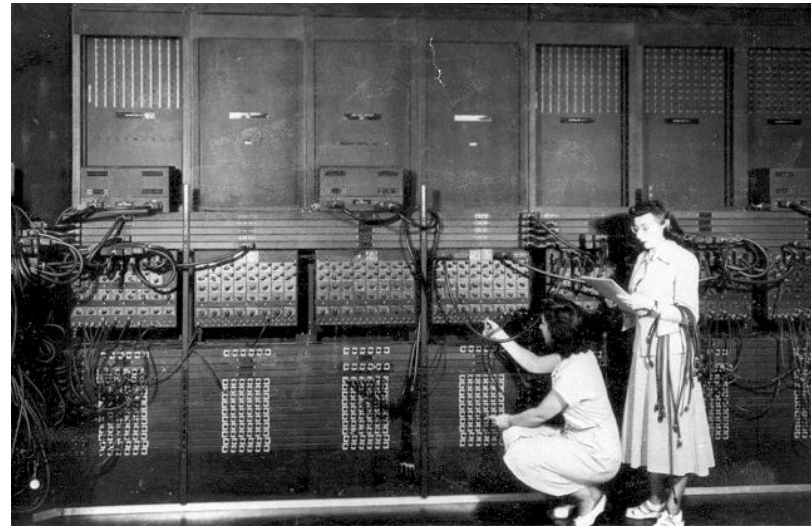




# 苦しんで 覚える C言語

4月18日：チーム1

# 0章：コンピュータとは何か？



ENIAC: <http://www.library.upenn.edu/exhibits/rbm/mauchly/jwm8b.html>

京: [http://www.kobe-np.co.jp/news/keizai/201402/p1\\_0006739022.shtml](http://www.kobe-np.co.jp/news/keizai/201402/p1_0006739022.shtml)

現代のコンピュータ達: <http://computertechsreno.com/> より参照。

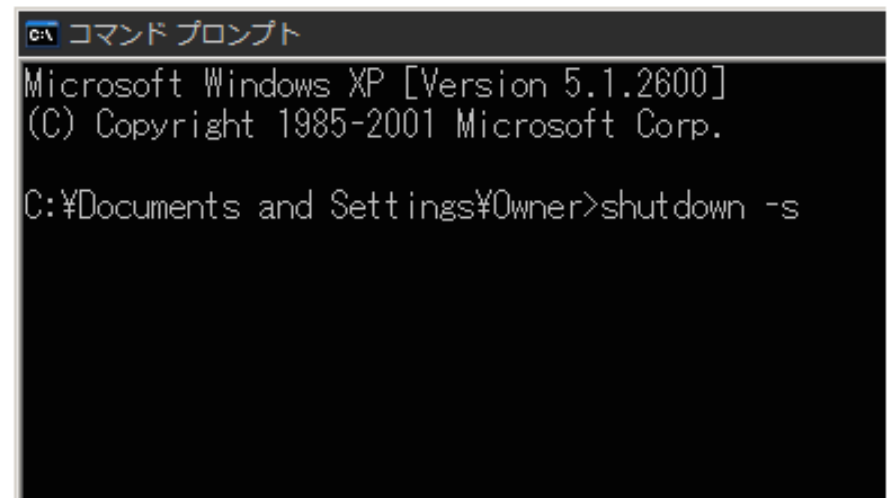
## 0.1.1 : 現代人とコンピュータ

- 現代では多くの人々がWindowsに代表されるPCを利用している。
- 技術の進歩により専門家でなくても、操作できるように工夫されているため普及した。
  - ↳ 現代人にとってとても身近な存在。

# 身近さゆえのわかりにくさ

- 現在のコンピュータは本質的な部分がうまく隠されている。

↳ 画面上のアイコンやボタンにプログラムを連動させている。



# コンピュータを理解する

- 0章では、
    - ・コンピュータという機械の正体
    - ・プログラムの正体
    - ・コンピュータの機能の正体
- などを通じて※本質的な側面の解説をしていく。

## 0.1.2 : コンピュータとは

- コンピュータ(電子計算機)とは電気を使って計算する機械のこと。
- コンピュータは多数の式を組み合わせ、多数の解を導き、再使用するまで記憶しておく。
- 計算結果から計算式の順番を自由に換え、新たな計算結果を出すことができる。
  - ↳ 複雑な処理を可能としている。

