基礎演習 III テキスト
part 1: プログラミング

三木　邦弘

平成 7 年 4 月 12 日

目 次

1　はじめに

平成7年度の私の基礎演習では、次の2つの事を行います。

- 簡単なプログラミング
- ワークステションを使ったネットワークの利用

このテキストではCを使用したプログラミングについて学びます。

今年で私の基礎演習も4年目になります。最初の年はBASICを使用したプログラミングをやりました。次の年はCを使用したプログラミングをやりました。昨年からCを使用したプログラミングに加えてワークステーションの利用も内容に加えました。コンピュータと少しでも深く付き合っていくには、プログラミングやその考え方を理解する必要があります。そのために3年次には「プログラミング演習」と言うものもあるのですが、内容が重複するなどの問題が出てきました。そこで今年は少し視点を変えて、少し内容を見直してみました。

本来ならば何かCの解説の本を買って頂いてそれを元に演習を進めたいのですが、なかなか適当な本がありません。Cを使用しますが、文法的な事項は最低限に限定します。昨年のこのテキストの表題は「基礎だけのC言語」でしたが、今年はさらに絞り込みました。要するに文法的な事項以外の点をこの演習では学べるようにしようとしています。すると通常の本では一通りCについて述べなければならないためか、余分な説明が多過ぎます。と言うわけでせっせとテキストを作成したのですが、もっと勉強されたい方は、Cに関する本は非常に多く出版されていますので、本屋で適当な本を購入して勉強して下さい。

2　Cについて

2.1　プログラミング言語とは

コンピュータはプログラムと言う形で与えられた指示を非常に高速に忠実に実行する事ができます。コンピュータが基本的にできることは数値の演算に限られるのですが、それらを高速に多様に組合わすことによって様々な仕事ができます。我々は既に何らかの用途に合わせたプログラムを内蔵したコンピュータをよく利用します。ワープロは文書編集用のプログラムが組込まれたコンピュータで、ファミコンはゲームを実行するためのプログラムが組込まれたコンピュータです。このようにプログラムが組込まれたコンピュータは電源スイッチをオンにするだけで使うことができますが、それ以外の用途には使えません。逆にパソコンを初め通常コンピュータと呼ばれているものは、何か仕事をするためのプログラムは持っていませんが、プログラムを実行するための用意はできているので、仕事をするプログラムを入れてやればその仕事をするようになります。

コンピュータに何か仕事をさせたい時には、それをするためのプログラムが必要です。既に多くのプログラムが市販されていますので、大抵の仕事はそれを購入することにより済ますことができます。しかし仕事の内容によっては、一部だけ市販のプログラムでは合わない部分がある事はよく生じます。また全く対応できるプログラムがないとか、貧乏なのでプログラムを購入できない事もあります。そう言った場合の対処の仕方として自分でプログラムを作成する手があります。

プログラムはコンピュータに何をやらせるか、とか仕事をどのように処理するかを記述したものです。具体的にかつ詳細に論理的に記述するために様々なプログラミング言語が考えられました。各言語はそれぞれが記述しようとするプログラムの対象をある程度想定し、対象が合えば楽に記述ができるようになっています。数値計算ならばFORTRAN、事務用ならばCobol、知識情報処理ならばPrologなどが有名です。もちろんFORTRANで全てのプログラムを記述するのも不可能ではありませんが、向いていない分野のプログラムを書くのは、記述が長くなったり複雑な手法が必要になります。

2.2 Cの生立ち

CはUnixと言うオペレーティングシステム(OS: Operating System)を記述するために生まれました。OSは基本ソフトとも訳されますが、他の実際に仕事をするプログラムと異なり、コンピュータシステムを管理・運用するためのプログラムで、通常のプログラムはこれの助け無しでは働くことができないという重要なものです。OSを記述するためにはコンピュータの非常に基本的な動作まで記述できる必要もあります。そのためにCは他のプログラミング言語では扱うことができないような、コンピュータの生の情報を扱えるようになっています。

Cは1972年にアメリカのベル研究所のDennis Ritchieによって開発されました。ただし単独で全く無の状態からCが作られた訳ではなく、Algol60、CPL、BCPL、Bと言う名前のプログラミング言語の系列の最後にできました。当時のベル研究所ではBを使用してUnixを開発していましたが、幾つかの点で問題があったため、Bを拡張する形でCが誕生しました。それ以降Unixの大部分はCで記述され、開発されたUnixは教育機関には無料で、他の企業にも極めて安価にて供給されたので急速に普及しました。

80年代になると、パソコンがより強力になりかつ普及してきました。当初パソコンにおいてはBASIC言語がよく使われました。これは大抵のパソコンに無料で添付されていたためと、初心者向きの簡単な構造を持っていたためと思われます。一方業務用等のパソコンの性能を限界まで引き出さなければならない分野ではコンピュータ本来の命令に近いアセンブリ言語が使われて来ました。C はまず後者のような分野で使われるようになりました。これはアセンブリ言語では大規模なプログラムの開発が困難である事や、OSのようなプログラムでも記述できるCの良さが認められて来たからだと思います。そしてそれにつられるような形で普通のプログラムもCで書かれる事が多くなりました。

現在コンピュータネットワークでよく用いられるワークステーションのOSはUnixです。Unixには標準でCが使用できるようになっているので、様々なプログラムがCで書かれています。

3 学園センターでCを利用するには

学園センターでCを利用するには、次のような手順をふむ必要があります。また現在は実習室1でのみCは利用可能です。以下画面及びキーボードでは¥のところが印刷の都合で全て\になっています。また改行のキーを押すところは△の記号を使っていますので間違えないようにしてください。また␣はスペースを示していますので忘れないでください。

1. Cの利用環境の設定
   - 電源スイッチを押す。
   - メニューが出たら ESC を押す。プロンプト(`C:\>`)が表示される。
   - フロッピーを左側のドライブに入れる。
   - a:makeup△ と入力する。ちょっとすると再びプロンプト(`A:\>`)が表示される。
2. 終了の仕方
   - bye△ と入力する。
   - 電気が切れたらフロッピーを取り出す。
3. プログラムの編集の仕方
   - red␣ファイル名.c△ と入力する。ファイル名は、英字、数字と一部の記号よりなる8文字以内のもの。

- 矢印キーで修正したい所へカーソルを移動し、入力や削除を行う。
- 終了するには、PF1を押し、1(save)を押す。
- 前回と同じファイルを編集する場合には、red␣-q△ と入力する。

4. プログラムのコンパイルの仕方
   - lcc␣ファイル名△ と入力する。
5. プログラムの実行の仕方
   - ファイル名△ と入力する。

［演習］

以下の手順で簡単なCで書かれたプログラムをコンパイルして実行してみて下さい。

```
A:\>red test1.c△
```

入力するプログラム

```
#include <stdio.h>△
main(){△
   printf("This is a program.\n");△
}△

A:\>lcc test1△
cpp -DLSI_C -IE:\LSIC\INCLUDE -j -o E:1.$$$ test1.c
cf -chut j E:1.$$$ E:2.$$$ E:3.$$$
cg86 -v E:2.$$$ E:3.$$$
main_ _...:;
r86 -o E:test1.obj -m test1 E:3.$$$
lld @link.i

A:\>test1△
This is a program.
```

4　タイプの練習ソフトについて

パソコンの標準的な入力装置はキーボードです。プログラムやデータの入力は基本的にこれを用います。そのためにパソコンを使いこなすには、キーボードで早く入力ができることが必須の技能となります。欧米ではコンピュータができる以前よりタイプライターが存在し、タイピストと呼ばれる専門家以外でも多くの人がこれを利用してきました。日本では最近こそワープロが普及していますが、ほとんどの利用者がキーボードのキーを探しながらキーを押しているような状況です。

理想的にはキーボードを全く見ないで打てるのが良いのですが、キーの押しにくい周辺のキーは確認して押すレベルでも、全く練習していない状況よりはましです。キーボードを全く見ない状況とは、キーの位置を全て憶えている状態ですが、いちいちキーの位置を意識の上で考えるのではありません。条件反射でキーを思うだけで反射的に指が動くようにするのです。

4.1　タイプ練習ソフトの起動方法

タイプの練習ソフトは各自のフロッピーに入っています。起動する際には、前節の「Cの利用環境の設定」と同じ事をします。その後で、

```
A:\>mikatype△
```

と入力します。するとプログラムが起動されて表題の画面になりますので、もう一回改行キーでも押しますと、メニューが出てきます。これ以降はやりたい項目の先頭に書いてある番号を入力するだけです。

練習が終わりましたら、前節の「終了の仕方」と同じ事をします。その前に続けてCのプログラム入力や実行などをしてもかまいません。

タイプ練習用のプログラムはフロッピーに入っているので、実習室１以外のパソコンでも実行可能です。

4.2 タイプの練習の進め方

やはりピアノが上手とか、運動神経に自信のある方は上達が早いようです。しかし普通のひとならば誰でも練習に応じて上達します。まだ20歳にもならない若さを信じて頑張ってください。

進め方は基本的に最初に表示されるメニューの順番に従って下さい。まず１番の「ポジション練習」でどの指でどのキーを押すのか憶えます。「ポジション練習」のメニューでは、やはり１番から順番にやってください。画面上部に押すべきキーが表示されますので、最初のうちはその下のキーボードの図や、その下のそれを押すべき指の表示を確認しながら、その通り押してください。両手にそれぞれ5本の指が無い人以外は必ずそれに従って下さい。

「ポジション練習」は1回につき60個のキーの練習ができます。終わったところで、まだ足りない人は改行キーをおせば続けて練習ができます。終わりたい人はESCキーを押せばメニューに戻ります。練習を続ける際は、1回終わる度に少しパソコンのディスプレイから目を離して休憩を取りましょう。

「ポジション練習」で1つマスターできた様ならば、「ランダム練習」に同じメニューがありますので、そちらに挑戦して見てください。とりあえず目標を70文字／分以上としてください。目標をクリアーできたら次のキーの位置を「ポジション練習」で憶えて下さい。

「ランダム練習」の全てのメニューで目標をクリアーできた方は、「英単語練習」のメニューに進んで下さい。ここでは「基本英単語練習」から「８０８６アセンブラ練習」まであります。大文字ばかり出てくるものもありますが、そのままキーを押せば済みます。これらに関しては100文字／分以上を目標にして、全てのメニューをクリアーしてください。

さらに「ローマ字練習」もありますので、余裕のある方は挑戦してみて下さい。間違って入力した場合、最初の子音の部分なのか母音の部分なのかわかりにくいので苦労するでしょう。また1文字当たり平均2つのキーを押さなければならないので、記録は「ランダム練習」の約半分になるでしょう。

ここでは目標を速度にします。もちろん「ポジション練習」以外でキーボードを見ながらやってはいけません。反射神経育成のために少々間違えてもどんどんキーを押してください。

メニューの「成績」のところでこれまでの記録や練習時間を見ることができます。と言うことは、ちゃんと練習をしたけれども遅い人と、さぼっているので遅い人との区別ができると言うことです。

4.3 練習ソフトの問題点

このタイプ練習プログラムはフリーソフトです。作者の指定した条件を守れば、このように無料で配布しても著作権法等に触れる心配はありません。このプログラムは様々な機種のパソコンでも動作可能なように配慮して作ってありますので、本来の対象機種はNECのPC98シリーズですが、学園センターのFMRでも利用できます。画面に若干の乱れが見られますが大きな問題ではありません。

そのような利点の反面、この練習ソフトでは文字と数字キーのみを練習の対象としているので、プログラムの入力の際に必要な記号の入力の練習ができない欠点があります。英文にしろ和文にしろ入力の際には最低句読点の入力は必要ですので、改善が望まれます。

5　初歩のプログラミング

5.1　Cのプログラムの形

今の段階ではCのプログラムは必ず次のような形をしているものと憶えてください。

```
#include <stdio.h>

main(){
  文がいくつか
}
```

