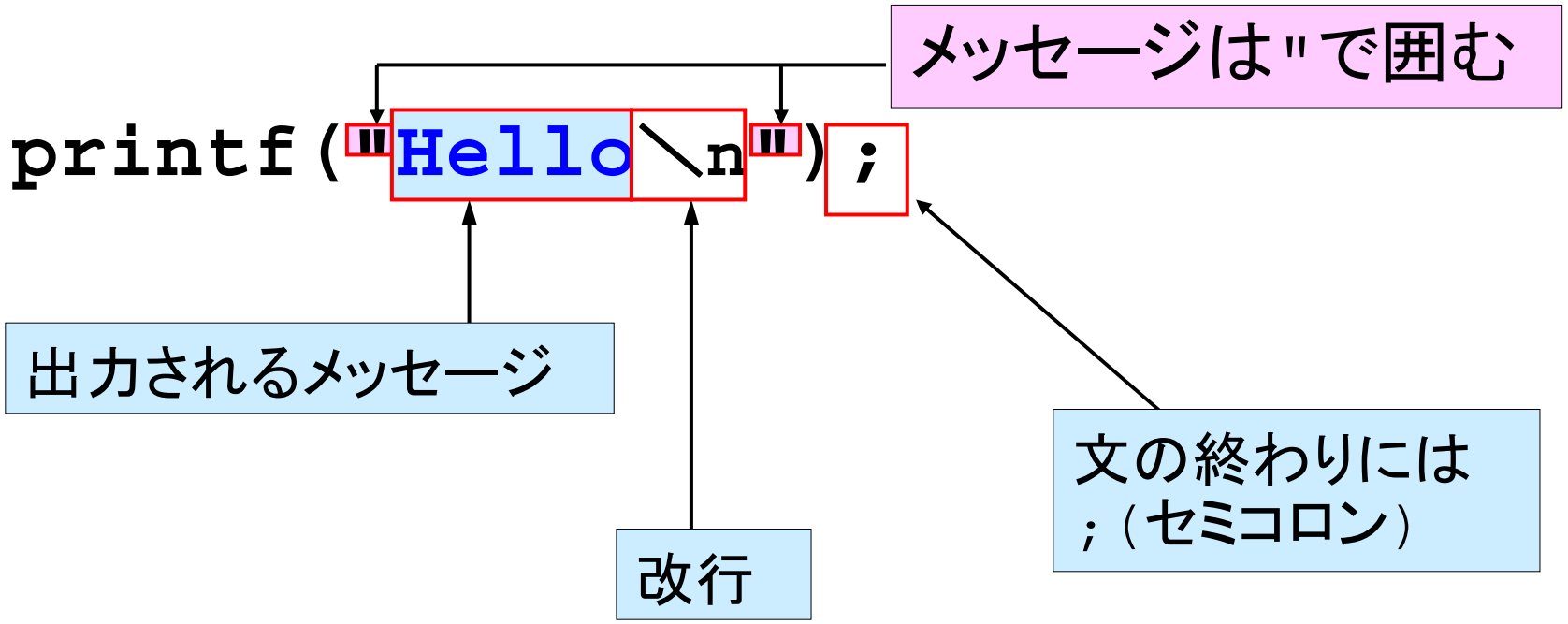
見易さのためにキー入力を青で、出力を赤で表示しています

Cプログラムの基本構造



printf文

標準出力（ディスプレイ）にメッセージを出力する



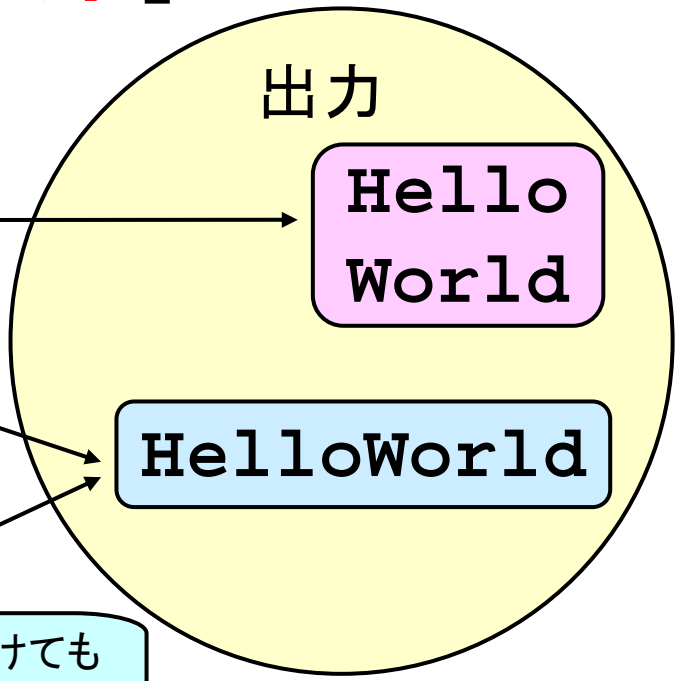
改行文字(p.34)