## 0.1.3 : CPUとは？

- 人間でいう脳の部分であり、コンピュータの核である部分。
- 現在、市販のCPUは2～3GHz程のスペックであり、1秒間に20～30億回程の計算をこなせるということになる。
- CPUは計算だけでなく、計算順序も変更できる。

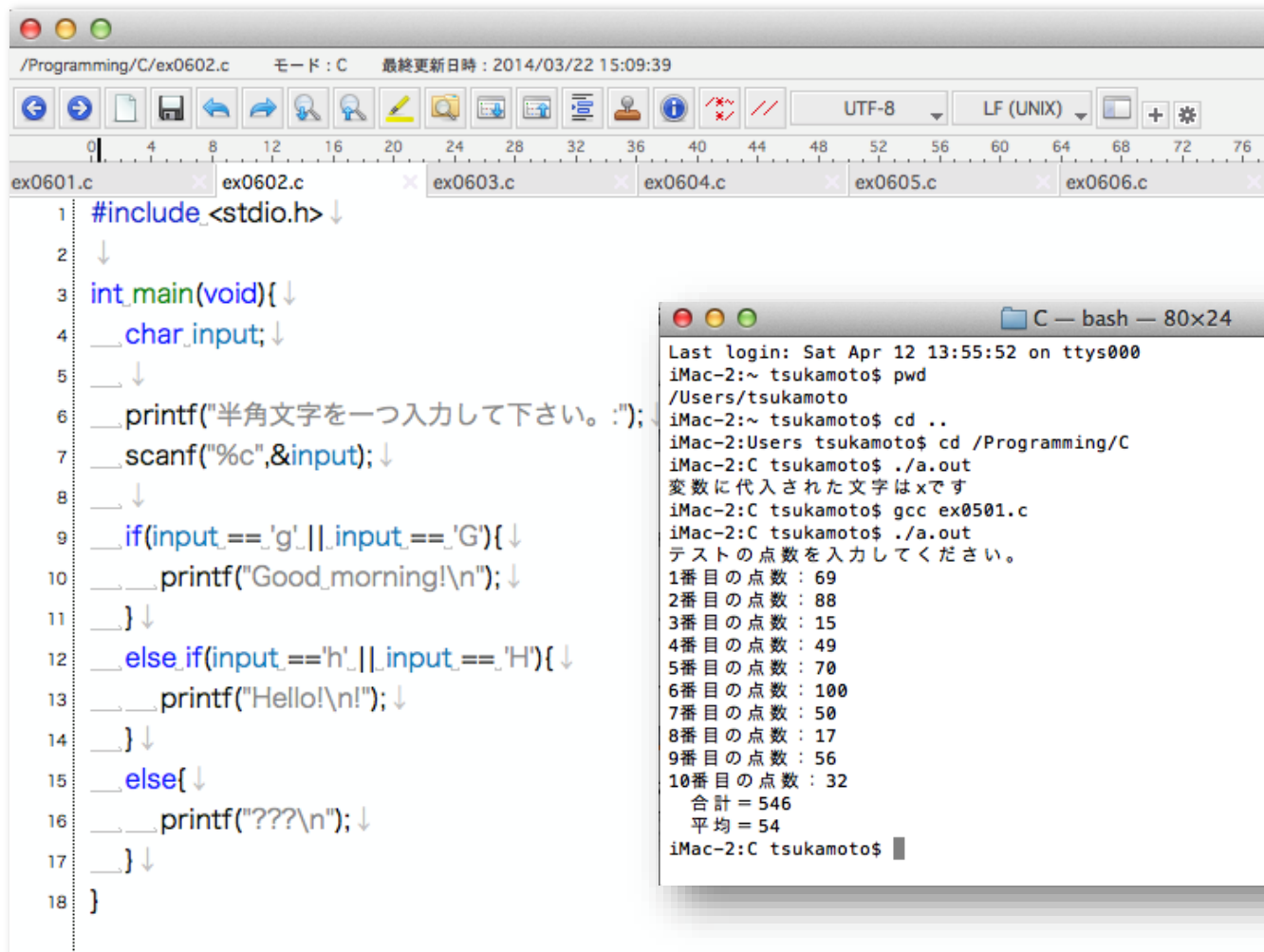
## 0.1.4 : メモリとは？

- CPUの計算により得た計算結果を記憶しておくところ。
- 現在、2～4GB程度のメモリが一般的であり、3桁の数字を20～40億個程記憶することができる。

↳ 記憶量が多いと何か良いことがあるのか？



## 0.2 : プログラムとは何か？



The image shows a code editor window with a file named `ex0602.c` open. The code is a C program that prompts the user to enter a character and then prints a message based on the input. The code is as follows:

```
1 #include <stdio.h> ↓
2 ↓
3 int main(void){ ↓
4     __char input; ↓
5     ↓
6     __printf("半角文字を一つ入力して下さい。:"); ↓
7     __scanf("%c",&input); ↓
8     ↓
9     __if(input == 'g' || input == 'G'){ ↓
10    __printf("Good morning!\n"); ↓
11    __} ↓
12    __else if(input == 'h' || input == 'H'){ ↓
13    __printf("Hello!\n!"); ↓
14    __} ↓
15    __else{ ↓
16    __printf("??? \n"); ↓
17    __} ↓
18 }
```

Below the code editor, there is a terminal window titled `C — bash — 80x24`. It shows the execution of the program. The user enters 'x' and the program outputs the scores for each character and the total and average scores.

```
Last login: Sat Apr 12 13:55:52 on ttys000
iMac-2:~ tsukamoto$ pwd
/Users/tsukamoto
iMac-2:~ tsukamoto$ cd ..
iMac-2:Users tsukamoto$ cd /Programming/C
iMac-2:C tsukamoto$ ./a.out
変数に代入された文字はxです
