目 次

第1章　第３回放送

1.1　第３回放送内容

1. 配列
2. if else 構文
3. 演習問題

1.2　配列

1.2.1　配列とは

前回の放送で変数というものを学びました。変数とは、一つの値を入れることができる箱と説明しました。では、ここで変数を５個作って見ましょう。

```
int hoge1;
int hoge2;
int hoge3;
int hoge4;
int hoge5;
```

同じような目的で使う変数を５個作ってみました。しかし、ここで１００個の変数が必要としましょう。そうしたらどうなるでしょうか。

```
int hoge1;
int hoge2;
int hoge3;
    ・
    ・
    ・
```

と宣言だけで大変な事になってしまうでしょう。そういったときに、同じ種類の変数を一まとまりのデータとして扱うために配列というものが存在します。つまり、１００個の変数を一気に作ってしまうものを配列といいます。

1.2.2　一次元配列

では、まずは変数を５個、配列として作るにはどのようにしたらよいでしょうか。配列の宣言は次のようになります。

```
int hoge[5];
```

変数の宣言と似ていますが、変数名[1]の後に [5] と書かれています。これは変数を５個作れという命令です。よって、１００個変数を作りたい場合は

```
int hoge[100];
```

とすればよいわけです。

書式

データの型　配列名 [要素の数];

1.2.3　配列への値の代入

配列を作った後は値を代入してみましょう。ここで、 `int hoge[5];` の配列のイメージは図 1.1 のようになります。

図 1.1: 配列のイメージ

ここで、配列の一番最初の変数に 10 という値を代入したい場合、

```
hoge[0] = 10;
```

[1]箱の名前の事 ここでは hoge

とします。注意して欲しいことは、図 1.1 を見てわかるように、配列は [0] は必ず 0 から始まります。よって代入できるのは

```
hoge[0]
hoge[1]
hoge[2]
hoge[3]
hoge[4]
```

です。hoge[5] は存在しないので値を代入できない事に注意してください。

1.2.4 配列の初期化

配列を初期化する場合は、つぎのような書き方をすることができます。

```
int hoge[5] = { 1,2,3,4,5 };
```

これは、次のように書いたのと「ほぼ」同じです。[2]

```
hoge[0] = 1;
hoge[1] = 2;
hoge[2] = 3;
hoge[3] = 4;
hoge[4] = 5;
```

箱の中身は図 1.2 となります。

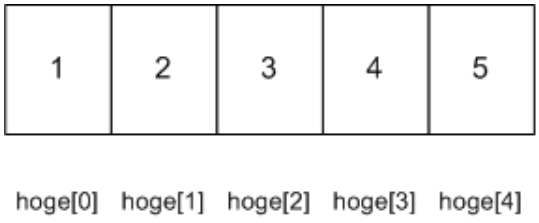

図 1.2: 配列のイメージ

また、次のようにしたらどうなるでしょうか。

[2]完全に同じではないです。なぜなら初期化と代入は違うからですが、あまり気にしなくて良いです。

```
int hoge[5] = {1,2};
```

残り3つはどうなるでしょうか。この足りない部分は0が自動的に代入されます。つまり、

```
int hoge[5] = {1,2,0,0,0};
```

と同じです。さて、今度は次のようにしたらどうなるでしょうか。

```
int hoge[] = {1,2,3,4,5};
```

今度は、[] の中の数字を省略しました。このように書くこともできます。こうすると、要素数を勝手にコンパイラが設定してくれます。つまり、[] の中に5を勝手に入れてくれます。しかし、こういうのはダメです。

```
int hoge[];
```

これは要素数が0なので、 `int hoge[0];` という事になりますが、変数を1つも作らないというのは意味がありません。こういったことはできないので注意してください。

1.2.5　文字配列

変数にはもちろん文字を入れることもできます。文字を入れる場合はchar型を使うと前回の放送で説明しました。ここで、実際char型に文字を入れてみましょう。入れることができるのは英字1文字だけです。

```
char str;
str = 'A';
```

ここで、英字を入れる場合、' 'で囲む事に注意してください。" "ではないです。

では、次に配列を使ってみましょう。

```
char str[6] = {'H' , 'E' , 'L' , 'L' , 'O' , '\0' };
printf("%c%c%c%c%c%c",str[0],str[1],str[2],str[3],str[4],str[5]);
```

str[5]、つまり最後の要素に見慣れない文字が含まれています。これはNULL文字[3]といい、文字列の最後には必ずいれなければならないものです。つまり、この記号があると、文章はここで終わりという意味になります。そして、printfで表示

