ソフトゼミA　第7回　ポインタ

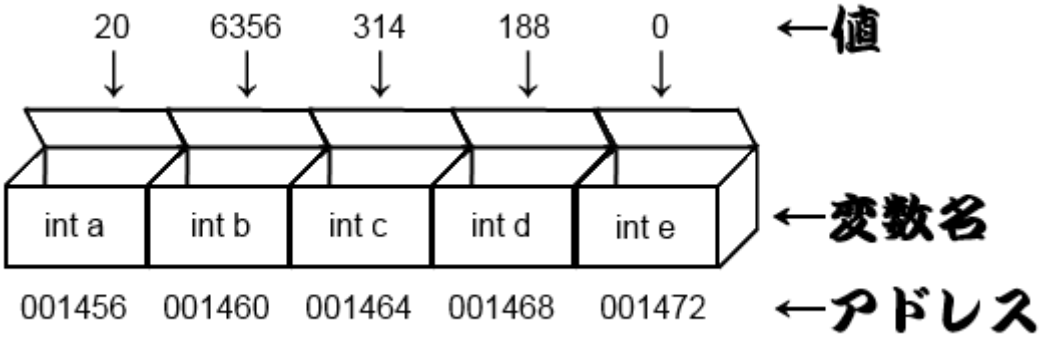
ゼミ A の最終回である第 7 回ではポインタを学習します。ポインタの概念は初めて触れ合う人にはかなり理解しにくいので、疑問を感じたら遠慮なく近くのソフト班員まで質問してください。

アドレス

ポインタとは変数を格納するアドレスを記録するための変数です。と、言われてもピント来ないと思います。そこでまずアドレスとはなんぞや、というのを説明します。

変数というのは第 2 回で説明したとおり、様々な値を格納するための**「箱」**です。そしてアドレスとはその**「箱」**がある場所を表した住所です。例えば変数を**「家」**とするとアドレスは**「住所」**、値は中に住んでいる**「住人」**となります。

アドレスは&[変数名]で表されます。例えば「test」という変数があれば、「test」のアドレスは「&test」で表すことができます。



ポインタ

ポインタとは、上で説明したアドレスを格納する専用の変数です。上の図で示すと「001456」「001460」「001464」「001468」「001472」というアドレスを格納する変数です。

宣言の仕方はこのようになります。

[変数の型] *[変数名];	（例）int *p;

※ここで*（アスタリスク）を付け忘れると普通の変数になってしまうので注意

※int 型の変数は int 型のポインタしか、char 型の変数は char 型のポインタしか使えないので注意！

ポインタの使い方①

今までアドレスとポインタの概念を説明しましたが、理解できましたか？
この概念が理解できないと先に進むのが難しいのでわからない人は是非とも質問してください。

それではポインタの実際の使い方を説明したいと思います。
文章で書いても分かりづらいと思うのでソースコードを載せます。
以下のコードは変数のアドレスをポインタに格納し、表示する動作を行っています

a07_1.c

```
#include<stdio.h>
int main(void){
        int a=20;          //変数を用意
        int *p;            //ポインタを用意
        p=&a;              //変数のアドレスをポインタに入れる
        printf("値:%d¥n",a);          //値の出力
        printf("アドレス:%d¥n",&a); //変数からのアドレス出力
        printf("アドレス:%d¥n",p);  //ポインタからのアドレス出力
        return 0;
}
```

このように打ち込み、実行すると以下のようになると思います。

```
C:¥Users¥[redacted]¥Dropbox¥2013年エレ研dropbox¥ゼミA暫定資料>A7_1.exe
値:20
アドレス:1638224
アドレス:1638224
```

あれ？アドレスの値が違う・・・と思うかもしれません。しかしアドレスというのはコンピュータが実行するたびに空いている番地に勝手に割り振っていくので、違っていても気にしなくて結構です。

上のソースコードのように、ポインタは「p=&a;」という形で代入します。
普通の変数に&を付けることでもアドレスを表現することができます。
その例が下から4行目の「printf("アドレス:%d¥n",&a);」であり、下から3行目のようにポインタを出力したのと全く同じアドレスを出力することができます。

これらをまとめると、ポインタｐにａのアドレスが入っている場合、以下のようにまとめることができます。

変数	ａの値	ａのアドレス
int a;(int 型の変数)	a	&a
int *p;（int 型のポインタ）	*p	p

＆（アンドは）アンドレス、＊（アスタリスク）はアタイと覚えると、覚えやすいです。

ポインタの使い方②

ここからは実際のポインタの使い方を説明します。
まずは以下のソースコードを見てください、内容は変数を交換するということです。
a07_2.c

```
#include<stdio.h>
void swap1(int a,int b){
        int c;
        c=a;
        a=b;
        b=c;
}
void swap2(int *a,int *b){
        int c;
        c=*a;
        *a=*b;
        *b=c;
}
int main(void){
        int a,b;
        a=10;
        b=20;
        printf("A:%d B:%d¥n");//A と B の値を出力
        swap1(a,b);//a と b の値を直接交換しようとする。
        printf("A:%d B:%d¥n");
        swap2(&a,&b);//a と b の値をアドレスで交換する。
        printf("A:%d B:%d¥n");
        return 0;
```

```
}
```