iMac-2:C tsukamoto$ gcc ex0501.c
iMac-2:C tsukamoto$ ./a.out
テストの点数を入力してください。
1番目の点数：69
2番目の点数：88
3番目の点数：15
4番目の点数：49
5番目の点数：70
6番目の点数：100
7番目の点数：50
8番目の点数：17
9番目の点数：56
10番目の点数：32
合計 = 546
平均 = 54
iMac-2:C tsukamoto$
```

## 0.2.1 : プログラムとは

- プログラムとは、コンピュータを動かす最も基本的な方法である。
- プログラムの特長=コンピュータの特徴
- プログラミングに習熟すれば便利なことができるプログラムを創ることができるが、あくまでも本質は計算である。

# プログラミング言語

- プログラミングを行うためには、  
プログラミング言語を習得する必要がある。
- 我々人間が使う日本語などは自然言語と呼ばれ、プログラミング言語は人工言語と呼ばれる。
  - ↳ 計算を表現することに特化している。

# 人気のプログラミング言語

1位 : JavaScript

2位 : Java

3位 : PHP

4位 : C#

5位 : Python

C言語は8位でした。(2014年1月)



<http://redmonk.com/sograpy/2014/01/22/language-rankings-1-14/>より。

# 最初のハードル

- プログラミング言語はコンピュータのための言語。
- 我々が簡単にプログラムが理解出来ないのは、プログラミング言語には自然言語と違う点が多いためである。
- つまり、プログラミング言語と自然言語の違いを理解することが、プログラミングを理解する上での最初のハードルとなる。

## 0.2.2 : 単純な文法

- プログラム言語と自然言語の一つ目の違いとして、プログラミング言語は自然言語よりも文法が単純なことである。
- プログラミング言語に使われる品詞は、実際のところ動詞と目的語がほとんどである。

# プログラムの例

プログラム

REFLECT HELLO

このプログラムは画面にHELLOという言葉映せ、という意味です。  
これを日本語の単語に置き換えるこういったものになる。

プログラム

映せ こんにちは

たった二つの単語だけだが、これがプログラミング言語の文法なのだ。

# プログラム言語の文法の単純さ

自然言語

画面に「こんにちは」という言葉を映せ。

プログラミング言語

REFLECT HELLO

日本語の単語

映せ こんにちは

⇒ プログラミング言語は単純すぎる？

⇒ 「どこ」に映すか指定する場合は？



# 「どこ」に映すか

## ① 画面に映したい場合

プログラム

映せ 画面 こんにちは

また、下のような書き方をすればプリンタに印刷してくれる。

プログラム

映せ プリンタ こんにちは

## 0.2.3 : 明確な意味

- プログラミング言語と自然言語の二つ目の違いは、プログラミング言語は自然言語に比べ、意味が明確なことである。
- 曖昧さが発生する余地は全くなく、厳格で明確な意味が決められている。

