Cプログラミング1(再)
第3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

関数と部品

モジュール (module)

　　取替え可能な部品（プログラム全体に対して使う言葉)

ルーチン(routine)

　　メインルーチン vs サブルーチン

　　Cでは　メイン関数、補助関数　と呼ぶのが普通

プログラム作成：すべてを作る必要はない

ライブラリ(library)

既に用意されているプログラム: 特に「コンパイル済み」のもの

ヘッダファイルで使用可能になる

リンカ(linker)によって「実行ファイル」に組み込まれる

標準ライブラリ

stdio

stdlib

math (数学関数)

string (文字関数)

…

おまじないのように書いていた
#include <stdio.h>
というのは、標準ライブラリ stdio を使えるようにするためのもの

ただし、どのようなプログラム（関数）が使えるようになるかは
予め（調べて）知っておかないといけない

決まった処理はコンピュータに任せよう

プリプロセッサ (preprocessor) --- 前処理プログラム

#include

#define

#ifdef　#else　#end

まずは #include を学ぼう

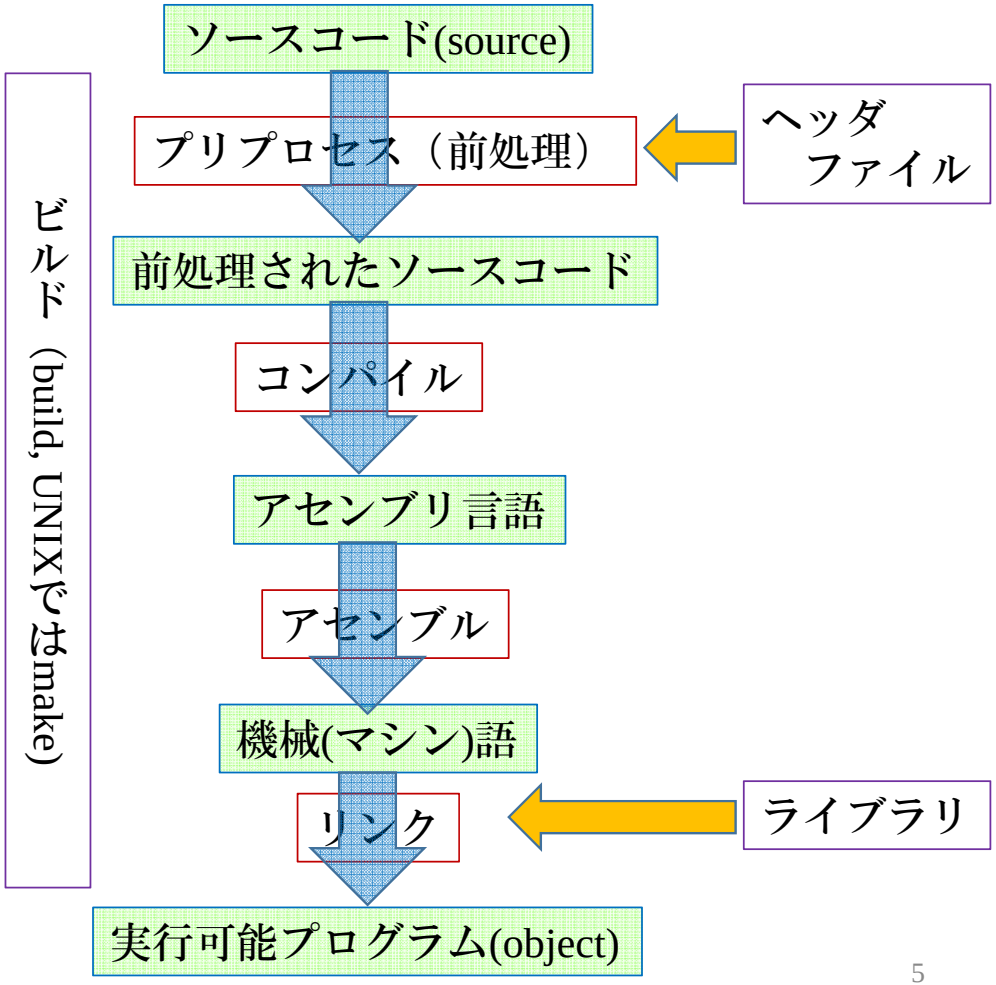
この役割は前のスライドからわかったはず

書いたプログラムが実行されるまで

(1) コンパイル、アセンブル
　　ソース(source)
　　オブジェクト(object)