最初の#includeはコンパイラが持っているstdio.hと言うファイルをここで読み込む事を指示しています。このファイルの中には通常のプログラムに必要な様々な関数や定数の定義が入っています。

main(){はここからmainと言う名前の関数の定義が始まることを示しています。関数については後で詳しく説明します。Cのプログラムには必ずmainと言う名前の関数の定義が存在し、プログラムを実行するとこの関数が最初に実行されます。

その次の「文がいくつか」の部分には様々なCで言う文が;で区切られて並びます。原則的に書いた順番に実行されます。どのような文があり、どのような働きをするのかを憶えないとプログラムは書けません。しかしそれほど種類はありませんから、憶えることには問題は生じません。いかに組合わせれば目的とする動作をするのか考えるのが難しいのです。

最後の}を忘れてはいけません。必ず{ }や( )は対になっています。この}はmainの定義の終わりを意味します。

5.2　printfの使い方

コンピュータは別名電子計算機とも呼びますので、計算の仕方からやっても良いのですが、計算の結果を画面に出す方法が判らなくては正しく計算できたのかどうかも判りません。Cではprintf()と言う関数を文字列や計算結果の出力に用います。printf()の全ての機能を一度に憶えるのは大変ですので、少しずつ分けて説明します。

まず一番簡単な形としてprintf(”何でもOK”)があります。( )の中に” ”で囲った文字列を入れるとその入れたものだけが、ほぼそのまま画面に表示されます。ここで「ほぼそのまま」と言ったのは、\や%については特殊な意味があるので、そのままの形ではでないからです。とりあえずここでは\nだけ憶えてください。\nがあると画面上のカーソルが次の行の先頭に動き、以後の文字列は次の行の先頭から表示されるようになります。(通常これを改行と呼びます。) 3章の演習では、printf(”This is a program.\n”)となっていたので、This is a program.と表示された後で改行がなされています。

［演習問題］

画面に次のような物を表示するプログラムを作れ。(hello.c)

以後の演習問題にも( )の中にプログラムを入れるファイル名を指定しますので、できるだけそれに従って下さい。

```
A>hello△
*********
* HELLO *
*********
```

5.3 整数型変数

コンピュータの基本的な機能は数値の計算です。複数の値の計算や複雑な計算式の実行には、途中の計算値を記憶する機能が必要です。そのためにどのようなプログラミング言語でも変数と呼ばれるものが利用できるようなっています。

整数型変数は整数を記憶することができます。パソコンのCでは通常-32768から+32767の範囲の値が記憶可能です。複数の変数を使い分けるために変数には全て名前(変数名)が付いています。変数名は英字、数字、下線(_)、で8文字程度以下にします。変数名の先頭は英字でなければばりません。

一部の言語を除き通常のプログラミング言語では、変数を利用する前に宣言をしなければなりません。Cの場合、int a,b,c; と宣言するとa、b、cと言う名前の整数型変数が利用可能になります。

実際の変数の利用は次のような感じになります。a=3; 変数aに3を入れる。a=3+5; 加算、a=5-3; 減算、a=5*3; 乗算、a=15/3; 除算、a=16%3;剰余（余り）、b=3; a=b+3; aに6が入る、a=3; a=a+3; aに6が入ります。最後の例では=の両側にaがあります。同じaですが左側のは結果を入れる場所、右側のは変数の値を意味しています。この他に通常の数式同様に( )を使用して演算の順序を指定する事も可能です。

こうして変数に入れた値を表示するにはprintfの次のような機能を使います。

```
printf("%d",a);  aに入っている値が表示される。
```

%はこの後に出力形式の指示があることを示しています。dは10進数を示します。さらに、printfの出力形式の指定の際に%とdの間に桁数を指定する事ができます。%4dとなっていれば、4桁以下の数値の出力の際には数字の左側に空白が補われます。

aの値	%d	%4d
1	1	␣␣␣1
12	12	␣␣12
123	123	␣123
1234	1234	1234

［演習］

次のプログラムを入力し、実行してみよ。(printf.c)

```
#include <stdio.h>

main(){
   int i,j;

   i=3;j=5;
   printf("i+j=%d  i-j=%d \n",i+j,i-j);
}
```

上記のプログラムを実行すると、

```
i+j=8  i-j=-2
```

のように表示されます。このように変数の代わりに式を書いても良いし、複数の値を一度に表示する事もできるし、%d以外の部分はそのまま表示されます。

5.4 for文

for文はC言語で非常によく使われるもので、処理のくり返しを表現する代表的なものです。その形式と動作は次のようになります。

```
for ( A ; B ; C ) D ;
```

まずAを実行する。Bの条件を調べて、だめならば、for文の実行を終えて次の文へ。OKならばD、そしてCを実行して、ふたたび条件判定のところへもどる。このような動作をします。具体例は次のようになります。

```
for (i=0;i<100;i++) printf("%d\n",i);
```

ここでi++と言う表記が出てきましたが、これは変数iの内容を1増やすと言うC独特の式です。まず変数iの値がゼロになります。条件判定は変数iの値が100未満か？となっているので、変数iの値が100以上になったらおしまいです。100未満ならば、iの値を表示して改行する。そしてiの値を1増やして条件判定にもどる。この繰り返しになります。

注意点として、Dの部分が複数の文からなる場合には{ }で囲う必要があります。次の例で左側はHaHaHaとCyaCyaCyaがそれぞれ交互に10個出ますが、右側ではHaHaHaが10個出た後に1回だけCyaCyaCyaが出ます。

```
for (i=0;i<10;i++) {
   printf("HaHaHa\n");
   printf("CyaCyaCya\n");
}
```

```
for (i=0;i<10;i++)
   printf("HaHaHa\n");
printf("CyaCyaCya\n");
```

［演習］

次のプログラムを入力して実行してみよ。ただし実行する前にどのような結果が得られるかよく考えてみること。例えば最初のfor文では何が得られるか？次のfor文ではcatを出している様だがどのような形に並ぶか？予想と異なる結果が得られた場合にはよくその原因を考えること。(fort.c)

```
#include <stdio.h>

main(){
   int i;

   for (i=1;i<10;i++) printf("%d %d %d\n",i,i*i,i*i*i);
   for (i=0;i<200;i++) printf("cat ");
}
```

［演習問題］

1. 以下の図形を縦横5匹？ずつ計25匹表示するプログラムを作れ。(tanuki.c)

```
 O O
(o.o)
=( x )=
 U U
```

2. 次のような九九の表を出力するプログラムを作れ。(kuku.c)

```
A:\> kuku△

   ***  The Multiplication Table  ***

   1   2   3   4   5   6   7   8   9
   2   4   6   8  10  12  14  16  18
       中略
   9  18  27  36  45  54  63  72  81
```

5.5 条件判断(if)

コンピュータは計算だけでなく判断することが可能です。もちろん人間のように玉虫色なファジーな判断は無理ですが、yesまたはnoの論理的な判断をし、その結果によって異なる文を実行することができます。

```
if（条件）A;
```

条件が成立したときのみAが実行されます。

```
if（条件） A; else B;
```

条件が成立するとAが実行されて、そうでない時にはBが実行されます。AとBともに複数の文からなる場合には{ }で囲みます。条件としては、

x<5	xの値が5より小さい。	x==5	xの値が5と等しい。
x!=5	xの値が5と等しくない。	x<=5	xの値が5と等しいか少ない。
(x==3) || (x==5)	xは3に等しいか5に等しい。		
(x>5) && (y<3)	xは5より大きく、かつyは3より小さい。		

などがあります。2つの値を比較するのが基本で3つの数が等しいかどうかを判断する際には、2つづつ比較するようにしなければなりません。つまりx、y、zがみな等しい条件は、x==y==zではなく、必ず(x==y) && (y==z) と書かねばなりません。

5.6 入力用関数(scanf)

キーボードから入力した数値などを変数に入れるためには、scanfと言う関数が使えます。

```
scanf("%d",&i);
```

&は、アドレスを求める演算子です。&iで、変数iが実際にメモリーのどこにあるかを示します。scanfは入力されたものを10進の数値とみなして解釈し、与えられた変数の位置(アドレス)にそれを書き込みます。

［演習］

以下のプログラムを入力し、実行してみよ。(hantei.c)

```
#include <stdio.h>

main(){
   int h,w;

   printf("Height (cm) = ");scanf("%d",&h);
   printf("Weight (Kg) = ");scanf("%d",&w);
   if ((h-100)*0.9>w) printf("You are smart!\n");
    else              printf("You are too heavy!\n");
}
```

scanf関数自体は何も画面に出力を行わないので、この例のようにscanfの前にprintfを使用して、何を入力しようとしているのか示すのが普通です。

［演習問題］

1. 身長と収入を尋ねて、それぞれ170と1000以上ならば、I love you.と答えるプログラムを作れ。条件に合わない場合には適当なメッセージを出す事。(2kou.c)

```
A>2kou△          A>2kou△                        A>2kou△
Height=177△      Height=150△                    Height=190△
Income=1500△     Income=2000△                   Income=600△
I love you.      Bye bye, you are too small.    Bye bye, you must work hard.
```

2. 3つの数値を入れたら、その中で最大のものを答えるプログラムを作れ。(max3.c)

```
A>max3△          A>max3△          A>max3△          A>max3△
A=5△             A=7△             A=7△             A=8△
B=6△             B=3△             B=12△            B=5△
C=7△             C=4△             C=8△             C=8△
Max=7            Max=7            Max=12           Max=8
```

［応用問題］

1. 3つの数値を入れたら大きいものの順に表示するプログラムを作れ。(sort3.c)

```
A>sort3△         A>sort3△         A>sort3△         A>sort3△
A=5△             A=7△             A=7△             A=8△
B=6△             B=3△             B=12△            B=5△
C=7△             C=4△             C=8△             C=8△
Answer=7, 6, 5   Answer=7, 4, 3   Answer=12, 8, 7   Answer=8, 8, 5
```

2. ｘの字で直角三角形を表示するプログラムを作れ。その大きさは入力して指定する。(3kaku.c、3kakua.c、3kakub.c、3kakuc.c)

```
A>3kaku△         A>3kakua△        A>3kakub△        A>3kakuc△
Size=8△          Size=7△          Size=6△          Size=5△
x                xxxxxxx             x             xxxxx
xx               xxxxxx             xx              xxxx
xxx              xxxxx             xxx               xxx
xxxx             xxxx             xxxx                xx
xxxxx            xxx             xxxxx                 x
xxxxxx           xx             xxxxxx
xxxxxxx          x
xxxxxxxx
```

5.7 MS–DOS

通常のパソコンは、基本ソフトウェアとしてMS–DOSを使用しています。これは、キーボードやディスプレイの管理の他に、フロッピーやハードディスクのファイルの管理を行っています。通常のプログラムはこのMS–DOSによってディスクからメモリーに読みだされ、MS–DOSの助けを借りながら実行されます。

MS–DOSで使用できるファイル名の形式は、ファイル名の本体として半角で8文字までと、ファイル名の拡張子として半角で3文字までが使用できます。拡張子は省略する事が可能です。拡張子のうちのいくつかはMS–DOSによって特別な意味を持っています。またプログラムによってはある特定の拡張子を持ったファイルのみを扱うものがあります。

MS–DOSにとって特別な意味を持つ拡張子は、exe、com、batです。また通常Cで書かれたプログラムはcと言う拡張子の付いたファイルに保存する事になっています。

以下はMS–DOSのコマンド (命令) の幾つかの例です。

A> dir△	Aドライブにあるファイル名の表示。
A> type␣test1.c△	ファイルtest1.cの内容を表示する。
A> copy␣test1.c␣abc△	ファイルtest1.cをabcと言う名前のファイルにコピーする。
A> ren␣abc␣def△	ファイルabcをdefと言う名前に変更する。
A> del␣def△	ファイルdefを消去する。

