

Part1. C++言語入門

C++言語の基本文法を学ぶ。

エディタによる編集から、コンパイル、実行をスムーズにできるように。

A0J を使った自動採点をできるようにする。

わからなくなったら自力で調べられるようにする。

ちなみに構造体やクラスはやらないので自分でやって下さい。

1. Hello world
2. 変数と演算子
3. 入出力
4. if 文
5. for 文
6. 二重ループ
7. 配列
8. 文字と文字列
9. 命令

1-1. Hello world

C++言語はprocessingみたいにボタン一発で動く言語と違って、プログラムを動かすまでがちょっと大変である。この章では、Hello Worldを表示するだけの簡単なプログラムを作りながら、プログラムを書いて、それを実行するまでの手順を学ぶ。

それと共に、A0Jというネット上の採点システムの使い方も学ぶ。

(1) 環境整備

- ・作業ディレクトリの用意

デスクトップにある「K2013summer」というフォルダの中に自分の名前のついたフォルダを用意しよう（今回は英語の名前をつけてほしい）。

この講習で作ったプログラムなどは、すべてここに保存しましょう。

- ・エディタ

今回は「サクラエディタ」を使う。起動しておこう。

基本操作のショートカットキー

新規作成	Ctrl+N
上書き保存	Ctrl+S （.cppで保存すると色をつけてくれる）
コピー	選択してCtrl+C
貼り付け	Ctrl+V
元に戻す	Ctrl+Z
検索	Ctrl+F

もちろん上のメニューから選んでも良い。

・ コマンドライン

さて、今回の講習では C++ というプログラミング言語を使う。

C++ は、

エディタでプログラムを書く

コンピュータが実行できる形に変換する（コンパイル）

実行する

という流れでプログラミングすることになる。この「コンパイル」と「実行」を行うために、「Cygwin」あるいは「コマンドプロンプト」という コマンドライン環境 を使うことにする。

コンピュータのデータは、ご存知の通りフォルダ（ディレクトリともいう）の中にフォルダがあって、その中にファイルがあって、という ツリー構造 で記録されている。

ふつうの Windows 環境なら、マウスを使ってフォルダの中に入ったり、ひとつうへのフォルダに戻ったり、フォルダの中のファイルを開いたりするが、コマンドライン環境では次のようなコマンドを使うことでその操作を行う。Cygwin の人とコマンドプロンプト (CMD) の人でちょっと違う。

今どのフォルダにいるか	Cygwin: pwd, CMD: cd
フォルダの中身の表示	Cygwin: ls, CMD: dir
今のフォルダ内にある hoge というフォルダに移動	cd hoge
一つ上のフォルダに移動	cd ..
移動先を絶対パスで指定	cd C:\hoge\hoge\hoge

1. まず C ドライブ（一番上位の階層にあるフォルダ）に移動しよう。
cd C:
2. 次にその中身を表示してみよう K2013summer というフォルダも表示されるだろうか？
3. K2013summer に移動しよう
4. 中身を表示して、自分の名前のフォルダが有るか確認しよう
5. そのフォルダの中に移動しよう
6. 今どのフォルダにいるか確認しよう

ちなみに「cd」はチェンジディレクトリの略だったりする。

pwd は print working directory, ls は list segments である。

(2) 実行！

1. サクラエディタで次のプログラムを書こう。

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      cout << "Hello World" << endl;
6      return 0;
7  }
8
```

書いたら「1-1-hello.cpp」のような名前で自分のフォルダの中に保存するべし。

2. コマンドラインに移り、自分のフォルダの中身を表示し、今書いたプログラムがちゃんとあるかどうか確認せよ。
3. 次のコマンドを実行し、「コンパイル」する
g++ 1-1-hello.cpp
4. 「a.exe」というファイルが新しくできるはずである。これがあるかどうか確認せよ。先ほどの g++ コマンドにより、1-1-hello.cpp という C++ プログラムが、a.exe というコンピュータが実行できる形に変換（コンパイル）されたということである。
5. Cygwin なら ./a.exe, CMD なら a.exe とコマンドを打って、このプログラムを実行せよ。Hello World と一行表示されるはずである。

・基本コマンド

hoge.cpp をコンパイル	g++ hoge.cpp
a.exe を実行	Cygin: ./a.exe, CMD: a.exe

・コンパイル時に次のようなこともできる（コンパイラオプションという）

a.exe でなく hoge.exe という名前にコンパイル	g++ hoge.cpp -o hoge.exe
コンパイル時にレベル2の高速化を行なう	g++ hoge.cpp -O2

～解説～

```

1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     cout << "Hello World" << endl;
6     return 0;
7 }
8

```

入出力のためのライブラリ `iostream` を読み込む
`cout` などが使えるようになる。

ライブラリの命令などの頭につく `std::` を省けるようにする。
これがないと `std::cout` などと書く必要がある。

`main` という名前の命令をつくる。
`main` 命令はプログラムが実行された時に、
最初に呼び出される命令となる。

コマンドラインに (`cout`)
"Hello World" という文字列を出力し、
次に改行 (`endl`) を出力する。

戻り値0を返す (`return`)。
`main` 命令は、プログラムの正常終了時に
0を返すという決まりがある。

このように各行にいろいろ意味があるわけだが、このプログラムの本質は5行目
だけである。Hello Worldという文字列を出力し、改行して終わりということだ。

・コンパイルエラー

プログラムの文法が間違っていると、コンパイル時にエラーが出て、コンパイル
に失敗する。

試しに5行目の最後のセミコロンを消して、コンパイルしてみよう。
僕の環境では次のようなエラーが出た。環境によってメッセージは少し違う。

6: error: expected `;' before 'return'

6行目の `return` の前にセミコロンがいるぜ！って意味である。

次に、5行目の `endl` を `enl` に変えてコンパイルしてみよう。

5: error: 'enl' was not declared in this scope

5行目の '`enl`' というのは存在しない（作られていない）と言われた。

このようにどこが間違っているかの手がかりを教えてくれるので、それを参考に
しつつプログラムを直して（デバッグして）いくことになる。

ちなみに"Hello World"の最後のセミコロンを打ち忘れると、次のようにたくさん
エラーが出てくる。カッコなどの閉じ忘れはプログラムの構造を分からなくして
しまうので、エラーもなんだかごちゃごちゃになることが多い。