(2) ライブラリのリンク

　クロスコンパイル



エラーになると実行できない

エラーメッセージをよく読む

エラーメッセージがたとえ

「英語であっても理解できるようにする」

実行中にエラーが起きることもある

暴走

Ctrl-C

バグ(bug)
デバッグ(debug)

処理系（コンピュータやOS）依存

ソース・プログラム（ソースコード、人間が読めるもの）
と
実行可能プログラム(バイナリ、EXEファイルなども）
は「全く別物」

実行ファイルは処理系依存
（CPUやOSなどにより実行できる、できないが決まる）

ソースプログラムは、簡単なものなら処理系に依存しない
処理系に依存しないように作るのが理想
Unix (Linux) の文化では configureと make を使って…

プログラムの作り方

例題プログラムを見て、意味もわからず適当に書き換えて、試行錯誤してプログラムを作ろうとする人は

いつまでたってもプログラムを作れるようにはならない

基本をおさえ、きちんと筋道をたてて実行しなければ意味が無い

プログラムを作るときの考え方を学ぶ

プログラムを作るときの考え方

(1) 手順を考える

1990年4月3日生まれの人は今何歳か？

プログラムを作る = 値が変わっても処理できるようにする

同じプログラムで「1979年10月10日生まれ」の人の年齢も分かる、のが望ましい

(2) 入力、処理、出力を考える

今日はx年y月z日とする．a年b月c日生まれの人の年齢を求める計算式はどのようなものか？

コンピュータ処理の基本:「入力、処理、出力」 --- 関数

注意: キーボード入力の『入力』、画面出力の『出力』と混同しないように
「処理（関数）」の『入力』とは処理に必要なデータ、『出力』とは処理の結果、のこと

具体的に

1990年4月3日生まれの人は今何歳か？

を計算するプログラムを考えていこう

そのために、Cの関数について。。。

関数とはなんだろう？(1)

Cは関数を多用する
だから、関数がわかっていないと、致命的につまずく

関数ってなんだろうか？

関数とは：形から入る

Cの関数は、基本的に次の形をしている

①関数が返す値の型 ②関数名（③引数リスト）
{
④いろいろな処理をする文;
…
}

注: ③で引数がない場合 void と書く
①で返す値がない場合 void と書く

例1:

```
①  ②    ③
int main(void) {
    printf("Hello, world!¥n");
  ④ return 0;
}
```

註：main関数だけは(エラーがなく実行できたら)
『返す値は0』と決まっている。

関数とは：形から入る

Cの関数は、基本的に次の形をしている

①関数が返す値の型 ②関数名 （③引数リスト）
{
④いろいろな処理をする文;
…
}

例2:

```
①int ②max(③int x, int y) {
④{    if (x > y)
          return x;
      else
          return y;
}
```

関数とは:その意味

数学の関数を考える：

sin(θ) : 引数θに対するsinの値を返す

Cの関数も基本的に同じ

③引数をもらい、

④計算し、

値を返す①

そしてそれに名前をつけたもの②

引数xとyのうち、大きい方を返す関数 max

例2:

```
①int ②max(③int x, int y) {
    if (x > y)
④     return x;
    else
      return y;
}
```

Cの関数の引数について

関数の呼び出し(使い方):

Cのどこからでも利用可能

呼び出し:「利用」すること

典型的な利用法：

ans = max(a, b);

max関数の呼び出しの例

max関数が計算できるよう、引数を与える

ここではaもbも整数型の変数とする

大事なこと：この呼び出しでは、aもbも値が入っていること

呼び出し側の引数を「実引数」という

それに対し、関数定義で用いられる引数を『仮引数』という

例2:

```
①int②max(③int x, int y) {
    if (x > y)
 ④    return x;
    else
      return y;
}
```

引数に2通り： 実引数と仮引数

この違いを学ぼう

ポイント：関数定義の「引数」は、関数の呼出しに使われる「引数」とは別のモノ

関数の呼び出しで何が起こっているか？

a =1; b= 2;

ans = max(a, b);