# 余談：BASICとPASCAL

- 1970年以降、初心者向けにBASICという
- プログラミング言語が広く使われた。
- FORTRANの文法を基にしている、現在の
- Microsoft Visual Basicの文法にも影を残している。
- BASICのおかげでプログラミングが身近になった。

# 余談：BASICとPASCAL

- BASIC登場以前のプログラミング言語は初心者にとって敷居が高かった。
- しかし、その取っ付き易さが仇となり難しい(複雑な)仕事には向かなかった。

# 余談：BASICとPASCAL

- 厳しさを教え、バグの起きにくいプログラムを作るために教育目的のPASCALが開発された。
- PASCALは非常に厳格で少しの曖昧さも許さない。
- 数字の「5」を書いた時、それは文字なのか数値なのかBASICなら勝手に判断してくれるが...

# 余談：BASICとPASCAL

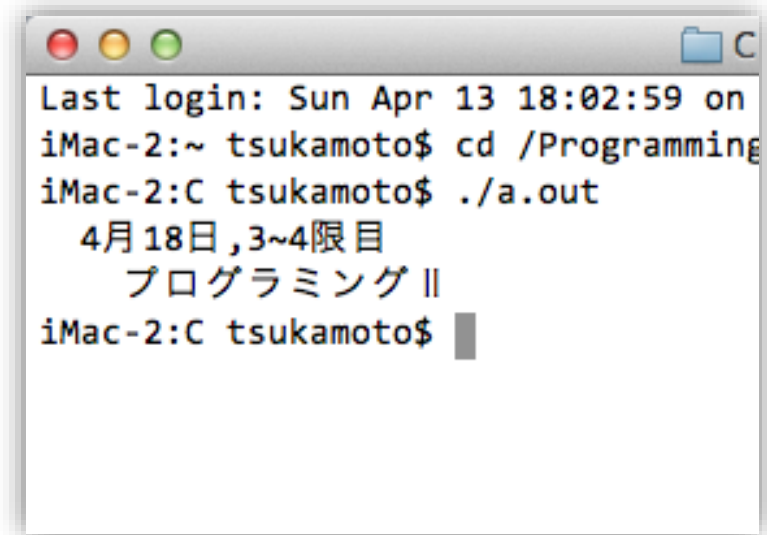
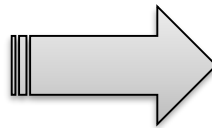
- なぜこうも厳格さが求められるのか？
- 複雑なプログラムになってくるとプログラムの信頼性に不安が出てくるからである。
- 厳格さを克服してプログラムを書けば、完成したプログラムは誤動作のない信頼性の高いプログラムになる。

# プログラミング言語は計算言語？

⇒「画面に文字を写す」という命令は、  
計算と関係があるのだろうか？



```
untitled
1  #include <stdio.h>
2
3  int main(void){
4      printf("  4月18日,3~4限目\n");
5      printf("    プログラミングII\n");
6
7      return 0;
8  }
```



```
iMac-2:~ tsukamoto$ cd /Programming
iMac-2:C tsukamoto$ ./a.out
  4月18日,3~4限目
    プログラミング II
iMac-2:C tsukamoto$
```

# プログラムの本質

- プログラミング言語は非常に緻密で厳格な計算しかできない。(＝本質)
- そのため、よく使われる命令パターンは計算内容を内蔵させて単純化をしている。
- したがって、我々は簡単にプログラミングを行えるのでプログラムの本質を忘れやすい。



三角形を書いてみよう！



# 「三角形を書け」という命令

- コンピュータには「三角形」という言葉が、あまりにも抽象的で曖昧である。
- コンピュータに三角形を書かせるには
  - どのような手順で
  - どのようにして書くか
  - どこから、どこまで線を引くかなど**明確な手順**で命令しなければならない。