```

5:10: warning: missing terminating " character
5: error: missing terminating " character
In function 'int main()':
6: error: expected primary-expression before 'return'
6: error: expected `;' before 'return'

```

(3) オンライン採点システム

この講習では A0J（会津大学オンラインジャッジ）という、ネット上で利用可能なプログラムの自動採点システムを使ってプログラミングの勉強を進めていこうと思う。

1. アカウントを作る

まずはアカウントを作ろう。

<http://judge.u-aizu.ac.jp/onlinejudge/>

にアクセス（または google で A0J で検索）して、右上の登録／設定ボタンを押す（英語になっている人は Japanese を先に押そう）。

ユーザ ID には好きな自分のハンドルネームを使おう。名前以下は書かなくても良かったはず。それで送信ボタンを押せば登録が完了して、A0J が使えるようになる！

2. Hello Worldの問題を開く

メニューから「コース」を選び、そのなかの「Lesson」を開く。

コース名「Introduction to Programming I」を開く。

一番上の「Getting Started」を開き、

A の「Hello World」を開く。

Hello Worldの問題が開かれたらどうか？

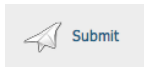
この Part 1 では、主に Introduction to Programming I 中の問題を解いていくことにする。

3. 自動採点してみる

A0J は素晴らしいことにプログラムを送信すると、自動で採点してくれる。さっそくこの Hello World を解いてみよう。

問題文を読むと、Hello World と一行出力すればいいことがわかる。つまりさっき作ったプログラムをそのまま使えばいいわけだ。

右上の



ボタンを押して、

プログラムをコピーしてソースコード欄に貼り付ける。言語は C++ を選ぶ。



そして下の「提出」ボタンを押そう。

すると「ステータス」画面が開いて、しばらくすると

< Problem A ✓ Accepted 100 % 100 C++ 0.00 sec 1116 KB 99 B 736419 >

このような表示が現れる。Accepted は問題に正解したことを表す！

4. バグった時の反応を見る

それでは誤ったプログラムを送ってしまうとどうなるか試してみよう。

まず、5 行目のセミコロンをわざと抜かして、コンパイルに失敗するプログラムを送ってみよう。

Compile Error

と出てくるはずだ。コンパイルに失敗したことを表す。

次に、セミコロンを戻して、Hello World ではなく Hello Word と出力するようにして提出してみよう。

Wrong Answer

と出てくるだろう。これは出力が間違っていることを表す。

次に、5 行目を

```
cout << "Hello World" ;
```

としてみよう (<< endl) を消去。こうすると、最後に改行が出力されなくなる。

Presentation Error

という反応が帰ってくるだろう。これは出力の形式（空白や改行の入り方、出力の文字数など）が間違っていることを表している。

他にもいくつかエラーはあるが、それはもう少し先の章でお目にかかるだろう。

1-2. 変数と演算子

C++プログラムを書いて、実行して、採点システムに送れるようになったところで、C++言語の中身の勉強に進んでいこう。

(1) データ型

	型の名前	例 1	例 2	例 3
整数	int	1	1234	-314
小数	double	0.1	5.67	-159.0
文字	char	'a'	'&'	'¥'
文字列	string	"aiueo"	"1"	" "

C++言語では、主に上の表の4種類のデータ型を扱うことができる（実はまだまだあったりするのだが）。

整数は1, 2, 3みたいなので -2147483648~2147483647 までOKである。

小数は小数点がついてる数で、有効数字15ケタくらい。整数なんだけど小数型のデータとして使いたいときは例3のように.0をつける。

文字はシングルクォーテーション(' ')で囲う。しかし改行やシングルクォーテーションそれ自体など、コンパイラを混乱させそうな一部の特殊文字は例3のように文字の手前に¥をつけてやる。

文字列は文字の列であり、ダブルクォーテーション(" ")で囲う。1文字や0文字の文字列もありうる。後の章で詳しくやる。

(2) 演算子

次の記号を使って演算を行うことができる。

計算と記号

足し算：+
引き算：-
かけ算：*
わり算：/
あまり：%
かっこ：()

これらは整数、小数に対して主に用いられる。

整数同士の計算は整数となる。

整数同士の割り算は結果が切り捨てられる。

どうしても整数同士の割り算の結果を小数にしたければ整数の前に(double)と書く。

(3) 変数

データを覚えさせておくのには「変数」を使う。使い方は他の言語と大体一緒と思って良い(というより、他の言語がC言語の影響を強く受けているのだ)。

変数

<宣言>

型名 変数名の順で書くことで、その型のデータを記録する変数を作る
カンマで区切ってまとめて宣言も可能

```
int hensu;
double x, y, z;
```

<代入>

変数にデータを代入するには = を使う
宣言と同時の代入も可能

```
hensu = 1;
char c = 'a';
```

変数は、次に別の値が代入されるまで、
前に代入したデータを覚えている

※変数名

英大小文字、数字、アンダーバー_が使える
ただし、数字で始まるのはダメ
すでに存在する何かとかぶってもダメ

整数または小数型の変数 x に 10 を足したいとき、

```
x = x + 10;
```

と書くと、x に、x+10 が代入されるので結果的に x に 10 が足される。

これは、次のように短く書くこともできる。同じように -=, *=, /=, %= もある。