通常は何かファイルに対して処理を行う場合には、それぞれのファイル名を個別に指定するのですが、ワイルドカードと呼ばれる複数のファイルを一度に指定する方法があります。

```
A>copy *.c b:
```

これはは拡張子がcである全てのファイルをBドライブにコピーします。

```
A>del test1.*
```

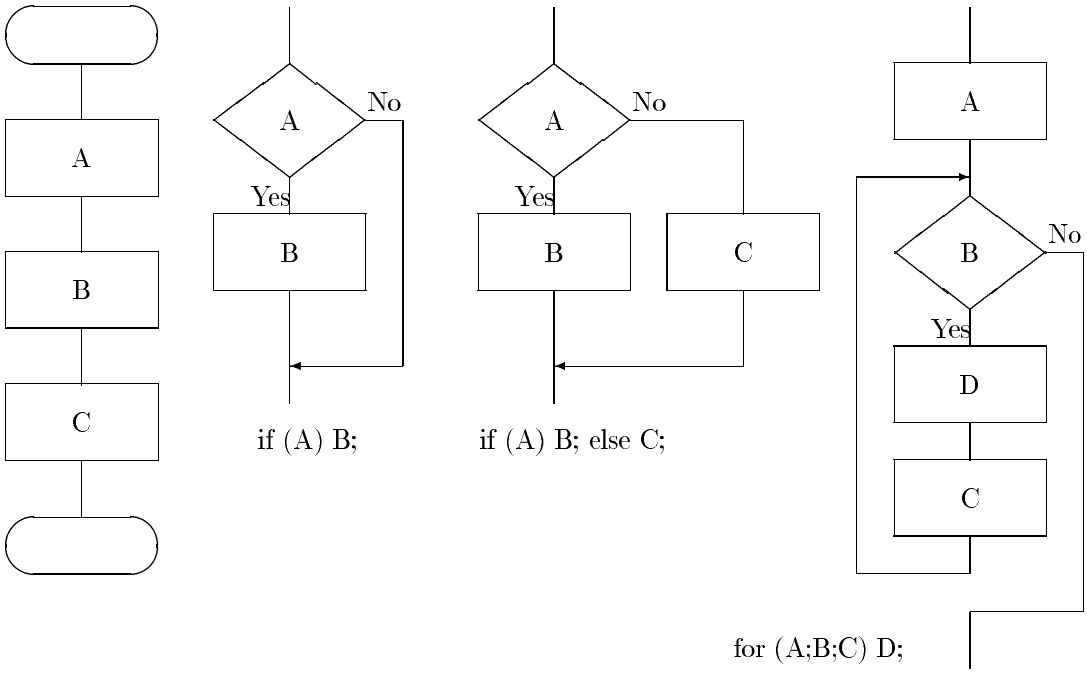
これでファイル名の本体がtest1であるファイルは全て消去されます。

［演習］

1. MS–DOSのコマンド例を実行してみよう。コピー、名前の変更、削除が確実に行われたかどうかdirコマンドで確認すること。
2. 自分のフロッピーにある拡張子がbakであるファイルを全て消去して、その確認をせよ。通常拡張子bakは、エディタで編集する前の古いファイルの内容を保存しているファイルにつく。

5.8 フローチャート (流れ図)

フローチャートはプログラムの構造を図示するのに用いられます。プログラムの最初と終わりを長丸、処理を長方形、判断をひし形で表します。



必ずしもCの1つの文を1つの箱に対応させる必要はありません。通常はプログラム全体の大きな流れを数個の箱で書き、次にその箱の1つ1つを別のフローチャートでより詳しく記述するような事が行われます。

［演習問題］

前節の演習問題のmax3.cのプログラムをフローチャートに直せ。

5.9 ループ (1)

コンピュータの処理能力を生かすにはできるだけ大量のデータを処理してもらうことが重要です。数値の計算にしても僅か数個の数値を計算するのならば、プログラムを書くよりも電卓を利用した方が速いでしょう。しかし数百個のデータを加えて平均を求めるような場合に、プログラムに数百回加算する文を書くのでは大変です。そのような場合には、その規則性に注目して少ない文で同様な働きをするプログラムを作成するのが普通です。

くり返しの回数があらかじめ決まっているような場合には、既に学んだfor文を使用するのが普通ですが、さらにちょっとした手法が必要になります。以下のプログラムは1から与えられた数までを全て加えるものです。

```
#include <stdio.h>

main(){
   int i,n,s;

   printf("N = ");scanf("%d",&n);
   s=0;
   for (i=1;i<=n;i++) s=s+i;
   printf("Sum = %d\n",s);
}
```

```
A>sum△
N = 5△
Sum = 15

A>sum△
N = 10△
Sum = 55
```

［演習問題］

1. nの階乗(n!)を計算するプログラムを作れ。なおn!=1*2*3*...*(n-1)*nである。(kaijo.c)

   ```
   A>kaijo△
   N=5△
   N!=120
   ```

2. 1からnまでの奇数の総和を求めるプログラムを作れ。(kisuwa.c)

   ```
   A>kisuwa△
   N=15△
   Kisuwa=64
   ```

［応用問題］

1. ${}_nC_r$(n個の中からr個を取り出す組み合わせの数: ${}_nC_r = \frac{n!}{(n-r)!r!}$)を計算するプログラムを作れ。ただし繰り返しはforを1個だけ使用すること。(ncr.c)

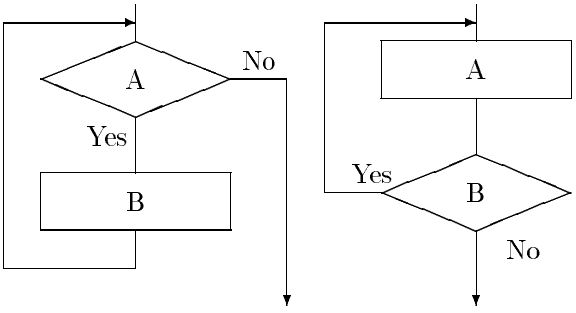
   ```
   A>ncr△
   N=5△
   R=3△
   nCr=10
   ```

2. 1からnまでの奇数の総和を求めるプログラムを作れ。(kisuwaa.c)

```
A>kisuwaa△
N=15△
1+3+5+7+9+11+13+15=64
```

5.10 while文

forと同様に繰り返しを行うもので、先に条件判定するものと、後で行うものがあります。



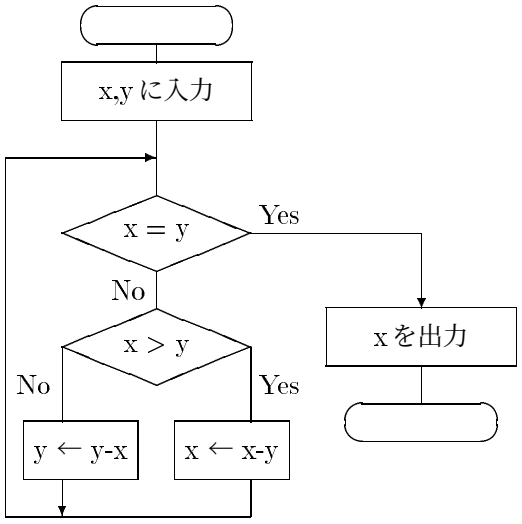
while (A) B;　　do A; while (B);

左側のwhileではまずAの条件を調べます。OKならばBを実行します。そして再びAの条件を調べます。Aの条件がOKの間Bを繰り返し実行します。実はこれはfor(; A;) B; と同じ働きになります。

右側のwhileはまずAを実行します。それからBの条件を調べます。条件がOK ならば再びAを実行します。Bの条件がOKの間Aを繰り返し実行します。先程のwhileと異なるのは、条件に係わらずこちらのwhileは最低１回の実行がなされることです。左側のwhileでは条件が最初から駄目な場合には１回も実行されません。

［演習問題］

1. フローチャートに基づいて最大公約数を求めるプログラムを作れ。(gcd.c)




```
A>gcd△
X=12△
Y=15△
G.C.D.=3
```

2. 任意の自然数に対して、偶数ならば2で割る、奇数ならば3倍して1を足すと言う操作を繰り返すと、必ず1になることが知られている。実際に入力した数に対して同様な操作を行うプログラムを作れ。(test231.c)

```
A>test231△
x = 5△
x = 16
x = 8
x = 4
x = 2
x = 1
```

［応用問題］

1. 最小公倍数を求めるプログラムを作れ。(lcm.c)

```
A>lcm△
X=12△
Y=15△
L.C.M.=60
```

2. 任意の自然数に対して、偶数ならば2で割る、奇数ならば3倍して1を足すと言う操作を繰り返すと、必ず1になることが知られている。実際に入力した数に対して同様な操作を行うプログラムを作れ。演習問題と計算は同じだが出力の形式が異なる事に注意せよ。(test231n.c)

```
A>test231n△
x = 5△
1 : 16
2 : 8
3 : 4
4 : 2
5 : 1
```

5.11 ループ (2)

以下は任意の数のデータの最大値を求めるプログラムです。データの個数が予め判っている場合にはforを使って繰り返すほうが簡単です。ここでは正のデータを入力して行って、最後に数値のゼロを入力してプログラムにデータの終わりを知らせるようにしています。

```
#include <stdio.h>

main(){
   int i,m;

   m=0;
   do {
      printf("Data = ");scanf("%d",&i);
      if (i>m) m=i;
   } while (i>0);
   printf("Max  = %d\n",m);
}
```

```
A>mmax△
Data = 5△
Data = 3△
Data = 10△
Data = 0△
Max  = 10
```

［演習問題］

1. 上記のプログラムをフローチャートに直せ。
2. 平均値を求めるプログラムを作れ。ただしデータは全て正の値でデータの終わりを示すのに0を用いるものとする。(heikin.c)

```
A>heikin△
Data=20△
Data=30△
Data=10△
Data=40△
Data=0△        （データの終わりを示す）
Mean=25        （(20+30+10+40)/4）
```

［応用問題］

1. 任意の数のデータの最小値を求めるプログラムを作れ。ただしデータは全て正の値としデータの終わりを示すのに0を用いるものとする。(mmin1.c)

```
A>mmin1△
Data=20△
Data=30△
Data=10△
Data=40△
Data=0△
Min =10
```

2. 任意の数のデータの最小値を求めるプログラムを作れ。ただしデータは全て正の値としデータの終わりを示すのに0を用いるものとする。なお以下の例のように、最初からデータの終わりの0が入力されても対応するようにする。(mmin2.c)

```
A>mmin2△
Data=20△
Data=30△
Data=10△
Data=40△
Data=0△
Min =-10
```

```
A>mmin2△
Data=0△
No Minimum
```

5.12 コメント

プログラムの説明等をプログラム中に埋め込むことができます。/* と */で囲みます。例は次のbreak文の例を見てください。

5.13 break文

breakを実行すると一番内側のループ(forでもwhileでも)から抜け出る事ができます。これを利用した際には、ループの次に実行が移るのが、正常にループを終了した場合と、途中でbreakで抜けてきた場合の２通りになるので注意が必要です。

```
/*      素数の判定プログラム      */

#include <stdio.h>
```

```
main(){
   int i,s;                    /*  iには割ってみる数、sには素数かどうか調べる数が入る。 */

   printf("?=");scanf("%d",&s);
   for (i=2;i<s;i++)                          /*  2からs-1までの数で割ってみる。        */
      if ((s%i)==0)                           /*  もしsがiで割れたとすると。。。        */
         break;                               /*  forを中断する。                       */
   if (i==s) printf("%dは素数である。\n",s);  /* forを完了して出てきたときはi==sである。*/
   else      printf("%dは素数ではない。%dで割れる。\n",s,i);
}
```