# 実際に三角形を書かせてみる

## プログラム

LINE 50, 50 - 250, 100

LINE 250, 100 - 120, 160

LINE 120, 160 - 50, 50

単語を日本語に訳してみると次のようになる。

## プログラム

線 50, 50 から 250, 100

線 250, 100 から 120, 160

線 120, 160 から 50, 50

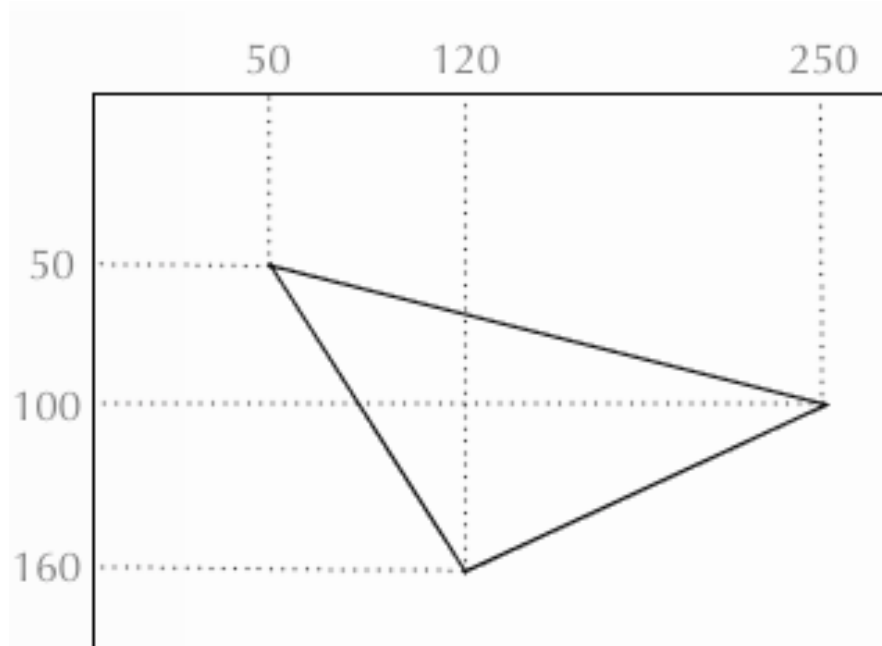
さらに普通の日本語に訳すと次のようになる。

### プログラム

画面左上から右に50ミリ、下に50ミリの位置から、  
右に250ミリ、下に100ミリの位置まで線を引け。  
画面左上から右に250ミリ、下に100ミリの位置から、  
右に120ミリ、下に160ミリの位置まで線を引け。  
画面左上から右に120ミリ、下に160ミリの位置から、  
右に50ミリ、下に50ミリの位置まで線を引け。