とする。

```
例2:
①int②max(③int x, int y) {
    if (x > y)
 ④    return x;
    else
      return y;
}
```

ステップ１．実引数の値が仮引数に渡される
　つまり、実質的に
　　x = a; y = b;
　が行われる（値渡し）
ステップ２．④の部分が実行される
　　この場合、x > yが成り立たないので
　　return y
　　が実行される　⇒　これが戻り値
ステップ3. 呼び出したmax(a,b)は「戻り値」で置き換えられる
　　その結果、ansにmaxの計算結果が代入される

クイズ：どれとどれがCとして『同じ』関数？

```
// Program 1:

int fact(int n) {
  int ans=1;
  for( ; n > 1; n--) {
            ans *= n;
  }
  return ans;
}
```

```
// Program 2:

int fact(int x) {
  int i=1;
  for( ; x > 1; x--) {
            i *= x;
  }
  return i;
}
```

```
// Program 3:

int fact(int n) {
  int i, ans=1;
  for( i=1 ; i <= n; i++) {
            ans *= i;
  }
  return ans;
}
```

```
// Program 4

int fact(int n) {
  if (n>=1)
      return n*fact(n-1);
  else
     return 1;
}
```

```
// Program 5:

int fact(int n) {
  int ans;
  for(ans=1; n > 1;   ans*=n--) ;
  return ans;
}
```

```
// Program 6:

int fact(int n) {
  int ans=1;
  for( ; n > 1; n--) {
            ans *= n;
  }
}
```

```
// Program 7

int fact(int n) {
  if (n>=1) {
            return n*fact(n-1);
  } else {
            return 1;
  }
}
```

1990年4月3日生まれの人は今何歳か？

今日は、2017年4月20日

次のように計算が進むだろう:

2017 - 1990 = 2 7　　この結果が年令？

ちょっとまった。5月生まれだったりするとまだ27歳になっていない。
だから 誕生日の月と「今日の月」とを比べる…

この例の場合、同じ「４」月だから、「日」もみないと決められない
なぜなら　　４月２９日生まれならまだ26歳！

だから、（誕生日の月と「今日の月」が同じなら）
　　誕生日の日と今日の「日」を比べる…

1990年4月3日生まれの人は今何歳か？

今の「処理」の流れをまとめよう

1. まず年数の差を求めて記憶する

   「記憶する」には変数が必要　age という名前にしておこう

   だけど、まだここでは決定できない…

2. 次に「月」同士を比較

   誕生月が今日の月より大きい場合、小さい場合、等しい場合、

   それぞれで年令はどうなるか？

3. 最後に「日」を比べる

   これが必要なのは、「月」が等しい場合だけ！

   誕生日の「日」が今日の「日」より大きい場合、小さい場合、等しい場合、

   それぞれで年令はどうなるか？

以上の結果から「何歳」かが決まる。このやり方が正しいかどうかは、

1997年1月1日生まれ、2000年5月3日生まれ、2003年4月29日生まれ、2011年4月20日生まれ、

それぞれの人の年令を考えてみて確かめてみよう

1990年4月3日生まれの人の年令を計算する関数 nenrei

関数nenrei が「処理」を行うには

(1) 今日の日付(2017年4月20日)

(2) ある人の誕生日の日付

が必要。これが関数nenreiの「入力」

日付の指定の方法にはいろいろあるが、ここでは(簡単のため)、「年、月、日」をそれぞれ別々に与えることにしよう。

すると、nenrei 関数は 入力として 3個の引数　を取る

誕生日の日付をa年b月c日とすると、a, b, c

1990年4月3日生まれの人の年令を計算する関数 nenrei

関数nenrei は計算をして、年令を返す

これは整数である。これが関数nenreiの「出力」

すると、入力をa, b, c の3個の整数とすれば、

関数 nenrei の定義は

```
int  nenrei (int a, int b, int c) {
        ここに処理が入る
 }
```

というような形になる

1990年4月3日生まれの人の年令を計算する関数 nenrei の処理

スライド20から:
1. まず年数の差を求めて記憶する
「記憶する」には変数が必要　age という名前にする
　だけど、まだここでは決定できない…
2. 次に「月」同士を比較
誕生月が今日の月より大きい場合、小さい場合、等しい場合、それぞれで年令はどうなるか？
3. 最後に「日」を比べる