［演習問題］

1000以下の素数を表示するプログラムを作れ。(s1000.c)

```
A>s1000△
   2   3   5   7  11  13  17  ...
  73  79  83  89  97 101 103  ...
 179 181 191 193 197 199 211  ...
 ...
```

［応用問題］

1. 与えられた数を素因数分解するプログラムを作れ。(bunkai1.c)

   ```
   A>bunkai1△
   N=120△
   120=2*2*2*3*5
   ```

2. 与えられた数を素因数分解するプログラムを作れ。(bunkai2.c)

   ```
   A>bunkai2△
   N=120△
   120=2^3*3^1*5^1
   ```

5.14　配列

同じ型の変数を複数個、配列と言う形で扱う事ができます。その場合宣言の際に必要な個数を要求しなければなりません。例えば、int a[10]; とすると、a[0]、a[1]、a[2]、... a[9]と言う変数が計10個使えるようになります。

このとき[ ]の中は変数を含む数式でも構いません。変数の値によって同じものが別の変数を意味するようになります。例えば、scanf("%d",&x); はいつも変数xに入力する意味ですが、scanf("%d",&a[i]); は変数iの値が0ならばa[0]に入力しますし、変数iの値が9ならばa[9]に入力します。だからと言って、宣言した範囲を越えた値を変数iに入れて使用してはいけません。通常[ ]の中の値は宣言した範囲内かどうか調べないので、プログラムや他の変数の内容を破壊する事になります。

次の例は配列を使った入力されたデータを入力した逆順に出すプログラムです。データの入力の部分を見ると1つのscanf()で全てのデータの入力が可能になっています。

```
#include <stdio.h>

main(){
   int i,n,data[100];
```

```
   printf("How many data? = ");scanf("%d",&n); /*  データの個数を入力する */
   for (i=0;i<n;i++) {                         /*  データを配列に読み込む */
      printf("data=");scanf("%d",&data[i]);
   }
   for (i=n-1;i>=0;i--)                        /*  データを逆順に出力する */
      printf(" %d",data[i]);
   printf("\n");
}

A>revers△
How many data? = 4△
data=1△                1はdata[0]に入る
data=2△                2はdata[1]に入る
data=3△                3はdata[2]に入る
data=4△                4はdata[3]に入る
 4 3 2 1
```

i--はi++の反対の働きで、変数iの内容を1減らします。

［演習問題］

上記のプログラムを改良し、データの終わりを0で示すことにし、最初にデータの個数を入れなくても良いようにせよ。(reverse.c)

```
A>reverse△
data=5△
data=8△
data=9△
data=0△
 9 8 5
```

［応用問題］

分散を定義通り計算するプログラムを作れ。データは1つ以上あり、全て正の値とし、データの終わりは0で示すものとする。分散は与えられた各データから平均を引き、自乗したものの総和を個数で割って求められる。(variance.c)

```
A>variance△
data=10△
data=30△
data=20△
data=40△
data=0△
Mean=25  Variance=125
```

5.15　文字型変数

Cのプログラムでは文字と文字列を区別して扱います。文字は俗に半角と呼ばれる大きさの文字1つを示します。プログラム中では、'a'の様に文字を'(Single quote)で囲います。一方文字列は文字が0個以上連続したものと定義されます。プログラム中では、"abc"の様に文字を"(Double quote)で囲います。0個でも文字列ですので、""も文字列です。また文字には空白も含みますので、"I have a pen."も1つの文字列です。

これらの文字データを扱うために文字型の変数があります。宣言はintの代わりにcharを用います。普通の文字型変数1つには1文字しか入りません。文字列を入れるためには文字型の配列を用います。配列には

文字列を構成する文字が1つずつ入りますが、それに加えて最後には必ず'\0'(ヌル文字)が入ります。例えばwordという文字型の配列に"bat"と言う文字列を入れると、word[0] には'b'、word[1]には'a'、word[2]には't'、word[3]には'\0'が入ります。

word	[0]	[1]	[2]	[3]	[4]	...	[29]
内容	'b'	'a'	't'	'\0'	?	...	?

文字型の配列に入った文字列は、1文字ごと扱うことも可能です。上記のような状態で word[0]='h'; を実行するとwordの中の文字列は"hat"になります。ただ一文字しか含まない様な"a"と言う文字列でも'a'と言う文字とは異なるので注意して下さい。

```
#include <stdio.h>

main(){
   int i,l;
   char word[30];                          /*  最大30文字入れる事ができる。              */

   printf("Word = ");scanf("%s",word);     /*  文字列の入力の時は%sにする。&や[ ]は不要 */
   for (l=0;word[l]!='\0';l++) ;           /*  変数lの値を文字列の終わりまで増やす       */
   printf("Word Length=%d\n",l);
   for (i=l-1;i>=0;i--)                    /*  単語を逆順に出力する                      */
      printf("%c",word[i]);                /*  文字を出すときには%cを用いる              */
   printf("\n");
}
```

```
A>testm△
Word = computer△
Word Length=8
retupmoc
```

これまでscanf()やprintf()の中で%dは数値を意味していました。同様に%sは文字列を、%cは文字を意味します。文字列の入力の時だけちょっと違うので注意して下さい。

	整数型	文字型	文字列型
宣言	int n;	char c;	char s[80];
入力	`scanf("%d",&n);`	`scanf("%c",&c);`	`scanf("%s",s);`
出力	`printf("%d",n);`	`printf("%c",c);`	`printf("%s",s);`
代入	n=10;	c='x';	不可

[演習問題]

1. 入力した単語の中に母音(a、e、i、o、u)が幾つ含まれるか計算するプログラムを作れ。(vowels.c)

   ```
   A>vowels△
   Word=computer△
   There are 3 vowels in 'computer'.

   A>vowels△
   word=big△
   There is a vowel in 'big'.
   ```

2. 入力された文字列の先頭3文字と残りの文字列を別々に表示するプログラムを作れ。なお、入力される文字列の長さは3文字以上あるものとする。(cut3.c)

```
A>cut3△
String=personal△
head=per, body=sonal
```

［応用問題］

1. 入力した単語の中でもっとも短いものの長さを表示するプログラムを作れ。(shortest.c)

```
A>shortest△
Word=book△
Word=are△
Word=personal△
Word=today△
Word=Z△          ←  一番最後はZを入れるが、これは考えない。
The shortest word length is 3.
```

5.16　文字列を扱う関数

Cにはいくつかの文字列を扱うための関数が標準で用意されています。今sとt を文字列変数（char s[100],t[100];のように宣言した物）とします。

```
gets(s);      sに１行丸ごと取り込みます。
puts(s);      sの内容を表示して改行します。
strcat(s,t);  sに入っている文字列の後にtに入っている文字列を追加します。
strcmp(s,t);  sに入っている文字列とtに入っている文字列を比較します。結果は数値です。
    strcmp("abc","def") → 負の値    strcmp("abc","abc") → ゼロ
    strcmp("def","abc") → 正の値
strcpy(s,t);  tに入っている文字列をsにコピーします。
strlen(s);    sに入っている文字列の長さを求めます。
```

Cでは文字列を代入する事はできません。つまりs="abc";はできないので、かわりにstrcpy(s,"abc");を使います。gets()とscanf()の違いは、入力されたものに空白が含まれるときに、gets()ならば空白も含めて全て変数に入りますが、scanf()ならば空白の手前までしか変数に入りません。puts(s)は、printf("%s\n",s)と全く同じです。

注意:strcat()、strcmp()、strcpy()、strlen()を使用する際には、プログラムの先頭に #include <string.h> が必要です。

以下は上記の関数を使用したプログラムの例です。

```
#include <stdio.h>
#include <string.h>

main(){
   char word[20],sentence[80];

   printf("Sentence = ");gets(sentence);
   printf("Sentence Length = %d\n",strlen(sentence));
   strcpy(sentence,"");strcpy(word,"");
   do {
      strcat(sentence," ");
      strcat(sentence,word);
      printf("Word = ");gets(word);
   } while (strcmp(word,"end")!=0);
   puts(sentence);
}

A>strtest△
```

```
Sentence = I love you△
Sentence Length = 10
Word = I
Word = love
Word = you
Word = end
 I love you
```

［演習問題］

1. 入力した姓と名前の順番を入れ替えて表示するプログラムを作れ。ただし姓と名前の間には1つだけ空白があるものとする。(exchg.c)

   ```
   A>exchg△
   Full Name=Miki Kunihiro△
   In English=Kunihiro Miki
   ```

2. 入力した文から単語を取り出すプログラムを作れ。（単語間にはスペースが必ず１つだけあり、行末には何もつかないものとする。）(getword.c)

   ```
   A>getword△
   Sentence = I have a book△
   Word = I
   Word = have
   Word = a
   Word = book
   ```

3. 文中に指定した文字列が含まれているときにyesと答えるプログラムを作れ。(search.c)

   ```
   A>search△
   Sentence = I have a book.△
   String = oo△
   yes
   ```

5.17　switch文

Ｘの値がＡだったらこうして、Ｂだったらああして。。。と言った事をしたい時には、ifを何個も書く必用がありましたが、このような場合にはswitch文と言うものが使えます。

```
switch（X）{
   case A : こうして; break;
   case B : ああして; break;
   default : その他する;
}
```

Ｘの部分には任意の式が書けます。ＡやＢの部分には、数値とか文字を書きます。ここには変数を含む式は書けません。「こうして」の部分には複数の文を{ }で囲わなくても書けます。その後のbreakは通常必要ですが、無くともエラーにはならず、「こうして」をやった後で「ああして」もするような動作になります。Ｘの値がＡでもＢでもなかった場合にはdefault以降の「その他する」の部分が実行されます。そのような物が必要でない場合にはdefaultの行は省略可能です。

5.18 関数の定義のやり方 (1)

Cでは自分で好きな関数を定義して利用することができます。と言うか数十行を越えるプログラムは、1画面に収まる程度の長さの関数に分割して作成するのが普通です。

これまでどのプログラムもmain(){...}と言う形をしていました。実はこれはmain()と言う関数を定義していた事になります。ですからtekitou()と言う関数を定義したい時にはmain()同様に行えば良いのです。

```
tekitou(){
   tekitou() の内容
}

main(){
   main() の内容
}
```

main()の内容でtekitou()を利用したいときには、tekitou();と言う行を書けばできます。何度利用してもかまいませんからプログラム中に繰り返し用いられる部分を関数として定義すると、プログラムの長さを短く、見易くすることができます。さらにtekitou()の中で別の関数を利用することもできます。この場合利用される関数の定義を、利用する関数の前に書くことを薦めます。

tekitou()の中で変数の宣言をしてそれを使うことができます。ただしそこでどのような事を行っても他の関数で宣言された変数の値は変化しません。例えば、

```
tekitou(){
   int i;
   i=100;
}

main(){
   int i;
   i=3;
   tekitou();
   printf("i=%d\n",i);
}
```

のような場合、main()で最初に3が変数iに入ります。そしてtekitou()の中では変数iに100を入れてますが、main()にもどるとそれとは関係無く、画面にはi=3と表示されます。このような関数の中でしか通用しない変数の事を局所変数(ローカル変数)と呼びます。逆に大域変数(グローバル変数)と呼ばれるものもあります。関数の定義の前に宣言した変数はそれ以降の関数の中で利用することができますが、どこかの関数でその値を変更するとその変更は他の関数に対しても有効になります。

```
int i;      /*  関数の定義の外にあるので、大域変数  */

tekitou(){
   i=100;  /* 変数 i は既に宣言されているので、すぐ使える  */
}

main(){
   i=3;
   tekitou();
   printf("i=%d\n",i);
}
```

この場合にはi=100と表示されます。各関数の定義の際にint i;が無いことに注意して下さい。もしint i;をそのまま残していると、ローカル変数の宣言の方が優先されます。つまり次のような場合には、i=3と表示されます。よく間違える点なので、注意してください。

```
int i;     /* 関数の定義の外にあるので、大域変数  */

tekitou(){
   int i;  /* tekitou() の中では、変数 i は局所変数  */
   i=100;  /* これは局所変数 i に入れるだけ          */
}

main(){
   i=3;
   tekitou();
   printf("i=%d\n",i);
}
```