↳ この手順に忠実に従うと次のような絵が書かれる。

# 三角形の完成



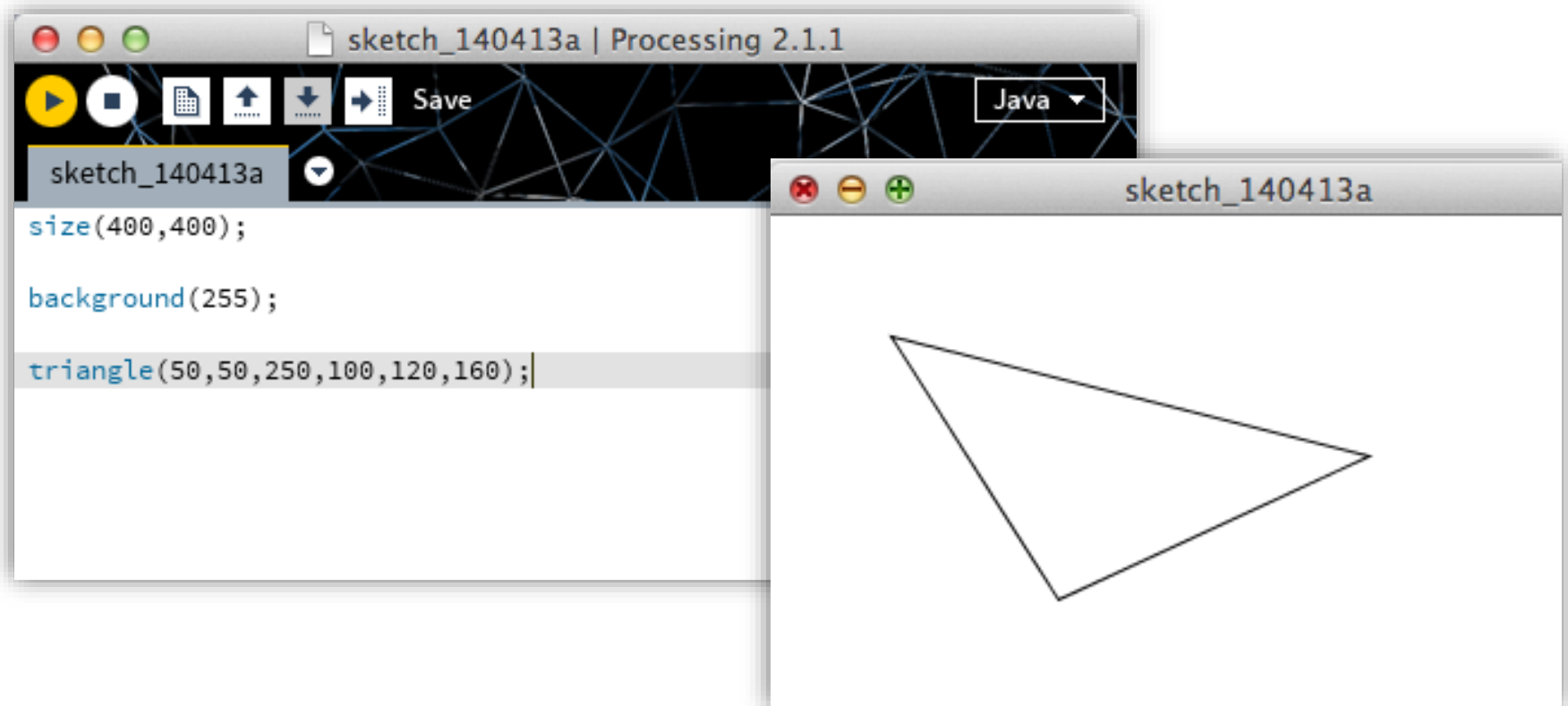
※ 1ミリを1つの点として  
考えている。

このように、極めて明確な手順で命令するのが  
プログラミング言語の特徴である。

その手順には曖昧さが全く無く、  
全ての動作をこと細かく命令しなくてはならない。

# Processingによる三角形

- プログラミング言語の一つであるProcessingで三角形を書いてみた。



# カップラーメンが食べたい

- ロボットにカップラーメンを作ってもらうには？  
↳「カップラーメンを作れ」では当然**不十分**。

棚からラーメンを取り出すのに、1m前進、右に90度回転、手を10cm前方に突き出す・・・  
といった感じで人間が**全ての動作**をこと細かく教えこんでやらなければならない。

# カップラーメンが食べたいのに...

- また、コンピュータにとっては  
個々の動作は全て独立した動作である。  
↳ カップラーメンを作る とは認識していない。
- もし動いている最中にお湯をぶちまけても、  
それは単なる動作の一つであり、ミスだとは  
認識しておらず命令通りに動き続けているだけである。



## 0.2.4 : まだまだ曖昧

- コンピュータ的には、「線を引く」というのもすでに曖昧な命令なのである。
- したがって、直線というのはどのような内容の計算なのかを人間が教えなければならない。
- 線は点の集まりであるので、一点一点について、どの点を描くかを命令しなければならない。

# 曖昧の壁

- 「線を引く」という命令に対して、完璧に明確な命令を行うのであれば、

## プログラム

DOT 50, 50

DOT 51, 50

DOT 52, 50

DOT 53, 50

DOT 54, 51

DOT 55, 51

DOT 56, 51

:

というように、「点を描け」という命令をひたすら繰り返さなければならない。

# 「点を描く」

- コンピュータの要求する緻密さはこの程度では終わらない...
- 「点を描け」ですらコンピュータにとっては  
曖昧な命令であり、コンピュータは点の計算方法を知らないのである。