[3]ナル、またはヌルと読む

させています。ここで、１文字表示する場合は%c を使うんでしたね。しかし、この printf は非常に面倒です。そこで、他に方法がないかと探してみると、%s というものが存在します。これは、文字列を表示というものです。使い方は次のようになります。

```
char str[6] = {'H' , 'E' , 'L' , 'L' , 'O' , '\0' };
printf("%s",str);
```

このようになります。str には添え字、つまり [0] などが付いていませんね。実はこれ、とても難しいことをしているのです。これを説明するには、ポインタをかなり理解していないとできません。簡単に説明すると、str は &str[0] の略した書き方なのですが、ぜんぜん分けがわかりませんね。今は文字列を表示したい場合このようにするのだということを覚えておいてください。

さて、気を取り直して、今度は配列の初期化の部分を更に簡単にしてみましょう。

```
char str[6] = {"HELLO"};
```

さらに略して、

```
char str[6] = "HELLO";
```

とすることができます。最後に\0 が付いていませんが、このように書くと勝手に\0 が付け加えられます。ここで気をつけなければならないことがあります。次のような場合です。

```
char str[5] = "HELLO";
```

このようにすると、５文字で配列の要素も５つですが、さっき説明したように最後に\0 が勝手に付け加えられるので、これでは要素が足りなくなってしまいます。こういった事に注意してください。

1.3 if else 構文

if else 構文は、もし何々だったら何々の処理をする、そうでなかったら違う処理をするといった場合に使います。イメージは図 1.3 のようになります。

1.3.1 if else の書き方

では、図 1.3 を C 言語になおしてみますと、次のようになります。

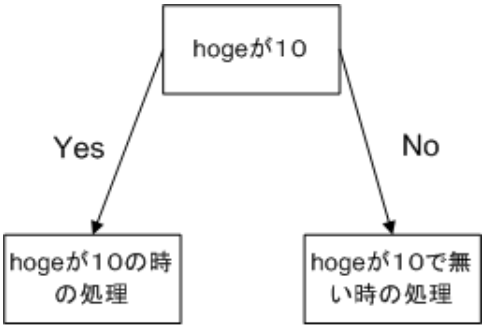

図 1.3: if else のイメージ

```
int hoge = 10;

if( hoge == 10 ) {
//yes の時の処理
} else {
//no の時の処理
}
```

ここで、if( ) のカッコの中には条件を書きます。この場合、もし hoge が 10 であったらという処理を表しています。ここで注意して欲しいのは、hoge = 10 ではなく、hoge == 10 であるということです。= とは代入という意味だと説明しました。数学で言うイコールという意味にしたい場合は、==とイコールを二つつなげて書かなければならないことに注意してください。さて、yes だけの処理しか必要が無い時には次のようにすることができます。

```
if( hoge == 10 ) {
 //yes の時の処理をここに書く
}
```

また、条件を複数書きたい場合、次のようにします。

```
int hoge = 8;
```

```
if( hoge == 10 ){
//A の処理
}else if( hoge == 9) {
//B の処理
}else if( hoge == 8 ) {
//C の処理
}else{
//D の処理
}
```

この場合 C の処理が実行されることになります。ここでもし hoge が 9 であるならばBの処理が行われますし、6 ならば D の処理が実行されることになります。

1.3.2 非０の判定式

次のように書いたらどうなるでしょうか。

```
int hoge = 1;

if( hoge ){
//処理を行う
}
```

さて、カッコの中には何か式を書かなくてはなりませんが、この場合は変数名の hoge としか書いてありません。これはどういう意味でしょうか。

実はこういった場合、カッコの中の値が０であるか０以外であるかで判定します。ここでは、０以外の場合何か処理を行い、０の場合は処理を行いません。つまり、この式は以下のように書き換えることができます。

```
if( hoge != 0 ) {
//処理を行う
}
```

ここで新しい表現 != というものが出現しました。これは hoge が 0 でなかったらということを意味しています。
では別の例を挙げてみます。

```
if( 1 ) {
//A の処理
} else {
//B の処理
}
```

この場合 A の処理が実行されます。さっき説明したように 0 以外の数字の時は A の処理が実行されます。もしカッコの中の数字を 0 にしたら B の処理が実行されます。