関数を利用する側から関数に値を渡すことができます。例えば、

```
tekitou(int height){
   if (height>170) printf("You are tall!\n");
}

main(){
   tekitou(155);
   tekitou(165);
   tekitou(175);
   tekitou(185);
}
```

のような形になります。最初の行の( )の中のintが変数の型を示し、その次が変数名です。tekitou()の中ではここで指定した変数がローカル変数と同様に使えます。複数の値を渡したいときには、型と変数名の後にカンマを入れて型と変数名を繰り返します。上のプログラムでは最初の呼び出しで155がheightに入り、次の呼び出しで165がheightに入ります。最終的には4つの値がheightに渡されるために、画面には2回You are tall!が表示される事になります。

［演習問題］

数回にわけて三目並べのプログラムを作成する。今回は関数に分割しないプログラムを元にして、これを関数を使った形に直せ。(ox1.c)

```
#include <stdio.h>

main(){
   int i,j,k,ban[10];

   printf("\n  ***  Tick-Tack-Toe Game  ***\n");
   printf("        ver.1 Man - Man\n\n");
   for (i=1;i<=9;i++) ban[i]=0; /* ban[i] がゼロならば i マスは空         */

   j=1;                         /* j が１ならば o の番、－１ならば x の番 */
   for (i=0;i<9;i++) {
      do {
         printf("==> ");scanf("%d",&k);
      } while ((k<1) || (k>9) || (ban[k]!=0));
      ban[k]=j;
      j=-j;
      printf("\n      |   |\n");
      switch(ban[1]) {
         case -1 : printf("    X |"); break;
         case  0 : printf("    1 |"); break;
         case  1 : printf("    O |");
```

```
      }
      switch(ban[2]) {
         case -1 : printf(" X |"); break;
         case  0 : printf(" 2 |"); break;
         case  1 : printf(" O |");
      }
      switch(ban[3]) {
         case -1 : printf(" X\n"); break;
         case  0 : printf(" 3\n"); break;
         case  1 : printf(" O\n");
      }
      printf("      |   |\n");
      printf("   ---+---+---\n");
      printf("      |   |\n");
      switch(ban[4]) {
         case -1 : printf("    X |"); break;
         case  0 : printf("    4 |"); break;
         case  1 : printf("    O |");
      }
/*   途中省略   */
      printf("      |   |\n\n");
   }
}
```

関数を使わないと大変長くて何をしているか判らない物になるが、ここでintialize()、input()、display()と言う関数の定義を適当に作れば、main() の部分は次のようになる。

```
main(){
   int i,j;

   initialize();      /*  メッセージの表示とban[ ]の初期化  */
   j=1;

   for (i=0;i<9;i++) {
      input(j);       /*  空いているところに入力  */
      j=-j;
      display();      /*  ban[ ]の表示   */
   }
}
```

関数に分割したために、main() の内容を見るだけである程度のプログラムの動作が理解できるようになっている。なお、単純にここで示した関数を別に定義するだけならば、全体の行数はむしろ増加する。ここでdisplay() の内容をさらに別に定義した関数を複数回呼ぶようなものにすると、全体の行数を減らすことができるので、是非試みること。

```
A>ox1△

  ***  Tick-Tack-Toe Game  ***
       ver.1 Man - Man

==> 1△
      |   |
    0 | 2 | 3
      |   |
   ---+---+---
      |   |
    4 | 5 | 6
      |   |
   ---+---+---
```

```
     |   |
   7 | 8 | 9
     |   |

==> 5△
     |   |
   0 | 2 | 3
     |   |
   ---+---+---
     |   |
   4 | X | 6
     |   |
   ---+---+---
     |   |
   7 | 8 | 9
     |   |
```

以下省略。

5.19 関数の定義のやり方 (2)

関数本来の働きと言えば、与えられた値に対して何か値を返すことです。C の関数は数値だけでなく、文字なども返すことができます。そのやり方を以下に説明します。まず関数定義の先頭に返す値の型を書きます。何も返さない場合にはvoidと書きます。(前節のように何も型を書かない場合には、整数が返されるものと解釈されます。)

関数から値を返す場合にはreturn(...);を使います。この( )の中に入っている値が関数から返す値になります。なおこのreturnが実行されると、関数の実行は終わってこれを呼び出した方へ実行が戻ります。

```
int abs(int i){              /*   abs() は整数の値を返す関数     */
   if (i>=0) return (i);     /*   i が正ならばそのまま返す       */
   else      return (-i);    /*   i が負ならば符号を変えて返す   */
}

main(){
   printf("abs(3)=%d\n",abs(3));
   printf("abs(-3)=%d\n",abs(-3));
}
```

関数は他の関数を呼び出せるだけでなく、自分自身を呼び出すことも可能です。例えばnの階乗（$n! = n*(n-1)*(n-2)*...*3*2*1$）と呼ばれる値は以前 for を利用して計算しましたが、以下のように関数を使っても計算可能です。

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

main(){
   printf("6!=%d\n",fact(6));
}
```

このように関数が自分自身を呼び出すことを再帰呼び出しと言います。

5.20 プログラムの挿入

次の演習問題は前の演習問題で作成した関数をそのまま利用します。その場合2つのやり方があります。includeを使用する方法と別々にコンパイルしたものを結合する方法です。残念ながら後者の説明は今年度

のテキストからは外しました。前回の課題の関数の部分がox1.cに入っているとすると、今日の課題のプログラムの中に、

```
#include "ox1.c"
```

と言う指示を入れておくと、入れた所にox1.cの内容が挿入されます。これまで通常のプログラムの先頭には`#include <stdio.h>`と言う行がありましたが、基本的には全く同じ動作になります。違いは利用者が独自に作ったファイルを挿入する場合にはファイル名を" "で囲み、Cのシステムが持っているファイルを挿入する場合には< >で囲む所だけです。ちなみにここで使っているstdio.hファイルは177行もあります。

［演習問題］

1. 前回の入ったox1.cから、先頭の`#include <stdio.h>`、大域変数の宣言、main()の定義の部分を取り除いて保存せよ。

2. 念のためにox2.cと言うファイルに以下の内容を入力し、前回同様に動作することを確認せよ。

   ```
   #include <stdio.h>
   大域変数の宣言
   #include "ox1.c"
   前回のmain()の定義
   ```

3. 次のような改良を行うこと。

   (a) initialize()の後で盤面を表示させるように直す。

   (b) owari()と言う関数を定義する。これはOかXがどこかに3つ並んでいないかどうか調べて、もしOが3つ並んでいれば、「あなたの勝ち」と表示して1を返す。もしXが3つ並んでいれば「あなたの負け」と表示して1を返す。どちらも3つ並んでいない場合には、0を返すような関数である。

   (c) main()に手を加えて、毎回盤面を表示してからowari()を呼び、勝負がついた場合はそこでプログラムが終了するようにする。

   ox2.cには主として、owari()の定義部分とmain()の定義が入ることになる。

5.21　式の値

Cにおいては、関数だけでなくプログラムを構成するほとんどの要素が値を持ちます。例えばj=1は1と言う値を持ちます。それをさらに利用することも可能でj=k=1とすると、変数kに1を代入した結果の値1を変数jにも入れることになります。

forの中によく使われるi++に似た物として++iと言うものがあります。どちらも変数iの内容を1増やす働きがありますが、前者の値は1増やす前の変数iの値であり、後者の値は1増やした後の変数iの値になります。

owari()==1の結果は0又は0以外の数になります。条件が成立した場合に0以外になります。ifはこの値によって判断するので実は if (owari()==1) ... はif (owari()) ... と同じ事になります。

逆にこのあたりが災いしてif (ban[1]=1) ... なんて間違えると、ban[1]=1はban[1]の値にかかわらず1になりますので、... の部分が必ず実行されることになります。さらにban[1]の値まで変化するので大きな被害が出ることがありますが、Cでは文法に従った正しい記述とみなされます。

5.22 乱数

コンピュータのプログラムは同じデータを与えると同じ結果が得られるのが普通です。しかしゲーム等では、コンピュータ側がいつも同じ手順で仕掛けてくるのでは面白くありません。大抵のプログラミング言語では、呼び出すたびに毎回異なる値を返す関数を用意しています。

Cではrand()と言う関数がそれにあたります。rand()は呼び出す度に0から32767迄の適当な値を返します。通常はもう少し狭い範囲の数が必要となります。例えばさいころの目の代わりをさせるには、1から6までで十分です。以下はさいころを20回振るプログラムです。

```
#include <stdio.h>
#include <stdlib.h>       /*  rand() を利用する際には必要  */

main(){
   int  i;

   for (i=0;i<20;i++)
      printf("%d ",(rand()%6)+1);  /* a % b でaをbで割った余りが求まる  */
   printf("\n");
}
```

このようにrand()の値を6で割って余りを求めると0から5までの数になるので、1を加えると1から6までの値になります。本当にでたらめな数列の事を乱数と呼びますが、このrand()の返す値は内部でそれらしくなるように計算した値なので疑似乱数と呼ばれます。実際上記のプログラムを実行すると、それらしき1から6までの数が20個出ますが、もう一回プログラムを実行するとまた同じ物が出てきます。

```
A>ran△
3 5 4 2 2 6 1 4 1 1 2 6 1 1 6 6 4 4 4 1
A>ran△
3 5 4 2 2 6 1 4 1 1 2 6 1 1 6 6 4 4 4 1
```

これでは困ることが多いので、通常は、時刻を得る関数とrand()の計算の元になる数を設定する関数srand()を利用して、プログラムの起動時刻によって異なる系列の乱数が得られるようにします。

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>     /*  clock() を利用する際には必要。  */

main(){
   int  i;

   srand(clock());  /*  現在の時刻の値をsrand() に与える。  */
   for (i=0;i<20;i++) printf("%d ",(rand()%6)+1);
   printf("\n");
}

A>ran△
5 2 2 6 3 6 2 2 2 3 2 2 6 4 4 4 5 1 4 6
A>ran△
4 5 6 4 2 5 3 3 3 5 6 6 2 1 4 3 3 2 6 6
A>ran△
4 3 5 6 5 3 6 1 1 2 6 3 2 4 4 3 2 2 5 4
```

［演習問題］

1. 前回作成したox2.cをox3.cにコピーせよ。それからox2.cの関数の定義の部分だけ残して保存し、逆にox3.cから関数の定義部分を消去せよ。念のためにここでプログラムのの動作確認をする。

2. Xの順番の時に、適当に空いた所にXを入れる関数umeru()を作れ。これはrand()を利用して1から9の値を求めて、対応する位置が空いていればそこにXを入れるものである。また以前作成したinitialize()の中にsrand(clock());の行を追加しておく。またmain()を少し直して、Oの番ならばinput(j);を呼び、Xの番ならばumeru()を呼ぶようする。

3. OやXが2つ並んでいる時には、空いた所にXを入れるtomeru()を作れ。tomeru()はXを入れた場合には0を返し、何もしなかった場合には1を返すようにすること。main()もまた少し直して、tomeru()をやってみて、何もなければ先程のumeru()を実行するようにする。

4. umeru()やtomeru()の定義部分はox3.cに入れてox3.cの内容を提出せよ。

［応用問題］

1. 必ず人が先行ではなく、最初に「先行ですか？(y/n)」とメッセージを出して、人がyと答えた場合のみこれまで通りとし、そうでない時にはコンピュータ側が先行になるようにせよ。

2. 関数umeru()を改良し、中心が空いていればかならず中心を埋め、そうでなく四隅のどれかが空いていれば、その中からランダムに選んで埋め、そうでもなければ残りの中からランダムに選んで埋めるようにせよ。