⇒「点を描く」完璧な命令例

# 「点を描く」完璧な命令

## プログラム

コンピュータに接続されている00番目の装置の記憶している数値の内、32050番目の数値を1に設定せよ。

横幅が640ドットのディスプレイを例とする：

00番目の装置＝ディスプレイ（正確にはビデオカード）  
32050番目が画面左上から50,下に50の位置になる。  
1に設定せよ＝画面のx番目の位置のその点を光らせよ。  
↳ 0なら光らせないことを意味している。

といった、**完璧に明確**で誤解の余地のない命令になっている。

# 三角形を画面に書き出す計算 しかし、実際には

コンピュータに接続されている00番目の装置の記憶している数値の内、  
プログラム

LINE 50, 50 - 250, 100

：  
コンピュータに接続されている00番目の装置の記憶している数値の内、  
と命令するだけで、簡単に直線が引けてしまう。  
32690番目の数値を1に設定せよ。  
コンピュータに接続されている00番目の装置の記憶している数値の内、  
32691番目の数値を1に設定せよ。

これら全てを点を打つ場所について繰り返す必要がある。

# 先代の方々へ感謝

- すでに述べたように、先代のプログラマー様達のおかげで緻密で厳格なプログラムがすでに**内蔵**されている。
- 我々も**全く新しい**便利なプログラムを作るには緻密で厳格なプログラムを書かなければならない。
  - ↳ コンピュータを動かすには緻密で厳格な計算をさせる必要があることを**意識する**。

# 抽象化

- コンピュータの厳密さはとてつもないもの。  
しかし、これがコンピュータという機械である。  
↳ コンピュータは初めから完璧で厳密だった。
- 厳密さは十分なので、むしろもっと抽象的に  
なってほしい。  
↳ C言語はコンピュータを抽象的にする  
手法の一つである。

# 抽象化

- コンピュータの作法に従うならば、  
プログラミングも全て数字だけで行う。  
↳ C言語はデータに名前をつけることが出来る。
- C言語には数字と名前を対応付けるための表  
を自動的に用意する仕組みがあるため。  
↳ 抽象化と呼ぶ。



# 抽象化

- プログラミングの歴史は抽象化の歴史でもある。
- C言語には様々な機能があるが、  
全ての機能は、コンピュータのあまりに厳密な  
仕組みを抽象的にするためにある。

# 抽象化

- 昔のプログラマ達もコンピュータの厳密さに  
ウンザリしていたので、より抽象的にするた  
めの仕組みを作り続けてきたのである。
- これからのプログラミングでも、数値をそのま  
ま使うのではなく、常に人間にとって意味ある  
形に表現できないか考える事が重要である。

⇒それはなぜなのか？

# 抽象化

**A. 作ったプログラムを最終的に使うのは人間なのだからである。**

↳ わかりやすく、見やすいプログラムは気持ち良いですね。

# 1章：世界最小のプログラム



# 1.1 何もしないプログラム

最初からいろいろな機能を持つプログラムを作成することはできないので、まずは  
何もしないプログラムを作ることから始める

## 1.1.1: C言語の構造

- C言語のプログラムは、関数が集まって作られている。
- 注意すべき点  
関数の並びではなく、関数の集まりであること。
- C言語の関数には一部の例外を除き、順番の概念がまったくない。

## 1.1.2: 関数の作り方

- C言語の関数は非常に明確な構造を持っている。

書式: 関数の構造

|    |         |      |
|----|---------|------|
| 型名 | 関数名(引数) | {処理} |
|----|---------|------|

## 1.1.2: 関数の作り方

- **型名**とは、関数が計算結果を返す時に使う数値の種類のこと。 例: `int`
- **関数名**とは、文字通り関数の名前のこと。

### 関数名の規則

1. 半角アルファベット、半角数字、半角アンダーバーが使える。
2. 1文字目には、数字を使うことはできない。
3. あらかじめ決められた予約語は使用できない。



## 1.1.2: 関数の作り方

- 予約語とは、C言語の中で使われているキーワードである。 例: `main`
- 引数とは、関数に渡す数値の種類のこと。
- 関数は渡された数値を元に計算を行って結果を返すことができる。