1.3.3　関係演算子

２つの変数を比較して、式が正しいか誤りかを求める時に使う演算子を関係演算子といいます。例えば次のようなものが存在しています。

```
x <  y xが y より小さかったら真、そうでなかったら偽
x >  y  xが y より大きかったら真、そうでなかったら偽
x <= y  xが y 以下であれば真、そうでなかったら偽
x >= y  xが y 以上であれば真、そうでなかったら偽
x == y  xが y と等しければ真、そうでなかったら偽
x != y  xが y と等しくなければ真、そうでなかったら偽
```

後で演習問題を行いますので、その時使い方を確認してください。

1.3.4　論理演算子

例えば次のようなコードを書いたとします。

```
int hoge = 10, piyo = 20;

if( hoge == 10 ) {

        if( piyo == 20 ) {
         //処理Aを行う
        }
}
```

さて、この式はつまり hoge が１０であり、かつ piyo が２０である時、処理Aが実行されるということです。ここで論理演算子というものが存在します。

```
x == 10 && y == 20      xが１０かつyが２０
x == 10 || y == 20      xが１０またはyが２０
!x                      xの否定
```

つまり、さっきの例をこの論理演算子を使って書き直すと次のようになります。

```
if( hoge == 10 && piyo == 20){
//処理A
}
```

ここで、x の否定とはどういうことを意味しているでしょうか。例を挙げてみますと、

```
int hoge = 10;

if( !hoge ){
//処理A
}
```

条件が !hoge ではなく hoge であった場合は処理 A が実行されますが、この場合はされません。逆に hoge の値が０であるとき、処理Aが実行されます。

1.4　演習問題

さて、今まで勉強したことを元にいろいろプログラムを作ってみましょう。

1.4.1　絶対値を求めるプログラム

まずは絶対値を求めるプログラムを作ってみましょう。絶対値とは0からの距離のことを言います。つまり、-7ならば0からの距離は7で、+10であったら0からの距離は10です。

絶対値を求めるプログラムの仕様

キーボードから数字を受け取り、その絶対値を表示する

```
#include <stdio.h>

int main(void)
{
int x;

printf("数字を入力してください:");
scanf("%d",&x);

if( x < 0 ) {
x = -x;
}

printf("絶対値は%dです\n",x);
return 0;
}
```

1.4.2　最大値を求めるプログラム

最大値を求めるプログラムの仕様

3つの数字を入力し、その中で一番大きなものは何かを返す

```
#include <stdio.h>

int main(void)
{
int x,y,z;

printf("３つの値を入力してください\n");
printf("１つ目："); scanf("%d",&x);
printf("２つ目："); scanf("%d",&y);
printf("３つ目："); scanf("%d",&z);

if( x >= y && x >= z) {
printf("一番大きな値は１つ目です");
}else if( y >= x && y >= z ) {
printf("一番大きな値は２つ目です");
}else if( z >= x && z >= y ) {
printf("一番大きな値は３つ目です");
}
return 0;
}
```

やり方は一通りではありません。他の例を示します。

```
#include <stdio.h>

int main(void)
{
int x,y,z;
int max;

printf("３つの値を入力してください\n");
printf("１つ目："); scanf("%d",&x);
printf("２つ目："); scanf("%d",&y);
printf("３つ目："); scanf("%d",&z);
```

```
max = x;
if( max < y ) {
max = y;
}
if( max < z ) {
max = z;
}

printf("一番大きな値は%d です",max);

return 0;
}
```

1.5　本日の講義のおさらい

- 配列

  変数をまとめて作りたい場合に使う

- if else 構文

  もし条件式が真であったら〜の処理を、そうでなかったら〜〜の処理を行う

参考文献

[1] Brian W. Kernighan, Dennis M. Ritchie: The C Programming Language Second Edition.

[2] ハーバートシルト著 トップスタジオ訳: 独習 C 第三版.

[3] 田中 敏幸著 C 言語によるプログラミングの基礎

[4] 柴田 望洋 明解C言語

[5] 松本 健一 上田 悦子 安室 喜弘 ２００６年度プログラミング演習（初級コース）課題

http://chihara.naist.jp/people/STAFF/yasumuro/Pub/c-ensyu2006/

[6] Ryo Kawahara C/C++プログラミング初心者講座

http://www.stat.phys.kyushu-u.ac.jp/~ryokawa/cbegin2_3/pdf/cbegin.pdf

[7] TOMOJI 初心者のためのポイント学習 C 言語

http://www9.plala.or.jp/sgwr-t/

[8] 小出 俊夫 C言語入門インターネット版

http://homepage1.nifty.com/toshio-k/prog/c/

[9] 赤坂 玲音 C 言語入門

http://wisdom.sakura.ne.jp/programming/c/index.html