＼n は表示を改行させる「改行文字」

```
printf("Hello＼nWorld");
```

```
printf("HelloWorld");
```

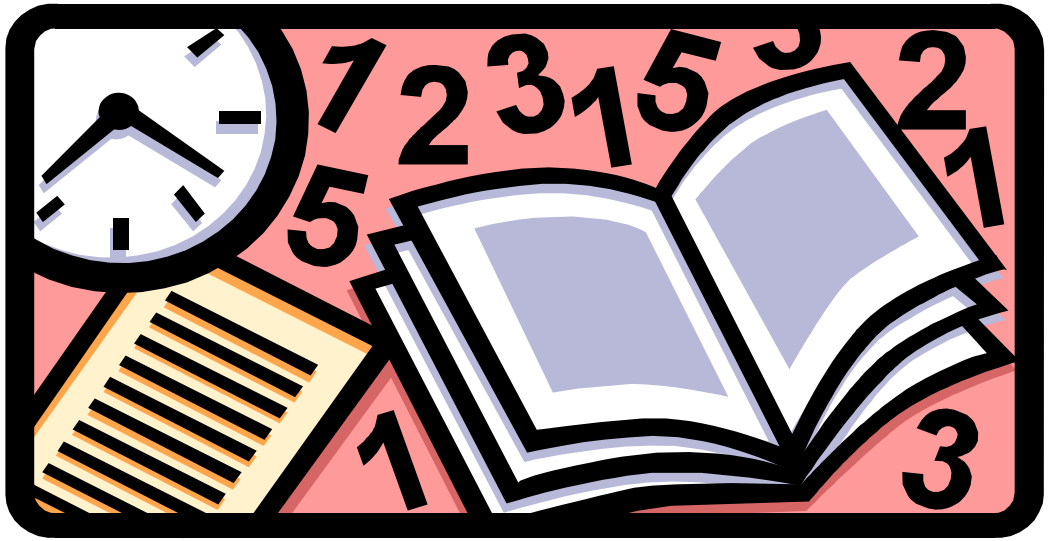
```
printf("Hello");
printf("World");
```



注:「＼」(逆スラッシュ)記号は日本語環境では「￥」記号となるので、本によっては改行を「￥n」と表示しているものもある

1 + 2 = 3 を計算するプログラム(p.40)

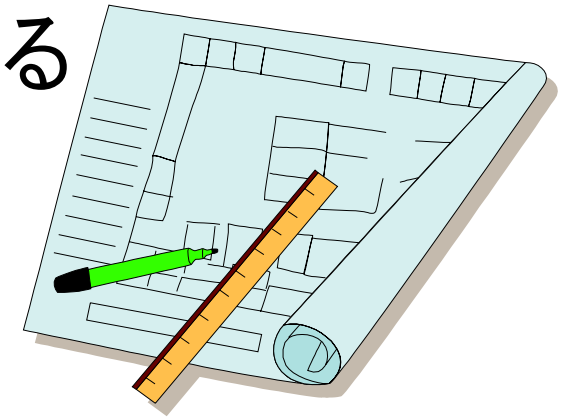
- 変数の扱い
- 整数演算
- 書式付出力



基本的な方針(p.40)
（言葉によるプログラム）

「言葉によるプログラム」を作ってみよう

1. 値を書きこむ変数「i」,「j」,「result」 を用意する
2. 2つの数をそれぞれ変数「i」,「j」 に代入する
3. 変数「i」と変数「j」の和を計算し、
   結果を変数 「result」 に代入する
4. 変数「result」 の値を表示する



1 + 2 = 3 を出力するプログラム

```
#include <stdio.h>

main()
{
  int i,j,result;

  i = 1;
  j = 2;
  result = i + j;
  printf("%d\n",result);
}
```

