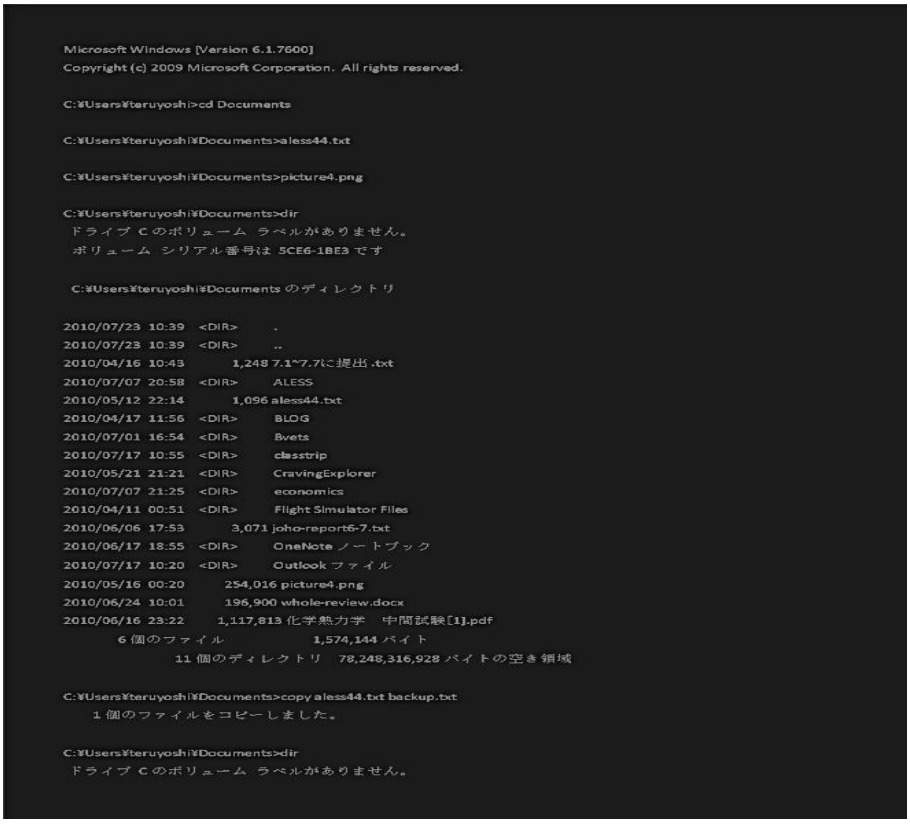
情報シケプリ　　藤垣先生の個別の部分について

## １．GUI と CUI について

皆さんがコンピューターの前に座って何かするとき、ただほげーと画面を眺めているわけではないですよね。何かしらコンピューターに指示を送っているはずです（例えば、マウスでアイコンをクリックする、とか)。この指示のことを、コンピューターへの入力といいます。Excel に数式を入れて行ったり、アイコンをクリックしてフォルダを開いたりと、様々な入力があります。でも、これらの入力のやり方（システム）を大きく 2 つに分けると、GUI と CUI に分けられるのです。

昔の PC は、主に CUI が用いられていました。イメージとしては、こんな感じです。そういえば、




プログラミングの時にも使いましたね。

こういう真黒な画面に文字を打っていって、このテキストファイルをコピーしろ、などという命令を実行していくわけです。例えば、aa.txt というファイルを backup.txt という名前でおんなじ場所にコピーしたいなら、

C:¥Users¥teruyoshi> copy aa.txt backup.txt

という風にこの黒画面に入力します。また、aa.txt を開きたいなら、

C:¥Users¥teruyoshi>　aa.txt

と打てばいいわけです。すると、お望みの操作が完了します。しかし、この方法だと、この命令（コマンド、といいます）を覚えておくか、いちいち本か何かを参照しなければならず、初心者には敷居が高いですね。というわけで、GUI の登場です。これは、graphical-user-interface の略ということからもわかると