```
x += 10;
```

また、1 だけ足したい、引きたいときには、もっと省略して

```
x++; x--;
```

と書くこともできる。for 文の中とかでよく使う。

以下はプログラム例、まあこんな感じでできますよ、と。

```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     cout << 12/5 << endl;           // 整数割る整数は整数になる
6     cout << 12/5.0 << endl;         // どちらか一方が小数なら小数となる
7
8     int x = 10;
9     cout << x << " " << x*x << " " << x+10 << endl;
10    // x, x*x, x+10をスペースで区切って出力。詳しくは次で。
11
12    x = 7373; // 新たな7373を代入した段階で、前に覚えていた10は忘れる
13    int y = 1507;
14    cout << x*y << endl;
15
16    return 0;
17 }
```

左のプログラムの結果

```
2
2.4
10 100 20
11111111
```

1-3. 入出力

入力というのはキーボードとかマウスとかからデータを受け取ること、出力というのは画面とかにデータを表示することである。今回の講習では、コマンドライン上で、キーボードからデータを受け取って、コマンドライン上に出力を表示するようなプログラムを書いていく。

(1) 簡潔な入出力方法

入力 `cin`

変数nにデータを読み込む

```
cin >> n;
```

変数a, b, cにデータを読み込む

```
cin >> a >> b >> c;
```

出力 `cout`

改行を出力

```
cout << endl;
```

変数nを出力して改行

```
cout << n << endl;
```

変数a, b, cをスペース区切りで出力

```
cout << a << " " << b << " " << c << endl;
```

とりあえず次のプログラムを書いて実行してみよう

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int a, b;
6
7      cin >> a;
8      cin >> b;
9      // 上の2行は cin >> a >> b; と略記してもいい
10
11     cout << a+b;
12     cout << " ";
13     cout << a*b;
14     cout << endl;
15     // cout << a+b << " " << a*b << endl; と同じ
16
17     return 0;
18 }
```

実行したら、

```
3 4
```

と書いてエンターキーを押そう。

```
7 12
```

と出てきたら OK だ。

このプログラムは空白で区切られた整数 a と b を読み込んで、その和と積をスペース区切りで一行に出力するものである。

cinの特徴

入力: 3 14 1.5
This is a pen.

```
cin >> a >> b >> c >> d >> e >> f >> g;
```

(
 cin >> a;
 cin >> b >> c;
 のように複数行に分けてもいい。)

空白（スペース、タブ、改行など）で区切られたかたまりを、順番に変数に代入していく。

この例では対応する色の変数に代入される。
 a, bはint型
 cはdouble型
 d, e, f, gはstring型
 (fは'a'だけなのでchar型でもいい)
 の必要がある

coutの特徴

```
cout << 10;  
cout << 20;  
cout << 30;
```

102030 と出力される

```
cout << 10 << " ";  
cout << 20 << endl;  
cout << 30 << endl;
```

" "(スペース一文字)
 10 20 — endl(改行)
 30 と出力される

改行や空白、区切りなどはちゃんと指示しないとイケない。

A0Jの次の問題を解け

- ・ 1-Getting StartedのBのX Cubic
- ・ 1-Getting StartedのCのRectangle

どちらも先ほどのプログラムを少し改良すれば解ける。Acceptedを得られるまで頑張ろう。

このcin, coutを使った入出力はわりと簡潔で使いやすいのだが、このままでは動作が遅い、空白を読み飛ばす、doubleの出力時に適当に切り捨てられる、という欠点がある。

これらの点の克服は後ほど問題になってくるので、そのときにまた扱う。

1-4. if 文

毎度おなじみの条件分岐、if 文です。

if 文

```
if(A){  
    (Aが正しければ実行される処理)  
}
```

```
if(A){  
    (Aが正しければ実行される処理)  
}else{  
    (Aが間違っていれば実行される処理)  
}
```

```
if(A){  
    (Aが正しければ実行される処理)  
}else if(B){  
    (Aが間違っていて、Bが正しければ実行される処理)  
}else{  
    (AもBも間違っていれば実行される処理)  
}
```

条件式の記号

X == Y :等しい
X != Y :等しくない
X < Y :XがYより小さい
X <= Y :XがY以下
X > Y :XがYより大きい
X >= Y :XがY以上

(X==Y) && (Y==Z) :XがYと等しく、かつ、YがZと等しい
(X==Y) || (Y==Z) :XがYと等しい、または、YがZと等しい

- ・ 次のプログラムを書いて実行してみよう。

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int n;
6      cin >> n;
7
8      if(n>=100){
9          cout << "Big" << endl;
10     }else{
11         cout << "Small" << endl;
12     }
13
14     return 0;
15 }
```

if ブロックの中（中括弧 { } の内側部分）は、タブキー 1 つ分字下げをすると良い。
（これは守らなくてもコンパイルできるが、もっと長いプログラムになると人間にとって読みにくくなってしまう。）

ひとつの整数を入力して、100 以上なら Big, そうでなければ Small を出力する。

AQJ の次の問題を解け

- ・ 2-Branch on Condition-A “Small, Large, or Equal”
出力は変数の中身とかではなく文字列であることに注意
- ・ 2-Branch on Condition-B “Range”
左ページの「かつ」の条件式を使う
- ・ 2-Branch on Condition-C “Sorting Three Numbers”
できるだけ少ない数の if 文で処理できるよう工夫してみよう
if 文の中に if 文を入れる手もある。

1-5. for 文

毎度おなじみの繰り返し、for 文です。

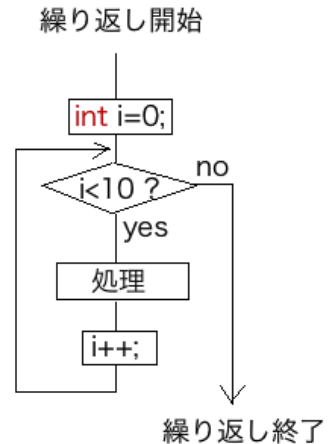
for文の基本