1:変数用意

2:変数に値代入

3:計算

4:結果の表示

/home/course/prog0/public_html/2006/lec/source/lec02-2.c

出力例

```
std1dc1{s1000000}1: gcc lec02-2.c
std1dc1{s1000000}2: ./a.out
3
std1dc1{s1000000}3:
```

変数(整数型変数)(p.40)

- コンピュータのメモリーに値を蓄えておく、名前つきの箱
- 今週は整数型のみを扱う
- 変数の内容は、何回でも書き換え可能
- 変数に書き込む事を「代入」と呼ぶ

変数名の規則(1)(p.62)

- 最初の文字は, 英字(A～Z, a～z)か、
  アンダースコア(下線： _) でなければいけない
- 2番目以降は, 英字, アンダースコア,
  数字(0～9) でなければいけない
- 大文字と小文字は区別される
- 変数名は31文字まで(32文字以降は無視される)

変数名の規則(2)

- 空白や, 英字, アンダースコア(下線), 数字以外の文字を含んではならない
  - `a123, Long_name, _123` ⇒ OK
  - `7abc, In-data, a.123` ⇒ NG
- 予約語(特別な意味を持つ単語)は使えない
  - `main, printf` などはだめ
- 意味の分かりやすい名前を付ける
  - `result,sum,total,average` など
  - これは「規則」ではないが、習慣にしよう！

変数宣言(p.40)

- 変数は宣言しなければ使えない

  「`int i,j,result;`」

  は整数型変数 `i,j,result` を用意する宣言
- 宣言はまとめてプログラムの最初に行う
  - プログラムの途中で宣言するとエラーになる！
- 宣言直後の変数の値は不定(何が入っているかわからない)
  - ⇒代入文(後述)を用いた初期化が必要

代入文(p.41)

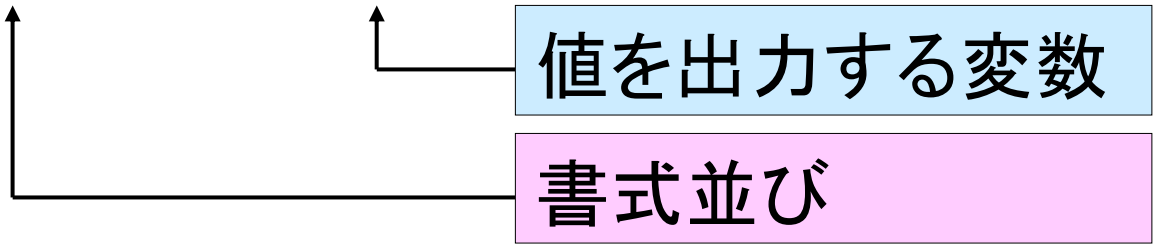
```
#include <stdio.h>

main()
{
   int i,j,result;

   i = 1;
   j = 2;
   result = i + j;
   printf("%d\n",result);
}
```

代入文：
右辺の値を左辺の変数に代入
数学の等号（ = ）とは意味が異なる

一旦値を代入してから使用する

書式付出力(p.43)

```
printf(  "%d＼n",result  );
```



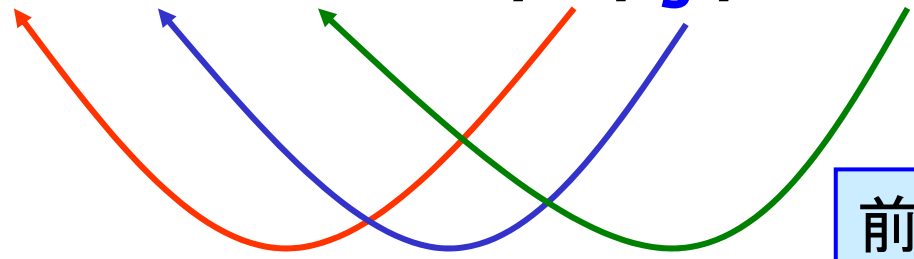
%d　:整数型変数の値を10進数で出力する

＼n　:改行

このハンドアウトの逆スラッシュ(＼)記号は良く見ると他の記号と異なり、全角になっている。これは日本語のシステムでは半角のバックスラッシュ記号が通貨記号(¥)になってしまうためである。見やすく表示するためにこのようにしている。実際にプログラムを作成する時には半角の逆スラッシュを使用する。