おり（？）、視覚的に入力をやってしまおう！、というものです。例えば、先ほどの、ファイルをコピーするという動作は、WINDOWS 上などでは、ファイルを右クリックして、コピーというタブをクリックすれば、どこにでもそのファイルのコピーを置くことができます。aa.txt を開きたいなら、そのアイコンをダブルクリック（環境によって異なりますが）すればよいのです。その時、画面上に自分がやりたいことが書かれているので、面倒な CUI のコマンド―ここでは、(copy <ファイル名>　<ファイル名>) や (aa.txt) ―を覚えなくて済む、という利点があります。また、暗喩を上手に用いることによって、入力をわかりやすくしています。WINDOWS ではデスクトップにゴミ箱というアイコンが置いてあって、そこにファイルをドラッグ＆ドロップすることで、そのファイルを削除することができますが、ファイル削除という行為をごみを捨てる、ということになぞらえているという点で、良い例として挙げられるでしょう。では、GUI はいいことづくめなのでしょうか？そうとも限りません。わかりやすい欠点としては、実際に動いているプログラムや処理が CUI だと比較的理解しやすいのに対して、GUI だと分かりにくい、というのがあります。GUI でボタンを一つ押したとき、コンピューター側はあらかじめ決められた規則にのっとって様々な処理を行います。もちろん、GUI もプログラミングによって作られているので、内部ではプログラムが動いていますから。CUI だと、プログラム実行のためにはそのプログラムに対応したコマンドを打たなければなりませんが、GUI だとボタンをクリックしただけでいろんなプログラムが実行されていくので、内部で何が起きているのかがわかりにくいでしょう。ほかにも、描画をする分だけ処理が重くなったりということもあります。現在のパーソナルコンピューターではほとんどが GUI システムを採用していますが、WINDOWS ではコマンドプロンプトなどの機能によって、CUI で操作することも可能です。また、UNIX などの OS には CUI を採用しているものもあります。

まとめ

CUI…コマンドを打って入力とするシステム。複雑かつ根本的な処理を行いやすいが、画面が文字だけで構成されるためわかりにくく難しい。

GUI…画面を見ながらマウスなどを使用して直観的・視覚的に操作可能。様々な情報をわかりやすく表示できるので、扱いやすい。一方、複雑な処理は苦手。

**2. 単語集**

○CPU…Central-Processing－Unit とは、中央処理装置と訳される、コンピューターの部品のうち最も重要な物の一つ。。CPU はレジスタ（CPU 内にある高速なメモリ）からデータ（注）を読み込み、内部で処理した後に、レジスタにデータを出力する。さらにその繰り返しで、様々な処理を行っていく。コンピューターの頭脳ともいえる部分。

（注）ノイマン型コンピューターでは、命令もデータとしてレジスタに格納されている。

○CPI…消費者物価指数（Consumer Price Index）は、消費者が実際に商品を購入する段階での、商品の小売価格（物価）の変動を表す指数。天候によって価格が左右される生鮮食料品を除いたものを、コア CPI という。さらに、エネルギー価格の変動も無視するため、コアコア CPI も用いられる。**ではなくて、**情報

の分野では、Cycles Per Instruction の略で、コンピュータが CPU の 1 命令を実行するのに必要なクロック数のことである（らしい）。クロックって？そうですね、クロックというのは、コンピュータ（またはデジタル回路）が動作する時に、複数の電子回路のタイミングを取る（同期を取る）ために使用される周期的な信号のことをいいます。CPU やら、メモリーやらが適当にデータを出力したり、入力したりしていては、ぐだぐだになってしまいますから、合図とともに、一斉にデータを次のところへ渡すのです。イメージとしては、自動車工場なんかの製造ラインを想像してください。例えば、自動車の組み立て、塗装、タイヤの取り付け、と並んでいて、車のボディーが各部署を流れていくと思ってください。この時、たとえタイヤの組み込みが早く終わったからと言って次の部署である座席の取り付け部門にちゃっちゃと回してしまったら、座席取付部門は困ってしまいます。そこで、全部の部門の処理が終わると同時にブザーを鳴らし、そのブザーに従って一斉に次の部署にボディーを回しましょう、というのがクロックの考え方です。電子部品ですから、そのクロックは 800MH z （＝一秒に 8 億回ブザーを鳴らす）などと、かなり高速ですけどね。つまり、CPU にデータを渡して、5 個分のクロックでようやくデータの出力があるとき、CPI は 5 だというのでしょう（確信はないけど）。

○メモリ…名前の通り、コンピュータのデータを蓄えておく場所ですが、メモリは一時的なデータの保存に使われます。CPU が処理するデータはレジスタに格納されているのですが、このレジスタ、あまり容量が多くありません。だから、CPU の外（そと）に容量の大きいデータを格納する場所を作りました。それがメモリです。しかし、CPU 内部にあるレジスタと違ってデータの転送速度が遅いので、使用される頻度が高いデータや、次に使うことが予想されているデータはあらかじめレジスタに転送されます。しかし、大容量化が進んでいるメモリといえどもすべてのデータを扱えるわけではありません。また、メモリ・レジスタには、電源を入れておかないとデータを保持し続けられない、という性質があります。したがって、HDD（Hard-Disc-Drive）にデータを格納するのです。これは、容量は馬鹿でかい代わりに転送速度は遅いです。あと、電源を切ってもデータを保持し続けられます。例えるなら、図書館の自習室で（今僕がいるところでは）勉強するときに、レジスタは目の前に貼ってあるポストイット、メモリは手元に置いてある辞書、HDD は本棚にある分厚い本、といった感じです。付箋→辞書→本棚と内容は大きくなっていきますが、すぐには目的のものが見つからなくなっていくでしょう。