3. rand()を使わずに人に負けないプログラムにせよ。

5.23　キーボード 入力関数

scanf()を使用して文字、文字列、数値をキーボードから入力する事ができました。しかし、毎回入力の最後にはリターンキーを押さなければなりません。また一度入力待ちになると、先に進むことができなくなります。特にゲーム等ではこれでは困ることがあります。

`#include <conio.h>`にはkbhit()と言う関数とgetch()と言う関数が定義されています。kbhit()は数値を返す関数で、現在キーボードで何かのキーが押されていると0以外を、何も押されていないと0を返します。またgetch()はキーボードから１文字だけ文字を入力してその文字を返します。（もしまだ何もキーが押されていない場合には押される迄待ちます。）

［演習問題］

1. プログラムを起動すると”Hit any key!”と表示され、何かキーを押すとサイコロの目がランダムに表示されるものをrand()を使用せずに作れ。(dice1.c)

   (a) まず指定した目を出す関数disp()を作る。（disp(1)とすると[・]が表示される）

   (b) 変数の値が1→2→...→6→1...と変るようにし、変える度にキー入力を調べて何か押されていたらdisp()を呼ぶようにする。

2. サイコロの目によるスロットゲームを作れ。プログラムをスタートすると画面が消去され中央付近にくるくる変化するサイコロの目が3つ表示される。何かキーが押されると左端のサイコロの目が停止し、もう一回押すと真ん中のサイコロの目が停止し、さらにもう一回押すと右端のサイコロの目も停止してプログラムも終了するものとする。(dice2.c)

   (a) 演習で作成したプログラムの先頭に、`#include "screen.c"` を挿入する。その結果次のような関数をプログラム中で利用することが可能になる。

```
locate(x,y) : xは1から80桁、yは1から24行で指定した位置にカーソルが移動する。
cls()       : 画面を消去する
cursol(ON)  : カーソルを表示する
cursol(OFF) : カーソルを見えなくする
```

(b) 関数disp()を改良する。disp(1,3,10)とすると、1の目を3行目の10桁目から表示するようにする。

(c) 前回の課題では最後に1回だけ呼んでいたdisp()を今回は何度も呼ぶ事になる。停止したサイコロは書き直さないようにすること。

［応用問題］

演習問題のdice2.cのmain()の定義を直してgame()とし、5ゲームして、その総得点を返す関数とする。得点は3つのサイコロの目がそろったら10点、2 つのサイコロの目がそろったら3点とする。(dice3.c)

ヒント：実際はmain()の内容はonegame()と言う別の関数にして、game()の中からこれを5回呼ぶ形にした方が良いでしょう。

5.24 ファイルの読み書き(1)

ゲームプログラムなどを除き、有用なプログラムはファイルの読み書きができることが最低必要です。例えば文書が保存できないワープロソフトでは困ります。通常ファイルからプログラムにデータを取り出すことをファイルの読みだし、プログラムからファイルにデータを入れる事をファイルへの書き込みと言います。ファイルにデータを入れることにより、コンピュータの電源が切れてもデータを残すこともできますし、フロッピー等を使用してデータの交換も可能になります。ここではファイルの読み書きの基本を説明します。

ファイルからデータを読み込むときには、まずfopen()を実行します。この時にファイルを読むのか書くのかの指定もします。通常1つのファイルに同時に読み書きを行うことはしません。正常にファイルを開くことができるとこの関数はファイルディスクリプタへのポインターを返します。(ファイルディスクリプターはファイルに関する様々なパラメタを記憶している所の事です。ポインターについての説明は今年度のテキストからは省略しましたが、この場合記憶している場所を示すものの事です。ファイルを操作するにはここに記憶しているパラメタが必ず必要となります。) その後はこの返された値を元にファイルを操作します。ファイルからデータを読む場合にはfscanf()等を使います。これはscanf()とほぼ同じ動作ですが、キーボード から読み込む代わりにファイルから読み込んでくれます。全てのファイルを読み込んだ後は、fclose()を必ず実行します。次は数値を1つdataと言う名前のファイルから読み込むプログラムの例です。

```
#include <stdio.h>

main(){
   FILE *fp;      /*   ファイルディスクリプターへのポインターを入れる変数の宣言      */
   int i;

   fp=fopen("data","r");            /*   ファイル名(data)と読み出し(r)            */
   fscanf(fp,"%d",&i);              /*   最初にfpが入るだけで後はscanf()と同じ   */
   printf("Read Data is %d.\n",i);
   fclose(fp);                      /*   使い終わったら必ずこれを呼ぶこと          */
}
```

逆にファイルへデータを書き込むときには、fopen()でファイル名と書き込みを指定して、fprintf()等でデータを書き込み、最後にfclose()を行います。以下はキーボード から入力した数値をdataと言うファイルに書き込むプログラムです。

```
#include <stdio.h>

main(){
   FILE *fp;
   int i;

   fp=fopen("data","w");     /*   ファイル名(data)と書き込み(w)          */
   printf("Number = ");scanf("%d",&i);
   fprintf(fp,"%d\n",i);     /*   最初にfpが入るだけで後はprintf()と同じ  */
   fclose(fp);
}
```

［演習問題］

応用問題で作成した得点の出るサイコロプログラムを改良して、ファイル(dice.max)に最高得点者の得点と名前がファイルに残るようにせよ。(dice4.c)

1. dice3.cの内容をdice4.cにコピーする。
2. redでdice.maxと言うファイルを作成する。その内容は、1行目に0、2行目に適当な名前とする。
3. ファイルdice.maxから過去の得点と得点者の名前を大域変数に読み込む関数read_score()と、得点と得点者の名前をファイルdice.max に書き込む関数write_score()を定義する。
4. main()の内容の最初にread_score()の呼び出しと最後にwrite_score() の呼び出しを追加する。
5. main()にゲームをして得られた得点は表示し、これまでの最高点と比較も行う部分を追加する。過去の最高点よりも高い得点であることが判明すれば、得点者の名前の入力も行う。

5.25　ファイルの読み書き(2)

ファイルへの出力に関しては、プログラムが出力したいだけデータを出力するで済みますが、入力の場合で特にデータの量が既知でない場合は、まだファイルにデータが残っているかどうか調べながら読むことになります。例えばfscanf()は正しく読めなかった場合-1を返すので、そうでなければファイルからデータが読めたことになります。

```
#include <stdio.h>

main(){
  FILE *fp;
  char line[80];

  fp=fopen("test.c","r");
  while (fscanf(fp,"%s",line)!=-1) puts(line);
  fclose(fp);
}
```

ファイルにデータをどんどん追加して行きたい場合には、fopen()の際に”w”のかわりに”a”を指定すると、以前出力した結果の後にこれから出力する結果を追加することができます。逆に言えば、”w”を指定した場合には以前ファイルに出力したデータは全て消去されます。

［演習問題］

サイコロプログラムを、最高点を記録した歴代の名前とその得点をファイルに保存するように、改良せよ。ファイルの内容は1行ごとに点と名前が交互に並んだ形とする。read_score()を修正して、ファイルにある得点と名前を読みながら画面に表示する。そして最後の点(歴代の中の最高点)のみ記憶する。write_score()も修正して名前と点をファイルに追加するようにし、write_score()は最高点が出た後で呼び出すようにmain()を直す。

5.26　いかにプログラムを作っていくのか？

与えられた課題のプログラムを考えて行くにはどうすれば良いのか？と言うのは難しい問題です。次のようなこの章末の演習問題はどうすれば良いでしょうか？

> 英語のテキストを１行ずつ表示して、それと同じ物をキー入力してもらい、その際にどのくらい正しくキーを押せたかどうか（つまり全文字数に対する誤って押したキーの数の割合）を調べるプログラムを作れ。ただし表示するテキストはtype.txtと言う名前のファイルに入っているものとするが、ファイルの内容は適当に英語の講義のテキスト等を参考に各自入力すること。(ttest.c)

まず、タイプの練習をする部分とtype.txtから例文を読み込む部分の2つには分けられそうです。Cではこのような場合、それぞれを別の関数として定義して使うのが普通ですので、それぞれtype_test()とread_text()と言う名前にしましょう。また2つの関数の間で共通な変数として少なくとも例文を入れておく変数が必要です。以上をプログラムの形にすると次の様なものになります。

```
#include <stdio.h>

char text[5][80]; /*  とりあえず５行分    */

main(){
   read_text();   /*  例文を読み込む      */
   type_test();   /*  例文でテストをする */
}
```

次にこの関数の定義を考えていきます。read_text()は何も値を返しません。また何もmain()から受け取りません。するとこの関数の定義の先頭は、

```
void read_text(){
```

となり、同様にしてもう1つの関数は、

```
void type_test(){
```

になります。ファイルから何かデータを読む例はなかったか？これは5.24に例がありましたね。例ではデータを一つだけ読み込んでいますが、ここではtext[]に5行分読み込む事になります。(ファイルに5行分データがなかったらどうする？)

さて問題はtype_test()の方になります。まだかなり複雑そうな動作なので探しても似たような例は見つからないでしょう。そうすると自分で考えないといけません。例文が5行ありますから、5回繰り返す事になります。するとforを使うのでしょう。それから全文字数と誤って押したキーの数を数えるための変数が必要です。その初期化と最後の結果の計算も考えると、

```
void type_test(){
   int i, miss, all;         /*  miss: 間違えた数、all: 全文字数 */

   miss=0;all=0;             /*  やる前はミスも文字数もゼロ。     */
   for (i=0;i<5;i++){
      .....                  /*  ここを考えなくてはならない。     */
   }
   printf("Your Score is %d %%\n",miss*100/all);  /*  miss/all*100 ではうまく行かない  */
}
```

通常文字数と言えば空白は除きますが、タイプの練習の場合は空白も正しく入力しなければならないので、これも数えます。そうすると、5.16 にある文字列関数がそのまま使えます。そしていよいよ課題の核心の部分となります。1 行表示してそれと同じ物をタイプしてもらう所です。これも明らかに表示する部分とタイプしてもらう部分に分けられます。でも表示は printf() 一発ですからわざわざ関数としてわける事もないでしょう。ではタイプしてもらう所を type_line() とでもしましょう。今何行目を表示してタイプしてもらうつもりかを知らせてやる必要がありますし、逆にタイプした際の誤りの数を教えてもらわないといけません。するとこの関数の定義の先頭は、

```
int type_in(int i){
```

と言った形になるでしょう。さて通常の文字列の入力ではリターンキーを押すまで自由に変更ができます。ですから通常の入力方式ではタイプのテストになりませんので、5.23 でやった getch() を使って 1 文字ずつ取り込む事にします。タイプされた文字が例文と同じ文字ならば当たりですから次の文字と比較するようにします。外れならば正しい字を打ってくれるまで待たなければなりません。何回間違えるかなんてやってみなければわかりませんから、for でないループになります。キー入力があってから判定をしますから、do { } while ( );が使えます。

だいたいこれでできてしまいますが、実際これでやってみると getch() は押したキーの文字を画面に出してくれないので、やっている人はどこまで自分が入力したのか判らなくなります。そこで正しく入力できた文字は、画面に出してやるようにします。

［演習問題］

英語のテキストを 1 行ずつ表示して、それと同じ物をキー入力してもらい、その際にどのくらい正しくキーを押せたかどうか（つまり全文字数に対する誤って押したキーの数の割合）を調べるプログラムを作れ。ただし表示するテキストは type.txt と言う名前のファイルに入っているものとするが、ファイルの内容は適当に英語の講義のテキスト等を参考に各自入力すること。(ttest.c)

注意事項：read_text() は簡単です。１行丸ごとファイルから読み込むのには fgets() 関数を用います。これは、

```
fgets(文字列変数, 読み込む文字数, ファイルポインター)
```