もう少し複雑な書式付出力(p.44)

```
printf("%d+%d=%d\n",i,j,result);
```



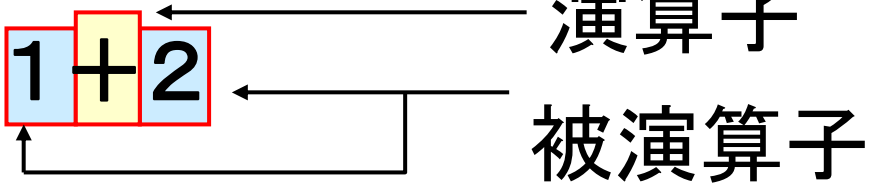
前から順に
値を表示

⇒ %dの部分が整数型変数の値に置き換わり、
「1+2=3」と出力される

（＋ や ＝ はそのまま出力される）

整数型のいろいろな演算

演算と演算子



演算：被演算子に対して演算子で定められた操作を行い、式の値とする

この場合1+2を計算し、答えの3を式の値とする

a = 1+2 とすると変数aには3が代入される

（整数型の）四則演算(p.43,89)

- ＋：加算
- －：減算
- ＊：乗算（「x」ではないので注意！）
- ／：除算（小数点以下切り捨て）
- ％：剰余算

  a%b ⇒ aをbで割った余り

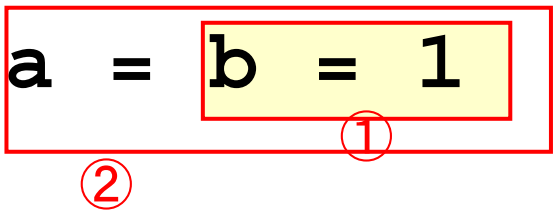
剰余算の注意(p.90)

- 整数型同士の演算でなければならない
  3.5 % 2 ⇒ エラーになる
- 剰余演算子を用いなくても余りは計算できる
  （整数型の場合）
  a % b ⇒ a − ((a / b) * b)

  上の式の意味を考えてみよう

代入演算子

- ＝も演算子（代入演算子）である
- 右辺の値を左辺の変数に代入し、式の値自体を代入する値にする(a = 5 の式の値は5)



①変数bに1を代入し、式「b=1」の値を1とする

②変数aに①で決まった式の値1を代入する

演算の順序と優先順序(p.94)

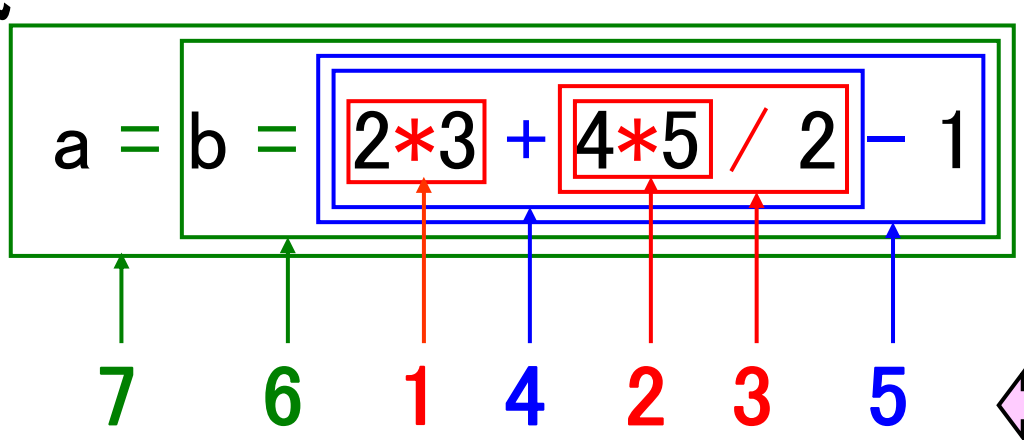
- 優先度の高い演算から実行される
  - １＊２＋３＊４ ⇒ (１＊２)＋(３＊４)
- 括弧つきの場合、演算子の優先度よりも優先される
  - ２＊(３＋４) ⇒ ３＋４が先に実行される
- 同じ優先度なら、計算は左から順に実行され、代入は右から順に実行される
  - １＋２＋３ ⇒ (１＋２)＋３
  - a = b = 1 ⇒ a = (b = 1)

演算子の優先度

高

1. 乗算 *、除算 /、剰余算 %
2. 加算 +、減算 −
3. 代入 =

低



⇦ 計算の順序