```
for(A; B; C){
    (繰り返し処理)
}
```

A: 変数用意	例 int i=0	最初にAが実行され、
B: 条件式	例 i<10	Bの条件が正しいかぎり繰り返し処理が実行される。
C: 変化分	例 i++	一回の繰り返しが終わるたびに、Cが実行される。

```
for(int i=0; i<10; i++){
    (処理)
}
```

と書いた時の変数 i のことを「ループ変数」と言ったりする。この i は最初 0 で、繰り返しの度に 1 加算 ($i++$) されていき、二番目の条件式 $i < 10$ を満たさなくなったとき (i が 10 になったとき) に繰り返しを終了する。



・ 次のプログラムは、一つの整数 n を入力で受け取り、0 から $n-1$ までの n 個の整数を一行ずつ表示するプログラムである。

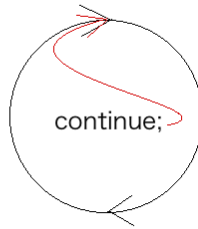
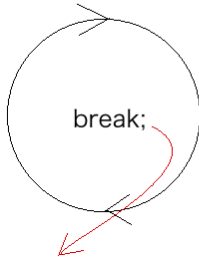
```

1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int n;
6      cin >> n;
7
8      for(int i=0; i<n; i++){
9          // ループ変数iは0, 1, ... n-1と変化する
10         int a = i*i;
11         // 変数aにはループ変数の二乗を入れる
12         cout << i << " " << a << endl;
13         // iとaをスペース区切りで出力し改行
14     }
15
16     return 0;
17 }
```

- ・ break や continue 文を使うと、ループを途中で抜けたりできる。

break: ループを抜ける

continue: 次の周回に飛ぶ



- ・ 無限ループ

次の for 文は絶対に i が 0 未満にはならないため、無限に繰り返し続ける。

```
8   for(int i=0; i>=0; i++){
9
10  }
```

このままだとただの終了しないプログラムとなってダメなのだが、前述の break などと組み合わせると便利ことがある。

なお、次のように for 文の中にセミコロンだけを書いても無限ループとなる。

```
8   for(;;){
9
10  }
```

- ・ 次のプログラムは無限ループと break を組み合わせたものである。

0 以上、入力した整数 n 以下のすべての平方数を表示する。


```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int n;
6      cin >> n;
7
8      for(int i=0; i>=0; i++){
9          // 無限ループにみえるが...
10         if(i*i>n){
11             break;
12             // i*iがnより大きくなったらループを抜けるのでOK
13         }
14         cout << i*i << endl;
15     }
16
17     return 0;
18 }
```

A0Jの問題を解こう

- ・ 3 -Repetitive Processing- A “Print Many Hello World”
これは”Hello World”と1000行出力するだけなので、簡単に解けるだろう。
そこで、解けたら試してみしてほしいことがある。

まず、このプログラムの for 文を無限ループするようにして、コマンドラインで実行しよう。Hello Worldが出続けて止められなくなると思うので、Ctrl と C を同時押ししてほしい。このように、無限ループなどでプログラムが止まらなくなったら Ctrl+Cで強制停止 できる。

次に、この無限ループするプログラムを A0J に送信してみよう。送信してから 30 秒くらい待つと、

 : Time Limit Exceeded

と表示されるはずだ。これは時間切れという意味で、タイムリミットエクシーデッド、略して TLE などという。もちろん不正解となる。情報オリンピックなどのプログラミングコンテストではそれぞれの問題に制限時間が設定されていて、その秒数以内にプログラムが終了しないとイケないのである。

- ・ 3 -Repetitive Processing- B “Print Test Cases”

若干面倒な問題だが、今後の問題を解いていく上で、この問題のテクニックは必須となる。

まず、無限ループを書き、その内側で毎回 cin から変数 x に入力を行う。その入力 x が 0 と等しいとき、break 文でループを脱出する。

Case i: x

は、cout の使い方を思い出しつつ、ループ変数 i と、入力された x を出力させれば良い。

```
1 #include<iostream>
2 using namespace std;
3
4 int main(){
5     int x;
6     for( ) {
7         cin >> x;
8         if( ) {
9             break;
10        }
11        cout << << endl;
12    }
13    return 0;
14 }
```


・ 3 -Repetitive Processing- C “Swapping Two Numbers”

スワップ(swap)というのは、プログラミング用語では二つの変数の中身を交換することを指す。この問題は入出力が二つずつになっただけで、前の問題が解けていれば特に難しいことはないだろう。

さて、今後問題が難しくなってくると、大量の入力データをとるものが出てくる。そのときそれをキーボードで打ち込むのはとても面倒なので、リダイレクトという方法を使う。

まずサクラエディタで新しいファイルを作り、本問の入力データ（右のやつ）をコピーして貼り付け、input.txt という名前で保存せよ。

```
3
5
11
7
8
19
0
```

次に、この問題の解答プログラムをコンパイルし、出来た a.exe を次のようにして実行せよ。

```
./a.exe < input.txt (Cygwin)
```

```
a.exe < input.txt (CMD)
```

ファイルの中身をキーボードで打たなくても、ファイルから読み込んで入力としてくれたのが分かるだろう。このテクニックは今後重宝するのでぜひ覚えておいてほしい。

なお、

```
./a.exe > output.txt < input.txt
```

のようにすると、出力結果をファイル output.txt に書き出すこともできる。

・ 4 -Computation- C “Simple Calculator”

ちょっと飛んで、4-C も解いてしまおう。

この問題では文字を読み込む必要があるが、char 型の変数をひとつ用意して、今までと同じように cin で読みこむだけで出来る。文字の比較は、文字の変数 c として、それが ? 記号と等しいか、だったら