のような形で使います。ファイルから読もうとした行が 2 つ目に指定した文字数よりも大きい場合は、指定した文字数だけ読み込まれます。(ですから通常文字列変数に入る最大の文字数を指定します。) また gets() と異なり読み込んだ行の最後にある'\n' も文字列変数に入ります。ファイルから全く読み込めなかった場合（ファイルの終わりに達した場合）に NULL を返すのは gets() と同じです。

type_test() で考えるのは miss や all を数える部分だけです。

type_line() の部分は、１文字ずつ getch() で取ってきて比較します。文字の比較ですから文字列の比較と混同しないように。不用意に getch() を呼ぶとその度にキーを押すことになりますので、getch() が複数出てきた人は注意して下さい。

A　演習問題の解答

A.1　hello.c

次が一番普通の答えでしょう。

```
#include <stdio.h>

main(){
   printf("*********\n");
   printf("* HELLO *\n");
   printf("*********\n");
}
```

実はこれでも同じ結果が得られます。

```
#include <stdio.h>

main(){
  printf("*********\n* HELLO *\n*********\n");
}
```

A.2　tanuki.c

根性のある方は次の答えでも構いませんが、考えるだけでも疲れませんか？

```
#include <stdio.h>

main(){
   printf("  0 0      0 0      0 0      0 0      0 0    \n");
   printf(" (o.o)    (o.o)    (o.o)    (o.o)    (o.o)   \n");
   printf("=( x )=  =( x )=  =( x )=  =( x )=  =( x )=  \n");
   printf("  U U      U U      U U      U U      U U    \n");
   printf("  0 0      0 0      0 0      0 0      0 0    \n");
   printf(" (o.o)    (o.o)    (o.o)    (o.o)    (o.o)   \n");
   printf("=( x )=  =( x )=  =( x )=  =( x )=  =( x )=  \n");
   printf("  U U      U U      U U      U U      U U    \n");
   printf("  0 0      0 0      0 0      0 0      0 0    \n");
   printf(" (o.o)    (o.o)    (o.o)    (o.o)    (o.o)   \n");
   printf("=( x )=  =( x )=  =( x )=  =( x )=  =( x )=  \n");
   printf("  U U      U U      U U      U U      U U    \n");
   printf("  0 0      0 0      0 0      0 0      0 0    \n");
   printf(" (o.o)    (o.o)    (o.o)    (o.o)    (o.o)   \n");
   printf("=( x )=  =( x )=  =( x )=  =( x )=  =( x )=  \n");
   printf("  U U      U U      U U      U U      U U    \n");
   printf("  0 0      0 0      0 0      0 0      0 0    \n");
   printf(" (o.o)    (o.o)    (o.o)    (o.o)    (o.o)   \n");
   printf("=( x )=  =( x )=  =( x )=  =( x )=  =( x )=  \n");
   printf("  U U      U U      U U      U U      U U    \n");
}
```

通常は次のようにします。

```
#include <stdio.h>

main(){
     int i;

     for (i=0;i<5,i++) {
```

```
        printf("  O O      O O      O O      O O      O O    \n");
        printf(" (o.o)    (o.o)    (o.o)    (o.o)    (o.o)   \n");
        printf("=( x )=  =( x )=  =( x )=  =( x )=  =( x )=  \n");
        printf("  U U      U U      U U      U U      U U    \n");
    }
}
```

さらに横方向にも繰り返しがあるので、forを重ねるとこの場合はかえって長いプログラムになります。

```
#include <stdio.h>

main(){
  int i,j;

  for (i=0;i<5;i++) {
      for (j=0;j<5;j++) printf("  O O    ");
      printf("\n");
      for (j=0;j<5;j++) printf(" (o.o)   ");
      printf("\n");
      for (j=0;j<5;j++) printf("=( X )=  ");
      printf("\n");
      for (j=0;j<5;j++) printf("  U U    ");
      printf("\n");
  }
}
```

A.3 kuku.c

printf() の中で"%4d"の4がないと、表示する数値によって桁数が変わるので四角い表になりません。

```
#include <stdio.h>

main(){
   int i,j;

   printf("\n  ***  The Multiplication Table  ***\n\n");
   for (i=1;i<10;i++) {
      for (j=1;j<10;j++) printf("%4d",i*j);
      printf("\n");
   }
}
```

A.4 2kou.c

本当は高学歴も入れたかったのですが、学歴を数値で入れるにはどうしたら良いと思いますか？昔ジャンケンのプログラムで0、2、5と入力するものがありました。

```
#include <stdio.h>

main(){
   int h,i;

   printf("Height=");scanf("%d",&h);
   printf("Income=");scanf("%d",&i);
   if (h>=170)
      if (i>=1000) printf("I love you.\n");
            else   printf("Bye bye, you must work hard.\n");
     else
      if (i>=1000) printf("Bye bye, you are too small.\n");
```

```
            else   printf("Bye bye.\n");
}
```

A.5 max3.c

3つの数の中の最大値ですから、まず2つを比較してその大きいほうと残りの数値を比較するやり方をとると次のようなプログラムになります。

```
#include <stdio.h>

main(){
   int a,b,c;

   printf("A=");scanf("%d",&a);
   printf("B=");scanf("%d",&b);
   printf("C=");scanf("%d",&c);
   if (a>b)
      if (c>a) printf("Max=%d\n",c);
         else  printf("Max=%d\n",a);
    else
      if (c>b) printf("Max=%d\n",c);
         else  printf("Max=%d\n",b);
}
```

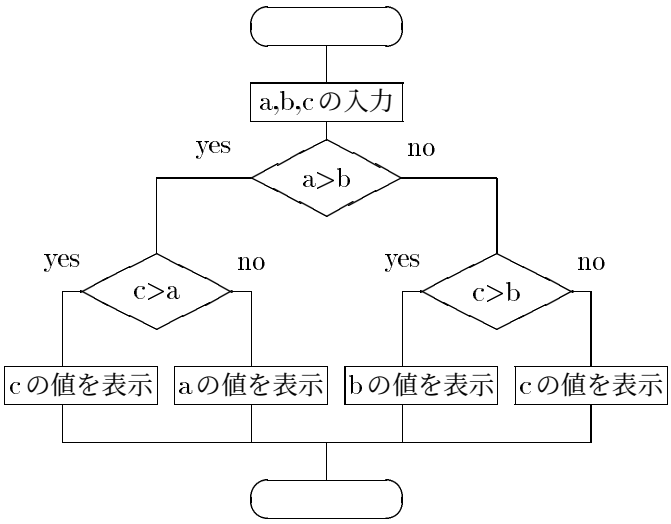
一方、変数aに入っている値を最大値として、他の値と比較してもし負けるようだったらその値を新しい最大値として変数aに入れる方法もあります。この方が短いプログラムになります。

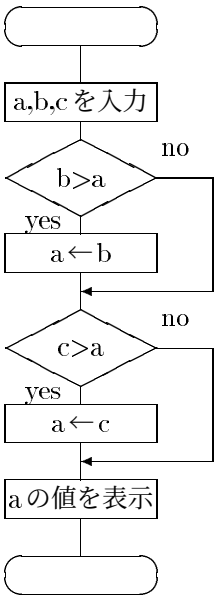
```
#include <stdio.h>

main(){
   int a,b,c;

   printf("A=");scanf("%d",&a);
   printf("B=");scanf("%d",&b);
   printf("C=");scanf("%d",&c);
   if (b>a) a=b;
   if (c>a) a=c;
   printf("Max=%d\n",a);
}
```

3つの数の最大値を求める場合、入力される値としては3つとも異なる、2つが等しい、3つとも等しいの大きく分けて3通りがあります。これらの全てに対して正しく結果が得られるものでないといけません。

A.6 max3.c のフローチャート

最初のプログラムのフローチャートが左側のものです。





A.7 kaijo.c

```
#include <stdio.h>

main(){
   int i,n,s;

   printf("N=");scanf("%d",&n);
   s=1;
   for (i=1;i<=n;i++) s=s*i;
   printf("N!=%d\n",s);
}
```

A.8 kisuwa.c

```
#include <stdio.h>

main(){
   int i,n,s;

   printf("N=");scanf("%d",&n);
   s=0;
   for (i=1;i<n;i=i+2) s=s+i;
   printf("Kisuwa=%d\n",s);
}
```

A.9 gcd.c

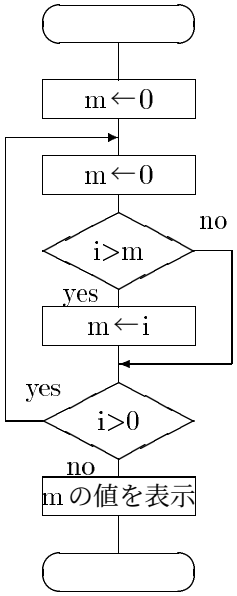
```
#include <stdio.h>

main(){
   int x,y;

   printf("X=");scanf("%d",&x);
   printf("Y=");scanf("%d",&y);
   while (x!=y)
      if (x>y) x=x-y;
```

```
      else     y=y-x;
   printf("G.C.D.=%d\n",x);
}
```

A.10 test231.c

```
#include <stdio.h>

main(){
   int x;

   printf("x = ");scanf("%d",&x);
   while (x>1) {
      if ((x%2)==0) x=x/2;
      else          x=x*3+1;
      printf("x=%d\n",x);
   }
}
```

A.11 ループ (2) のフローチャート



A.12 heikin.c

データの個数を数える際に、最後の 1 個は単なるデータの終わりの印なので除かないといけません。次の例のように割り算をする際に 1 を引く方法の他に予め n に-1 を入れておく方法があります。

```
#include <stdio.h>
main(){
   int d,n,s;

   n=0;s=0;
   do {
      printf("Data=");scanf("%d",&d);
      s=s+d;
      n++;
```

```
   } while (d>0);
   printf("Mean=%d\n",s/(n-1));
}
```

A.13 s1000.c

ここでは素数かどうか調べたい数より 1 少ない数まで割って調べていますが、そんなに大きな数までで割って調べる必要はありません。普通は$\sqrt{n}$までで十分で、もしn=10000ならば100倍ぐらい速くなります。

```
#include <stdio.h>

main(){
   int i,s;

   for (s=2;s<=1000;s++) {
      for (i=2;i<s;i++)
         if ((s%i)==0) break;
      if (s==i) printf("%4d",s);
   }
   printf("\n");
}
```

A.14 reverse.c

data[i]に入力した時にはdata[i]を調べて終わりかどうか判断しなければなりません。この入力と調べるの間にi++が入っていると見た目は同じですが、正しく動きません。

```
#include <stdio.h>

main(){
   int i,data[100];

   i=-1;
   do {
      i++;
      printf("data=");scanf("%d",&data[i]);
   } while (data[i]>0);
   for (i=i-1;i>=0;i--)
      printf(" %d",data[i]);
   printf("\n");
}
```

A.15 vowels.c

```
#include <stdio.h>

main(){
   int i,n;
   char word[30];

   printf("Word=");scanf("%s",word);
   n=0;
   for (i=0;word[i]!='\0';i++)
      if ((word[i]=='a') || (word[i]=='e') || (word[i]=='i') ||
          (word[i]=='o') || (word[i]=='u')) n++;
   if (n==1) printf("There is a vowel in '%s'.\n",word);
   else      printf("There are %d vowels in '%s'.\n",n,word);
}
```

A.16 exchg.c

```
#include <stdio.h>

main(){
   char fname[40];
   int  i;

   printf("Full Name=");gets(fname);
   for (i=0;fname[i]!=' ';i++) ;
   fname[i]='\0';
   printf("In English=%s %s\n",&fname[i+1],fname);
}
```