～データの保存場所のまとめ～

| | データ転送速度 | 容量 | データ保持に電源 | 場所 |
|---|---|---|---|---|
| レジスタ | 速い | 少ない | 必要 | CPU 内部 |
| キャッシュ | ⇳ | ⇳ | 必要 | （今は）CPU 内部 |
| メモリ | | | 必要 | MB 基盤上 |
| ハードディスク | 遅い | 多い | 不必要 | MB 基盤の外部 |

（本来はレジスタとメモリの間の存在としてキャッシュなるものが存在する（しかも、一次・二次・三次くらいまで）のですが、解説には不要なので割愛しました＾＾）

○インターフェイス…インターフェースには、ハードウェアインターフェース、ソフトウェアインターフェース、ユーザーインターフェースの三種類があります。Wikipedia によると、「ものごとの境界となる部分と、その境界でのプロトコルを指す」らしいです。過去問 06 年と 07 年に単語説明で聞かれているのでここで取り上げましたが、実際、何を応えればいいのか不明です。Google 先生にききませう。

○プロトコル…コンピュータ等の電子機器間で通信する際の取り決めのことで、どのような状況でどのようなタイプのデータをどのような順番で送るかが記載されているものです。TCP/IP の P は protocol の略ですが、これはインターネット通信の際の取り決めを決めたものです。なお、情報以外の分野では、外交儀礼、議定書なんかと訳されたりもします。

○ちなみに、プロセスについては僕の知る限り講義で触れられていなかったので割愛しまう。

### 3. WCプログラムについて

```
1:  #include＜stdio.h＞
2:  #define IN        1
3:  #define OUT   0
4:  main()
5:  {
6:      int c, nl, nw, nc, state;
7:      state = OUT;
8:      nl = nw = nc = 0;
9:      while ((c = getchar()) != EOF) {
10:         ++nc;
11:         if (c =='¥n')
12:             ++nl;
13:         if (c ==' ' || c =='¥n' || c == '¥t')          ←(A)
14:             state = OUT
15:         else if (state == OUT) {
16:             state = IN;
17:             ++nw;                                      ←(B)
18:         }
19:     }
20:     printf("%d %d %d¥n", nl, nw, nc);
21: }
```

プログラムというのは一つの言語です。しかし、英語が多用されているうえ、論理だけで構成されているので、パッと見ただけで結構わかります。基本的にプログラムは上から読まれて処理されていきます。それではさっそく、(A) ～ (B) までの意味を解説していきましょう！

まず、基本から

A＝B というのは、A が B に等しいという意味ではなく、A に B を代入しろ、という意味です。例えば、

```
1:      A=5;
2:      B=10
3:      A=B
```

とすると、1 行目で A に 5 を、2 行目で B に 10 を代入して、3 行目で A に B を代入したので、3 行目ののちは、A は 10 になっています。

では、“等しい”はどうやって表現するのでしょうか。それは、==とイコールを二つつなげて表現します。また、ノットイコールは、！=と表現します。大小関係は、< , > , <= , >=と表現されます。

次に、論理関係ですが、論理積（かつ、の関係）は、&&　であらわし、論理和（または、の関係）は ||で表します。なお、&も｜も一つだと別の意味（&はアドレス、｜はビット和）を表すので注意です。

では、++nw というのはどういう意味なのでしょう？これは、＋＋の後の変数（ここでは nw や nc など）を、この行に来たら 1 だけ増やせ、ということです。

```
1:      x = 10;
2:      ++x;
```

とすると、1 行目で１０だった x は、2 行目で 1 足されて 11 になります。簡単ですね。同じようなものに、--というのもあります。これは想像通り、ここまで来たら 1 だけ減らしてね、という意味です。

あとは、約束として、¥n　¥t　などは特殊な意味を表わしていて、¥n は改行を、¥t はタブを表します。これらはエスケープシークエンスと呼ばれ、ほかにもたくさんありますが、まあ、いいでしょう。

さて、本文中にある、

```
if （条件）｛
文 1
文 2
 }
```

というのは、() 内の条件が正しければ、そのあとの{　　}の中身のことをやってね、正しくなければ次の処理へ、ということです。ただし、｛　　｝内が一文で収まる場合、{} は不要です。それはここでも用い