```
if(c == '?')
```

のように if 文を書き始めればいい。

4-A と 4-B を飛ばしたのは、闇の入出力技術 printf と scanf が必要だからである。これらはもうちょっとあとで扱う。

1-6. 二重ループ

for 文の中に for 文を入れることで、より複雑な処理を行うことが出来る。
次のプログラムを書こう。これは九九の表を表示するプログラムである。

```

1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      for(int i=1; i<10; i++){
6          for(int j=1; j<10; j++){
7              cout << i*j;
8              if(j!=9) cout << " ";
9          }
10         cout << endl;
11     }
12
13     return 0;
14 }
```

このプログラムについて、注目してほしい点があるいくつかある。

・ 字下げについて

二重目の for 文では、さらに字下げをする。これによって、その for 文がどこからどこまでなのかわかりやすくなる。

・ if 文の書き方

8 行目の if 文のように、if 文の 中身が 1 行だけなら、中括弧にくくらずに続けて書くことも可能。for 文でも同様に、中身が 1 行なら中括弧を略記できる。

・ 変数の寿命

ループ変数はそのループが終わるまでが 寿命 である。ループの外側ではもうその変数は残っていない。ループの内側で作られた変数も、そのループが終わると消滅する。

なお、新たに作る変数はすでにあるものと同じ名前にはしてはいけないのだが、すでに寿命が付き変数なら問題ないので、ループ変数 `i` のループの外側で、またループ変数 `i` のループを作ることが出来る。

・ 出力方法

このプログラムの出力法は、

```

1 2 3 4
2 4 6 8
```

のようなスペースで区切られた表を出力するときの常套手段である。変数を出力したあと、その行の最後の変数でなければスペースを出力するのがポイント。

A0J の問題を解こう

- ・ 5 -Nested Controls I- A “Print a Rectangle”
 - ・ 5 -Nested Controls I- B “Print a Frame”
 - ・ 5 -Nested Controls I- C “Print a Chessboard”
- この 3 問は続き物なのでサクサクと解いてしまおう。

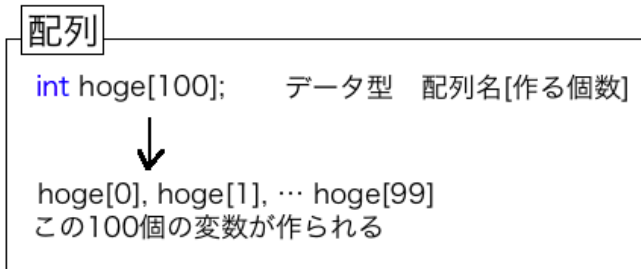
- ・ 7 -Nested Controls II- A “Grading”
- 二重ループは必要ない。if 文の練習のつもりで解こう。

- ・ 7 -Nested Controls II- B “How Many Ways?”

全通り試して数え上げる……のだがそのまえに「全通り」が何通りくらいあるか考えてみよう。この問題の制限時間は 1 秒、そしてコンピュータは 1 秒間に 1000 万から 1 億通りくらいを計算することが出来る。制限時間内に計算できそうだろうか？

1-7. 配列

大量の変数をまとめて作るのが配列。まとめて作られた変数たちは、個別の名前は与えられず番号で呼ばれるという悲しい運命にある…。



このように、配列名[番号]の形式で各変数を使うことが出来る。こうすることによるメリットは、この[番号]として変数とかも使えるので、いろいろと面白い操作ができることにある。

番号は0から始まるのに注意。個数100個の配列を作ったら、最後の要素は100-1番目となる。

```

1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int hoge[10];
6      // 要素数10の配列hogeを作る
7
8      for(int i=0; i<10; i++){
9          hoge[i] = 0;
10         // hogeの中身を全て0にする
11     }
12
13     for(;;){
14         // 無限ループ
15         int n, m;
16         cin >> n >> m;
17         // 二つの整数を入力、それぞれn, mに代入
18
19         hoge[n] = m;
20         // 配列hogeのn番をmに変える
21
22         for(int i=0; i<10; i++){
23             cout << hoge[i];
24             if(i!=9) cout << " ";
25             // hogeの中身を一行に表示する
26         }
27         cout << endl;
28     }
29
30     return 0;
31 }
```

・配列に慣れていない人はとりあえず左のプログラムを書いてみるべし。

このプログラムでは配列の基本的な挙動を確認できる。
無限ループの各周回では二つの整数入力 n , m をとる。そして配列の n 番目を m に変えて、配列の中身を全て出力する。

次のことを試してみよ

配列の中身が

3 1 4 1 5 9 2 6 5 3

になるような入力をつくれ。

9行目の `hoge[i] = 0` をコメントアウト（先頭にスラッシュ二つ `//` を書いて、コメントにするとどうなるか。

配列のパワーは、for 文を使って全部の要素をまとめて操作できることにある。

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5      int a[100];
6      int b[100];
7      int c[100];
8      // 配列a,b,cをつくる。どれも要素数は100
9
10     for(int i=0; i<100; i++){
11         // 繰り返しごとにループ変数は0, 1, ... 99と変化する
12         a[i] = i;
13         b[i] = i*i;
14         // 配列aのi番にはiそのものが、
15         // 配列bのi番にはiの二乗を入れる
16     }
17
18     for(int i=0; i<100; i++){
19         c[i] = a[i] + b[i];
20         // 配列cのすべての要素を、同じ番のa,bの要素の和にする
21     }
22
23     for(int i=0; i<100; i++){
24         cout << c[i] << endl;
25         // 配列cの中身をすべて出力
26     }
27
28     return 0;
29 }
```

このプログラムでは配列 a, b, c の 100 個もある要素を for 文でまとめて操作している。配列に慣れてない人は書き写してみると良い。

ちなみに 100 とはいわず 100 万くらいまでの要素数ならなんとかなる。

練習として、このプログラムの配列 c の要素の和がいくつになるか求めるプログラムを書き足せ。(正解は 333300)