文字列の中から文字を探す関数をやっていないので、これがもっとも簡潔な答えでしょうが、ちょっと解りにくいですね。

A.17 getword.c

頭から1文字ずつ見て行って、普通の文字はそのまま出して、空白が来たら改行して、Word=を出しておくだけです。他には、前の空白の位置を憶えて置いて、一気に書く方法もあります。むしろこの方が普通かもしれません。

```
#include <stdio.h>
#include <string.h>

main(){
   int i;
   char sentence[80];

   printf("Sentence=");gets(sentence);
   printf("Word = ");
   for (i=0;sentence[i]!='\0';i++)
      if (sentence[i]==' ') printf("\nWord = ");
      else                  printf("%c",sentence[i]);
   printf("\n");
}
```

A.18 search.c

```
#include <stdio.h>
#include <string.h>

main(){
   int i,j;
   char sentence[80],string[20];

   printf("Sentence = ");gets(sentence);
   printf("String = ");gets(string);
   for (i=0;sentence[i]!='\0';i++){
      for (j=0;string[j]!='\0';j++)
         if (sentence[i+j]!=string[j]) break;
      if (string[j]=='\0'){
         printf("yes\n");
         break;
      }
   }
}
```

A.19 ox1.c

1つの関数の中でのみ使う変数(同じ名前でも全く関係ない変数も同様)はローカル変数とし、他の関数でも使用する変数のみ大域変数にするのが普通です。よってこの場合ban[ ]だけが大域変数となります。この例ではsuuji() とyoko() と言う関数を使用してプログラムの長さを短くしています。

```
#include <stdio.h>

int ban[10];

initialize(){
   int i;

   printf("\n  ***  Tick-Tack-Toe Game  ***\n");
   printf("        ver.1 Man - Man\n\n");
   for (i=1;i<=9;i++) ban[i]=0;
}

input(int j){
   int k;

   do {
      printf("==> ");scanf("%d",&k);
   } while ((k<1) || (k>9) || (ban[k]!=0));
   ban[k]=j;
}

suuji(int j){
   switch(ban[j]) {
      case -1 : printf(" X "); break;
      case  0 : printf(" %d ",j); break;
      case  1 : printf(" O ");
   }
}

yoko(int i){
   printf("      |   |\n");
   printf("   ");suuji(i);printf("|");suuji(i+1);printf("|");suuji(i+2);printf("\n");
   printf("      |   |\n");
}

display(){
   yoko(1);
   printf("   ---+---+---\n");
   yoko(4);
   printf("   ---+---+---\n");
   yoko(7);
}
```

main() の定義は演習問題文にあったものと同じ。

A.20 ox2.c

```
#include <stdio.h>

int ban[10];

#include "ox1.c"
```

```
int a(int i, int j, int k) {  /* 関数aは３つの引き数i, j, kを取り整数値を返す */
   return (ban[i]+ban[j]+ban[k]);
}

int test(int f){
   if (a(1,2,3)==f || a(4,5,6)==f || a(7,8,9)==f || a(1,4,7)==f ||
       a(2,5,8)==f || a(3,6,9)==f || a(1,5,9)==f || a(3,5,7)==f)
      return (1);
    else
      return (0);
}

int owari(){
   if (test(3)==1) {
      printf("あなたの勝ち\n");
      return (1);
   }
   if (test(-3)==1) {
      printf("あなたの負け\n");
      return (1);
   }
   return (0);
}

main(){
   int i,j;

   initialize();              /*  メッセージの表示とban[ ]の初期化  */
   display();                 /*  ban[ ]の表示   */

   j=1;
   for (i=0;i<9;i++) {
      input(j);               /*  空いているところに入力  */
      j=-j;
      display();              /*  ban[ ]の表示   */
      if (owari()==1) break;  /*  ゲーム終了ならば for から抜ける  */
   }
}
```

A.21 ox3.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int ban[10];

#include "ox1.c"

void umeru(){
   int i;
   while (ban[i=(rand()%9)+1]!=0) ;
   ban[i]=-1;
}

int t(int i, int j, int k){
   switch (ban[i]+ban[j]+ban[k]) {
      case   2 : ;
      case  -2 : if (ban[i]==0) ban[i]=-1;
```

```
                else if (ban[j]==0) ban[j]=-1;
                else ban[k]=-1;
                return (1);
      default  : return (0);
    }
}

int tomeru(){
   if (t(1,2,3) || t(4,5,6) || t(7,8,9) || t(1,4,7) ||
       t(2,5,8) || t(3,6,9) || t(1,5,9) || t(3,5,7))
      return(0);
    else
      return(1);
}

main(){
   int i,j;

   initialize();              /*  メッセージの表示とban[ ]の初期化  */
   display();
   j=1;
   for (i=0;i<9;i++) {
      if (j==1) input(j);
      else      if (tomeru()) umeru();
      j=-j;
      display();              /*  ban[ ]の表示   */
      if (owari()) break;     /*  ゲーム終了ならば for から抜ける  */
   }
}
```

A.22 dice1.c

```
#include <stdio.h>
#include <conio.h>

void disp(int i){
   switch (i) {
      case 1 : printf("      \n  ●  \n      \n");break;
      case 2 : printf("    ●\n      \n●    \n");break;
      case 3 : printf("    ●\n  ●  \n●    \n");break;
      case 4 : printf("●  ●\n      \n●  ●\n");break;
      case 5 : printf("●  ●\n  ●  \n●  ●\n");break;
      case 6 : printf("●  ●\n●  ●\n●  ●\n");
   }
}

main(){
   int i;

   printf("Hit any Key!\n\n");
   for (i=1;kbhit()==0;i++)    /*  キーが押されていない間繰り返す  */
      if (i>6) i=1;            /*  iの値が6を越えたら1に戻す       */
   getch();                    /*  押された内容を読み捨てる        */
   disp(i);
}
```

A.23 dice2.c

```
#include "screen.c"
#include <conio.h>
```

```
void disp(int i, int y, int x){
   locate(x,y);
   switch (i) {
      case 1:         printf("      ");break;
      case 2: case 3: printf("    ●");break;
      default:        printf("●   ●");
   }
   locate(x,y+1);
   switch (i) {
      case 1: case 3: case 5:
                      printf("  ●  ");break;
      case 6:         printf("●   ●");break;
      default:        printf("      ");
   }
   locate(x,y+2);
   switch (i) {
      case 1:         printf("      ");break;
      case 2: case 3: printf("●     ");break;
      default:        printf("●   ●");
   }
}

main(){
   int i;

   cls();cursol(OFF);
   locate(20,10);printf("Hit any Key!");
   for (i=1;kbhit()==0;i++){   /* 3つのさいころが同時に動いている状態  */
      if (i>6) i=1;
      disp(i,15,10);disp(i,15,20);disp(i,15,30);
   }
   getch();
   for (;kbhit()==0;i++) {     /* 2つのさいころが同時に動いている状態  */
      if (i>6) i=1;
      disp(i,15,20);disp(i,15,30);
   }
   getch();
   for (;kbhit()==0;i++) {     /* 1つだけさいころが動いている状態  */
      if (i>6) i=1;
      disp(i,15,30);
   }
   getch();
   cursol(ON);
}
```

さいころがいくつ動いているかをmと言う変数で表すとすると、2重ループ1つでもできます。以下はmain()の定義だけで他の部分は上記のプログラムと同じです。

```
main(){
   int i,j,k,m;

   cls();cursol(OFF);
   locate(20,10);printf("Hit any Key!\n\n");
   i=j=k=1;
   for (m=0;m<3;) {
      if (kbhit()) {  /*  キーを押すとmが増える。→動くさいころの数が減る。 */
         m++;
         getch();
      }
```

```
      switch (m) {
         case 0 : if (++i>6) i=1;disp(i,15,10);  /* breakが無いことに注意 */
         case 1 : if (++j>6) j=1;disp(j,15,20);
         case 2 : if (++k>6) k=1;disp(k,15,30);
      }
   }
   cursol(ON);
}
```

A.24 dice4.c

```
#include "screen.c"
#include <stdio.h>
#include <conio.h>

int top;            /*  topとnameはread_score()でも使うので大域変数にする  */
char name[16];

void disp(int i, int y, int x){
   この関数の定義はdice2.cのものと同じなので省略。
}

int onegame(){                          /*  数値を返すので int が付く  */
   int i,j,k,m;

   cls();cursol(OFF);
   locate(20,10);printf("Hit any Key!");
   i=j=k=1;
   for (m=0;m<3;) {
     if (kbhit()) {
        m++;
        getch();
     }
     switch (m) {
        case 0 : if (++i>6) i=1;disp(i,15,10);
        case 1 : if (++j>6) j=1;disp(j,15,20);
        case 2 : if (++k>6) k=1;disp(k,15,30);
     }
   }
   cursol(ON);
   if ((i==j) && (j==k)) return (10);             /*  ここから３行が追加分  */
   if ((i==j) || (j==k) || (i==k)) return (3);
   return (0);
}

int game(){
   int i,s;

   s=0;
   for (i=0;i<5;i++) s=s+onegame();
   return (s);
}

void read_score(){  /*  何も返さないのでvoid  */
   FILE *fp;

   fp=fopen("dice.max","r");
   fscanf(fp,"%d",&top);
   fscanf(fp,"%s",name);
   fclose(fp);
```

```
}

void write_score(){  /*  何も返さないのでvoid  */
   FILE *fp;

   fp=fopen("dice.max","w");
   fprintf(fp,"%d\n%s\n",top,name);
   fclose(fp);
}

main(){
   int p;

   read_score();
   printf("\n\n総得点 = %d\n",p=game());    /*  得点をpに入れておく。  */
   if (p>top) {
      top=p;
      printf("Your Name = ");scanf("%s",name);
   }
   write_score();
}
```

A.25 dice5.c

dice4.cとの変更点のみ示します。

```
void read_score(){
   FILE *fp;

   fp=fopen("dice.max","r");
   while (fscanf(fp,"%s",name)!=-1) {
      fscanf(fp,"%d",&top);   /*  名前があれば得点もあるので無条件に読んでも問題はない  */
      printf("%d by %s.\n",top,name);
   }
   fclose(fp);
   printf("\nHit Any Key!! ");/*  これと次の行が無いとすぐゲームが始まってしまう  */
   getch();
}

void write_score(){
   FILE *fp;

   fp=fopen("dice.max","a");    /*  ここだけ w が a に変っている  */
   fprintf(fp,"%s\n%d\n",name,top);
   fclose(fp);
}

main(){    /*  write_score()の位置だけ変更あり  */
   int p;

   read_score();
   do {
      printf("\n\n総得点 = %d\n",p=game());
      if (p>top) {
         top=p;
         printf("Your Name = ");scanf("%s",name);
         write_score();  /*  記録が更新された時のみファイルに書き込む  */
      }
      printf("Do you want play again? (y/n) : ");
   } while (getch()=='y');
```

```
   printf("\n\nHigh Score is %d by %s.\n",top,name);
}
```

A.26 ttest.c

```
#include <stdio.h>
#include <conio.h>
#include <string.h>

char text[5][80];

void read_text(){
   FILE *fp;
   int  i;

   fp=fopen("type.txt","r");
   for (i=0;i<5;i++){
      fgets(text[i],80,fp);  /*  80は読み取る文字列の最大の長さ。 */
      text[i][strlen(text[i])-1]='\0'; /* fgets()が付ける文字列の最後の\nを除く。*/
   }
   fclose(fp);
}

int type_line(int i){
   int j, m;

   m=0;   /*  mで誤って押した数を数える。 */
   for (j=0;text[i][j]!='\0';j++){
      while (text[i][j]!=getch()) m++; /* getch()で読んだ文字が違ったらmを増やす。*/
      printf("%c",text[i][j]);
   }
   printf("\n\n");
   return (m);
}

void type_test(){
   int i, miss, all;

   miss=all=0;
   for (i=0;i<5;i++) {
      all=all+strlen(text[i]);
      printf("%s\n",text[i]);
      miss=miss+type_line(i);
   }
   printf("Your Score is %d %%\n",miss*100/all);
}

main(){
   read_text();
   type_test();
}
```