# 予約語の決まり

- C言語には予約語のほかに、  
予約済み識別子も名前として使用できない。
- この予約語の数は少ないので大きく気にすることはない。

# 予約済み識別子

- 予約済み識別子は、内部で使用されている名前のこと。
- アンダーバーに大文字が続く名前や、C言語で標準的に使われている名前は使用できない。

## 1.1.2: 関数の作り方

- 何もしないプログラムを作るので、数値を渡す必要が無い。ここでは数値がないことを表す `void` を使用する。
- `処理` とは、その名の通りである。何もしないプログラムにおいて、唯一必要な処理が関数を終了させることである。

## 1.1.2: 関数の作り方

- 関数を終了させるため、**return文**を使用する。
- return文には計算結果の数値を返す機能があるが、何もしないプログラムのため0にしておく。

プログラム

```
int main(void) {return 0;}
```

## 1.1.3 : main関数は特別

- 関数には順番が存在しない。  
→どの関数から目をつければ良いのか迷ってしまう。
- C言語ではmainという関数が、最初に動作すると決められている。
- プログラムの中にmain関数がどこにもない場合、そのプログラムは動作できない。

## 1.2 コンパイラは翻訳ソフト

プログラムを動作させることについてより詳しく説明をする

## 1.2.1:すべては機械語

- コンピュータは2進数で動いている。  
→2進数で動いているコンピュータがなぜ文字列を認識して動くのか。
- 本来、コンピュータが理解できるのは、2進数で書かれた命令、マシン語（機械語）だけである。



## 1.2.1: すべては機械語

プログラム: 機械語のイメージ

```
000101001101011001100010110000110  
011101110100110011110100001000101
```

- 上記はデタラメに0と1を並べただけだが、イメージとしてはこれで間違いない。
- どんなコンピュータでも、この形の命令しか認識できない。

## 1.2.2: プログラミング言語の登場

- コンピュータ黎明期: トグル式スイッチで直接膨大な量の0と1を入力していた。  
→ 手間がかかりすぎるため、数字の桁を縮めることに。
- 2進数4桁分を1桁で表現した16進数が考えられる。

## 1.2.2: プログラミング言語の登場

- また、数字1つ1つを記号化したアセンブラも考えられた。

プログラム: アセンブラ

|     |         |
|-----|---------|
| MOV | AH , BH |
| ADD | AX , 70 |
| JPN | AF , 01 |

- この記号化によって、プログラミングは非常に効率的になった。

## 1.2.2: プログラミング言語の登場

- 普通の人にはわかりにくい記号であったのを大きく変えた世界初のプログラミング言語、**FORTRAN**。

プログラム: FORTRAN

```
DO 10 I=1, 10000  
  READ *, X  
  IF (X, GT, MAX) MAX=X  
10 CONTINUE
```

- これら英単語や数式が使える言語を**高級言語**という。

# 高級言語

- プログラミング言語までになると人間にわかりやすくなり勉強すれば理解できるようになる。
- C言語も、英単語や数式が使える言語、**高級言語**のひとつとして数えられている。
- しかし、機械語へ翻訳が必要。
  - ↳ 翻訳方法
    - インタプリタ方式
    - コンパイル方式

## 1.2.3 : C言語翻訳ソフト

- インタプリタとは、プログラムを実行可能な形式に変換しながら実行する方法。**速度が低速**。
- **コンパイル**とは、プログラム全体を一括して機械語に変換し単独のソフトウェアにする方法。翻訳を行うソフトを**コンパイラ**という。
- C言語では主にコンパイラが使われている。

## 1.2.3 : C言語翻訳ソフト

- コンパイラは3段階の仕組みで動作している。

### コンパイラの仕組み

プリプロセッサというソフトにより、文字列の調整を行う。

↓

コンパイラによってコンパイルされる。

↓

翻訳した機械語プログラムがリンカというソフトによって結合される。

## 1.2.3 : C言語翻訳ソフト

- 文字列の調整では、空白や改行の結合、記号の置き換えなどを行いプログラムを解析しやすくする。
- コンパイルされるとき、より高速に工夫して機械語に翻訳する機能、最適化が行われる。
- リンカで結合することをリンクと呼び、リンクされることで機械語のデータが実行可能ファイル(EXEファイル)になる。



# プログラムを動作させる

- 作成した何もしないプログラム

プログラム

```
int main(void) {return 0;}
```

- これを実行させると何も表示されない。

ご清聴ありがとうございました。