A0J の問題を解こう

・ 6 -Array- A “Reversing Numbers”

```
1  #include<iostream>
2  using namespace std;
3
4  int main(){
5      int a[100];
6      int n;
7      cin >> n;
8      for(int i=0; i<n; i++){
9          cin >> a[i];
10     }
11
12     
13
14
15     return 0;
16 }
```

まず 7 行目で数列の長さ n を受け止めて、8-10 行目の for 文を使って続く n 個の数を配列 a に入力している。あとは頑張る。

・ 0533 “Contest”

これは今までとは違うところにある問題で、上のメニューの PROBLEM から Volume5 を選び、その中にある 0533 番の問題を開いてほしい。

なお、0533 で Volume 5 の 33 番目の問題ということだ。

これは 2008 年情報オリンピック予選 2 番の問題で、配列を使わなくても解けるが、まあ使ったほうが考えやすいだろう。

- ・ 多次元配列

配列を二次元的に確保することも出来る。三次元とか四次元とかも可能。

1次元配列A

A[0]	A[1]	A[2]	A[3]	A[4]
------	------	------	------	------

2次元配列A

A[0][0]	A[0][1]	A[0][2]	A[0][3]	A[0][4]
A[1][0]	A[1][1]	A[1][2]	A[1][3]	A[1][4]
A[2][0]	A[2][1]	A[2][2]	A[2][3]	A[2][4]

int A[3][5];と書けば、上の表のようなA[0][0]からA[2][4]までが作られる。全部の要素にたいして何かしたかったら、二重ループを使えばいい。

- ・ 6 -Array- B “Finding Missing Cards”

二次元配列。方針は問題文のとおりによればよい。

なんかうまく行かなかったら次の問題を先にやってから戻ってきてもいいかも。

- ・ 7 -Nested Controls- C “Spreadsheet”

二次元配列。慎重にやろう。

※配列の範囲外アクセス

```
int a[10];
```

で配列を作ると、a[0]からa[9]までが作られるが、ではa[20]とかa[100]とかを無理に使うとするとどうなるか…これが「配列の範囲外アクセス」と呼ばれるバグである。

実はa[20]のような範囲外の要素はコンピュータのメモリ上では、別の変数の記録場所だったり、あるいは何でもない空虚な場所だったりする。もしそれが大事な場所だったときは、実行中に

 : Runtime Error

というエラー（ランタイムエラー、実行時エラー）を食らうことがある。コマンドライン上では Segmentation Fault と出てくるかも。

しかし本当にやばいのは何でもない場所にあたって特にエラーが出ず、しかし想定外の動作をしているので答えは間違っているという事態で、なかなかデバッグに苦労するはめになる。for 文のループの終了条件を間違えた時などにやりがちなので注意。

1-8. 文字と文字列

IT用語辞典

e-Words

ASCII文字コード

文 字	10 進	16 進	文 字	10 進	16 進	文 字	10 進	16 進	文 字	10 進	16 進	文 字	10 進	16 進	文 字	10 進	16 進
NUL	0	00	DLE	16	10	SP	32	20	@	64	40	P	80	50	`	96	60
SOH	1	01	DC1	17	11	!	33	21	A	65	41	Q	81	51	a	97	61
STX	2	02	DC2	18	12	"	34	22	B	66	42	R	82	52	b	98	62
ETX	3	03	DC3	19	13	#	35	23	C	67	43	S	83	53	c	99	63
EOT	4	04	DC4	20	14	\$	36	24	D	68	44	T	84	54	d	100	64
ENQ	5	05	NAK	21	15	%	37	25	E	69	45	U	85	55	e	101	65
ACK	6	06	SYN	22	16	&	38	26	F	70	46	V	86	56	f	102	66
BEL	7	07	ETB	23	17	'	39	27	G	71	47	W	87	57	g	103	67
BS	8	08	CAN	24	18	(40	28	H	72	48	X	88	58	h	104	68
HT	9	09	EM	25	19)	41	29	I	73	49	Y	89	59	i	105	69
LF*	10	0a	SUB	26	1a	*	42	2a	J	74	4a	Z	90	5a	j	106	6a
VT	11	0b	ESC	27	1b	+	43	2b	K	75	4b	[91	5b	k	107	6b
FF*	12	0c	FS	28	1c	,	44	2c	L	76	4c	\	92	5c	l	108	6c
CR	13	0d	GS	29	1d	-	45	2d	M	77	4d]	93	5d	m	109	6d
SO	14	0e	RS	30	1e	.	46	2e	N	78	4e	^	94	5e	n	110	6e
SI	15	0f	US	31	1f	/	47	2f	O	79	4f	_	95	5f	o	111	6f
															DEL	127	7f

* LFはNL、FFはNPと呼ばれることもある。

* 赤字は制御文字、SPは空白文字(スペース)、黒字と緑字は図形文字。

* 緑字はISO 646で割り当ての変更が認められており、例えば日本ではバックスラッシュが円記号になっている。

(http://e-words.jp/p/r-ascii.html)

C言語で文字は char 型のデータ型で、シングルコーテーションに囲んで表す。
さて、この文字たちは、実は0から127までの特定の数字と対応している。それが上の表だ。

```
int a = 'a';
cout << a << endl;
```

のように無理やり int 型に代入して表示しても、その文字が数字のいくつかのかわかることが出来る。

表の0から31番の見慣れない人たちは「制御文字」と呼ばれるもので、ディスプレイとかプリンタとかの機械を操作するためのものである。

正直ほとんどが気にしなくてよいものなのだが0番と10番は必要で、0番のNULはC言語では「文字列の終端」を表していて、C言語文字列（後述）ではこれが出てくるところまでが文字列となる。'¥0'と書く。10番のLFはもっと大事で、これは改行であり、これが出てくると次の行に移る。'¥n'と書く。