スライド22から

現在の日付: 2017年4月20日
誕生日：a年b月c日

```
int  nenrei (int a, int b, int c) {

    ここに処理が入る

}
```

```
int  age;

age = 2017 – a;

if  (b > 4) {
    age の値をどうするか？
}  else {
    if (b < 4) {
        age の値をどうするか？
    } else { // b == 4
        3の内容を書く
    }
}
return age;
```

完成形

入力: 現在の日付: 2017年4月20日、 誕生日：a年b月c日

```
int nenrei(int a, int b, int c) {
    int age = 2017 - a;
    if (b > 4) {
        age -= 1;
    } else {
        if (b < 4) {
            // OK
        } else { // b == y
            if (c > 20)  age -= 1;
            else ;
        }
    }
    return age;
}
```

使用例:

```
#include <stdio.h>

// 関数nenreiの定義をここに埋め込む：

int main(void) {
    int a,b,c;
    a = 1997; b=1; c=1;
    printf("%d年%d月%d日生まれの人の年令=%d¥n",
            a,b,c,nenrei(a,b,c);
    a = 2000; b = 5; c= 3;
    printf("%d年%d月%d日生まれの人の年令=%d¥n",
            a,b,c,nenrei(a,b,c);
    // 以下略
    return 0;
}
```

1997年1月1日生まれ、2000年5月3日生まれ、2003年4月29日生まれ、2011年4月20日生まれ、で試す

プログラムの手直し

気がついたように、前のプログラムは冗長

```
int nenrei(int a, int b, int c) {
    int age = 2017 - a;
    if (b > 4) {
        age -= 1;
    } else {
        if (b < 4) {
            // OK
        } else { // b == y
            if (c > 20)  age -= 1;
            else ;
        }
    }
    return age;
}
```

```
int nenrei(int a, int b, int c) {
    int age = 2017 - a;
    if (b > 4) {
        age -= 1;
    } else {
        if ((b == 4) && (c > 20)) {
            age -= 1;
        }
    }
    return age;
}
```

演習問題1.

今日の日付を西暦x年y月z日とする．　西暦a年b月c日生まれの人の年齢を求めるプログラム（関数）を考えてみよう

　プログラムの形にできるだろうか？

　関数の「入力」は何か、「出力」は何か、またその「処理」はどうすればよいか、を答えよ

演習問題2

今日の日付を 西暦 x年y月z日とする. 元号(平成または昭和） a年b月c日生まれの人の年齢を求めるプログラムを考えてみよう

元号はＨもしくはＳで入力するとする：

例 H280401 ⇒ 平成28年4月1日

S601225 ⇒ 昭和60年12月25日

注意１：プログラムを書かなくても良い

どのようにすれば答えが出るか、そのアルゴリズムを答えよ

注意２: 1989年は1月7日まで昭和64年、1月8日から平成元年

ファイル入出力

ファイルからデータを「読む」、ファイルに結果などを「書く」ことをファイルの入出力と言います

ファイルを活用する

大きなデータをキーボード入力していられない

手間を減らす、入力間違いを減らす

データがファイルの形で渡されることが多い

プログラムが処理した大量のデータを画面に書出す(表示)してくれても読めない ⇒ ファイルに書出しておく

このようにプログラムが出力した「大量の」データをさらに他のプログラムで処理することが多い

---その時のデータのやり取りは「ファイル」

だからファイルの入出力が扱えるようになるのはとても大事

ファイルの入出力の基本

- 実は、キーボード入力、画面出力と基本的には同じ
- 本質的な違いは、「ファイルを開く」という作業が必要

  ファイルには「読み込み」（入力）用に開くのと、
  「書出し」（出力）用に開くのと
  大きく2通りがある　…. 本当のことを言うと、もうちょっとある

詳しくはそのうち述べる
ファイルからの読み込みは、キーボードからの入力と「ほぼ」同じ方法で、
ファイルへの書き出しは、画面への出力と「ほぼ」同じ方法で行われる