られていて、{} 内が state = OUT の文だけなので、{　} は省略されていますね。

再掲（A）~(B)

```
if (c ==' ' || c =='¥n' || c == '¥t')          ←(A)
    state = OUT
else if (state == OUT) {
    state = IN;
    ++nw;
```

つまり、c が“空白（　）または、改行（￥n）または、タブ（￥t)”なら、state　に OUT　を代入しろ、ということです。

```
 else if (条件＊){
文＊
}
```

はなんとなくわかるでしょう、前の if 文で条件が正しくなかったとき、更に条件＊が正しければ、文＊を実行しろ、ということです。ここでは、c が空白でも改行でもタブ（以下、この 3 つを単語の切れ目という）でもないとき、更に、変数 state が OUT ならば、state に IN を代入しろ、ということです。

以上ですが、何をしているのやら、と思われた方も多いのではないでしょうか？初めに、nc はキーボードを打った回数で、nl は改行の回数です。それらは、c に何かを代入する回数と、 c に¥n という改行を表すものが入ってくる回数を数えることで比較的簡単に数えられるのですが、単語の数は少し面倒ですね。単語の切れ目の回数を数えて植木算的に数えようとしても、誤って空白を 2 回打っている人もいるかもしれません。

そこで、state なる変数を用意して、c が単語の切れ目の時は state は OUT になり、c が文字の時は state が IN になるようにしてやり、state が IN から OUT に代わるときに変数 nw を 1 増やしてやれば、単語の数が数えられるだろう、という仕組みなのでした。これだと、単語の切れ目を連続して 2 回打っても単語の数は変わりませんしね

注：wc プログラム全体の解説

```
1:  #include＜stdio.h＞
2:  #define IN         1
3:  #define OUT    0
4:  main()
5:  {
6:      int c, nl, nw, nc, state;
7:      state = OUT;
8:      nl = nw = nc = 0;
9:      while ((c = getchar()) != EOF) {
10:         ++nc;
11:         if (c =='¥n')
12:             ++nl;
13:         if (c ==' ' || c =='¥n' || c == '¥t')
14:             state = OUT
15:         else if (state == OUT) {
16:             state = IN;
17:             ++nw;
18:         }
19:   }
20:   printf("%d %d %d¥n", nl, nw, nc);
21: }
```

stdio.h を取り込め。
IN を 1 と定義
OUT を 0 と定義
Main 関数（具体的なプログラム）の始まり！！

int 型の整数 c,nl,nw,nc,state を定義
state に OUT を代入
nl,nw,nc に 0 を代入
c に入力された文字を代入し、c が終わりでなければ【
nc を 1 増やす
(もし、c が改行（¥n）だったら
nl を 1 増やす)
{もし c が単語の切れ目なら
state に OUT を代入
さらに、state が OUT なら（
state に IN を代入し、
nw を 1 増やす
)
}
Nl,nw,nc の値を表示しろ
main 関数終わり

○ついでに、摂氏一華氏対応表のプログラム（06 年の過去問中にあるやつ）

こっちはそんなに難しくないと思うので、さらーと流します。1 つ注意。。%3.0f というのは、少なくとも 3 桁は表示してね、ただし整数部分は最低 0 ケタね、ということです。同じく、%6.1f は、「少なくとも 3 桁は表示し、ただし整数部分は最低 0 ケタ」となります。

```
#include<stdio.h>
/* fahr=0,20, ... 300 に対して，；
摂氏一華氏対応表を印字する；
浮動小数点版*/
main()
{
   float fahr，celsius；
   int lower，upper，step；
   lower = 0； /*温度の下限 */
   upper = 300； /*上限 */
   step = 20； /* きざみ */
   fahr = lower；
   while (fahr <= upper) {
      celsius = (5.0/9.0) * (fahr-32.0)；
      printf("%3.0f %6.1f ¥n"，  fahr，celsius)；
      fahr = fahr + step；
   }
}
```

stdio.h を取り込んで読み込め
コメント（/*文章*/とすると、その文章はコメントとみなされ無視される）
コメント終わり
main 関数の始まり！！

float 型の変数、fahr,celsius を定義
int 型の変数、lower,upper,step を定義
lower に 0 を代入
upper に 300 を代入
step に 20 を代入
fahr に lower の値を代入
【もし、fahr が upper より大きくないなら
celsius に、(fahr*32)*5/9 を代入←セ氏カ氏の対応
fahr,celsius の値を表示せよ
fahr の値を step の分だけ増やせ
】←while の括弧（ここまで来たら、【からまたやる）
main 関数終わり