さて、この表をみていると面白いことに気がつく。まず、AからZは65から90に、aからzは97から122に配置されている。つまり、大文字の文字に32を足せば小文字になるのである！

```
char a = 'A' + 32;  
cout << a << endl;
```

これを実行すると確かにAではなくaが出力される。32という数は'a'-'A'を計算すれば出てくるので、別に覚えなくてもいい。

次に、文字'0'から'9'は48から57に配置されている。つまり、数字文字から48を引けば、それが表している本当の数へと変換できるわけだ。

```
int a = '3' - 48;  
cout << a << endl;
```

これは3がちゃんと出力される。48という数字は'0'をint型にすれば出てくるので、別に覚えなくてもいい。

・ 文字列型 string

string型

```
ライブラリ読み込み  
#include <string>  
  
宣言  
string st;  
代入  
st = "This is a pen.";  
n番目の文字を調べる  
st[n] (配列と同様にnは0から始まる)  
  
連結する  
st = st+st; (This is a pen.This is a pen.になる)  
  
比較演算  
st1 < st2 (辞書順で(英語辞典で出てくる順番)どちらが大きいかわかる)  
st1 == st2 (同じ文字列か?)  
  
長さを調べる  
st.length()  
  
n文字目からm文字を取り出す  
st.substr(n, m)
```

課題

・ string 型には他にもいろいろな機能があるから、冊子の末尾にある参考サイトの C++リファレンスで、C++文字列の項目を調べてみよう。サンプルなどもたくさん出ている。

・ 9 -String- A “Finding a Word”

まずは次のプログラムを書いて、sample input をリダイレクトで入力してみよう。

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main(){
6      string w, st;
7      cin >> w;
8      for(;;){
9          cin >> st;
10         if(st=="END_OF_TEXT") break;
11
12         cout << st << endl;
13     }
14     return 0;
15 }
```

10 行目の比較演算で文字列” END_OF_TEXT” の出現を検知して、ループを脱出できていることと、空白や改行で入力が区切られていることを確認してほしい。

これが書けたら、この問題の正解を目指そう。

「大文字小文字は区別しません」をどうやって克服するかがポイント。普通の比較演算==では大文字小文字が異なれば異なる文字列だとされてしまう。

・ 9 -String- B “Shuffle”

問題文に書いてあるとおりに substr と連結を使えばできます。

・ 9 -String- C “Card Game”

比較演算を使えば簡単にできる。「辞書順比較」という用語はよく使うので覚えておこう。

1-9. 命令(関数)

命令(関数)とは、何か値を渡したら、それに何かをして、何らかの値が返ってくるようなものである。ただし、何も渡さなかったり、何も返ってこないものもある。渡す値を引数、返ってくる値を戻り値という。それらが存在しないとき、voidという。

構文 命令を作る

作る命令の名前

```
void hoge(){
  ...
}
```

引数なし、戻り値なし
voidは戻り値なしを表す

引数の型
引数名

```
void hoge(int a, int b, ... ){
  ...
}
```

引数あり、戻り値なし
小カッコのなかに引数を列挙する

戻り値の型

```
int hoge(){
  ...
  return ...;
}
```

引数なし、戻り値あり
returnの後ろに戻り値を書く

```
int hoge(int a, int b, ... ){
  ...
  return ...;
}
```

引数あり、戻り値あり

・ サンプルコード

```
1  #include <iostream>
2  using namespace std;
3
4  // 1を返すだけの命令
5  int return1(){
6      return 1;
7  }
8
9  // Hello!と表示する命令
10 void hello(){
11     cout << "Hello!" << endl;
12 }
13
14 // 引数nの回数だけHello!と表示する
15 void nhello(int n){
16     for(int i=0; i<n; i++){
17         hello();
18         // 上で作ったhello命令が使われている
19     }
20 }
21
22 // 引数aとbの大きい方を返す命令
23 int larger(int a, int b){
24     if(a>b) return a;
25     else return b;
26 }
27
28
29 int main(){
30     cout << return1() << endl;
31     nhello(10);
32     cout << larger(3, 1) << endl;
33     return 0;
34 }
```

ここでは5つの命令

return1

hello

nhello

larger

main

が作られている。

1. それぞれの命令の働きを答えよ
2. larger 命令と nhello 命令を組み合わせて、引数を2つとって、それらのうちの大きい方の回数だけHello!を出力する命令 lhello を作れ
3. larger 命令を組み合わせて、引数を3つとって、それらのうち一番大きい値を返す命令 larger3 を作れ。

K 会夏期講習 2013 情報講座

- ・ グローバル変数

A0J の上のメニューの COURSE

→Lesson

→Algorithms and Data Structures I

→1 Getting Started

→A Insertion Sort

を解くことを考える。

この問題では、適当な数列を小さい順に並び替える（ソートするという）ことを目標とする。そこで、次の方針を使う。

- ・ $k-1$ 番目までがソートされているとき、次のような操作を行えば k 番目までがソートされた状態になる。これを $k=0$ から $n-1$ まで繰り返す。

a[4]までがソートされている

1 3 4 6 7 5 2 9 8

X

7>5なので交換

1 3 4 6 5 7 2 9 8

X

6>5なので交換

1 3 4 5 6 7 2 9 8

4<5なので交換せず

1 3 4 5 6 7 2 9 8

a[5]までがソートされている

自信のあるひとは、ページをめくらずに、このままこの問題の正解を目指そう。
自信のない人は、次のページの空欄を埋めて、この問題に正解しよう。

```

1  #include <iostream>
2  using namespace std;
3
4  int n;
5  int a[100];
6
7  void printa(){
8      配列aの中身を
9      3 1 4 1 5
10     のような感じで一行出力
11
12 }
13
14
15 void swap(int i, int j){
16     a[i]とa[j]を入れ替え
17
18 }
19
20
21 void movek(int k){
22     a[k-1]までがソート済みの時
23     swapでa[k]を適切な所に動かす
24
25 }
26
27 int main(){
28     cin >> n;
29     for(int i=0; i<n; i++){
30         cin >> a[i];
31     }
32     for(int i=0; i<n; i++){
33         movek(i);
34         printa();
35     }
36     return 0;
37 }

```

これらはグローバル変数といって、
どの命令からでも読み書きできる変数である

まず注目して欲しいのは、4, 5 行目にある二つの変数である。28~31 行目を見て分かるように、これらには入力される数列の情報が記録される。このようにすべての命令の外側で作られた変数はグローバル変数と言って、どの命令からでも読み書きすることが出来る変数である。（逆に、main の中で作られた変数は swap 命令では使えなかったりする）

空欄になっている 3 つの部分の部分を埋めて、プログラムを完成させよう。行数などは多少違っていい。

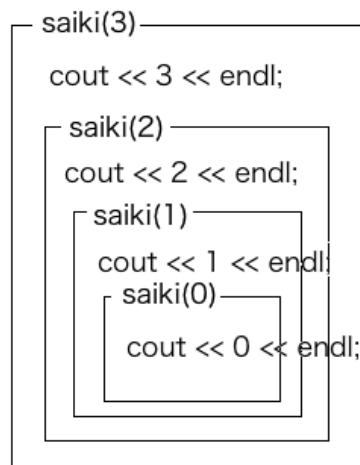
・再帰呼び出し

前のプログラムでは命令の中で別の命令を呼び出したりしたが、実は命令の中で自分自身の命令を呼び出すことも出来る。これを再帰呼び出しという。そんなことしたら無限ループが出来てしまいそうな気がするが、if文を使って、特定の条件の時だけ再帰呼び出しするようにすればそれは防げる。さて、こんなことをしてなにか良いことがあるのだろうか？

次のプログラムを書いてみよう。

```

1  #include <iostream>
2  using namespace std;
3
4  void saiki(int n){
5      cout << n << endl;
6      if(n>0){
7          saiki(n-1);
8      }
9  }
10
11 int fact(int n){
12     if(n==0) return 1;
13     else return n * fact(n-1);
14 }
15
16 int main(){
17     saiki(10);
18     cout << fact(5) << endl;
19     return 0;
20 }
```



4行目から始まる命令 saiki(n)は、引数 n をまず出力し、n が 0 より大きければ、saiki(n-1)を呼び出すというものである。

右図は saiki(3)を実行した時に起こることを表した図で、

saiki(3)が saiki(2)を呼び出し

saiki(2)が saiki(1)を呼び出し

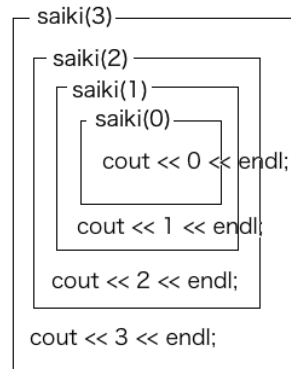
saiki(1)が saiki(0)を呼び出し

n が 0 より大きくなるなくなったのでここで再帰終了

という流れになる。つまり、3 2 1 0 の順番の出力が得られる。

11 行目からの命令 fact(n)は、n の階乗(1*2*3*... *n)を求めるプログラムである。(n-1)の階乗に n を掛けたら n の階乗になることから、このような再帰呼び出しでプログラムを書くことが出来る。

1. 命令 `saiki(n)` の実行の流れが右の図のようになるようにせよ。



2. `fact(n)` を書き換えて、1 から n までの和が出力されるようにせよ

3. 次のような再帰関数 `void saiki(int n)` を書け。
引数として整数 n をとり、
まず n をコマンドラインに出力し、
 n が 1 より大きければ、
 n が 2 で割り切れたら `saiki(n/2)` を呼び出し
 割り切れなければ `saiki(n+1)` を呼び出す

例 : `saiki(9)` → 0, 10, 5, 6, 3, 4, 2, 1

`saiki(13)` → 13, 14, 7, 8, 4, 2, 1

4. 次の再帰命令を書こう。これは 1 からまでの数を使ってできる 5 桁の数を小さい順に全て表示するものである。どうしてそうなるのか納得せよ。

```

4  int a[5]; // グローバル変数
5
6  void five(int k){
7      if(k==5){
8          // 配列aの中身を全て出力
9          for(int i=0; i<5; i++){
10             cout << a[i];
11             if(i!=4) cout << " ";
12         }
13         cout << endl;
14     }else{
15         for(int i=1; i<=5; i++){
16             // a[k]の値を1から5まで動かし、その都度再帰
17             a[k] = i;
18             five(k+1);
19         }
20     }
21 }
  
```

5. この命令 `five` を書き換えて、1, 2, 3, 4, 5 を並び替えてできる 5 桁の数を小さい順に全て列挙せよ。(同じ数字を 2 個以上含まないように改良する)
6. A0J Lesson: Algorithms and Data Structures の 5-Recursion/Divide and Conquer の、A を再帰を使って解け
7. A0J 1045 (Volume10 45 番) を再帰を使って解け

(おまけ) 参考サイトとか

僕がよく参考にしているサイト
問題を解いてみてわかんなくなったら見てみるといいかも

Programming place

http://www.geocities.jp/ky_webid/index_old.html

C/C++リファレンス (準公式)

<http://www.cpp11.jp/cppreference/index.html>

有名なところ

ロベールの C++教室

<http://www7b.biglobe.ne.jp/robe/cpphtml/index.html>

苦しんで覚える C 言語

<http://9cguide.appspot.com/>

※このテキストでやり残している C/C++の大事な機能

- ・ printf と scanf
- ・ C 言語の文字列
- ・ 数学関数 cmath
- ・ 構造体、クラス
- ・ STL
- ・ ポインタ
- ・ ファイルへの入出力 (リダイレクトではなく)

一度ちゃんとした本が、上にあげたような web サイトを使って勉強した方がいい。