このように入力して実行すると、以下のような結果が出ます。

```
C:¥Users¥ [redacted] ¥Dropbox¥2013年エレ研dropbox¥ゼミA暫定資料>bcc32 A7_2.cpp
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
A7_2.cpp:
警告 W8004 A7_2.cpp 7: 'b' に代入した値は使われていない(関数 swap1(int,int) )
警告 W8004 A7_2.cpp 6: 'a' に代入した値は使われていない(関数 swap1(int,int) )
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

C:¥Users¥ [redacted] ¥Dropbox¥2013年エレ研dropbox¥ゼミA暫定資料>A7_2.exe
A:20 B:10
A:20 B:10
A:10 B:20
```

ここで警告が出ていますが、これは swap1 が機能していないということを示しているので、正常に動いています。

出力結果を見てみると、一番上から交換前、swap1で交換した後、swap2で交換した後となりますが、2番目の swap1では交換できていないことがわかります。これは何故かというと swap1での変数 a,b,c は関数内のローカル変数であるため、変数の有効範囲はその関数の中に限られてしまうからです。void 型ではなく int 型の関数であれば値を１つ返すこともできますが、今回のように2つ以上の変数を戻すことができません。

しかしながら swap2は「swap2(&a,&b);」とあるように。ポインタを関数に受渡しています。そのために関数内のポインタ a,b は main 文の変数 a,b を直接指しているために main 関数の外であってもローカル変数 a,b を操作することができるのです。

このようにポインタは関数の外からであっても複数の変数を操作することができるのです。

・複数の変数を関数から戻す方法

少々道を外れますが、関数から複数の変数を戻す方法を紹介します。

上ではポインタを使って戻す方法がありましたが、そのようなやり方は他にいくつか存在します。4つくらい例を挙げます。

1つ目は、戻したい変数全てを前回（第6回）学んだグローバル変数にしてしまう、という手です。これは main 文の更に外であるグローバルなところ（#inculde <stdio.h>と同じ階層）で変数を宣言することにより、同じファイル内ならばどこからも変数を参照できるようになります。

しかしながら、グローバル変数はどこからでも書き換えができるためバグが起きやすく、どれが使用した関数かわからなくなってしまうので、多用しすぎるのは禁物です。

2つ目は、第5回で勉強した構造体を使うという手です。例えば「int test(void);」という関数では1つしか変数を返すことができませんが、これを構造体で返すようにすれば、構造体に含まれる多くの変数を一気に返すことができます。

3つ目は前回学習した static 指定子を用いる方法です。static 指定子を用いると関数が終了しても値が保存され続けますが、こちらもグローバル変数と同様の問題があるので、多用は控えるべきです。

そして４つ目に配列引数という方法があります。配列を配列名の形で関数内に受け渡すことで、受け渡した後の関数の中でも配列の値を操作することが出来ます。

具体例として以下のプログラムを上げてみます。

a07_3.c

```
#include <stdio.h>
void hasu(int a[]){
	int i;
	for(i=0;i<5;i++){
		scanf("%d",&a[i]);
	}
}
int main(void){
	int a[10],i;
	hasu(a);
	for(i=0;i<5;i++){
		printf("a[%d]:%d¥n",i,a[i]);
	}
	return 0;
}
```

本来、a[0]～a[9]の値は hasu 関数の中で入れているので main 文の中には影響しないはずですが、実際に動かしてみると入れたとおりの値が出力されます。

実は配列というのはポインタであり、a というアドレスに[]の中の数字の分だけ足し算を行っているのです。そのために関数の外からでも書き換えができるのです。

・演習問題

１，配列 test[10]を宣言し、test[0]から test[9]までのアドレスを表示せよ。

２． main 関数内で、それぞれ異なる値の入った変数 a,b,c を宣言し、
関数 void asce(int *p, int *q, int *r)を、main 関数からこの関数を asce(&a,&b.&c)
と呼ぶと、a,b,c が昇順に並ぶように実装せよ。

・ゼミ B の日程について

次回のソフトゼミ A 第8回は復習回ですので、今までの内容が理解できている方や予定が入っている方などは来なくても構いません。もし来る場合は自分がわからなかった分野の資料を持ってきてください。ただ、1問だけ上級問題を用意しましたので解いてみたい！という方もよかったら来てみてください。

本題に入りまして、いよいよ来週からゼミ B が始まります。ゼミ B ではゼミ A で習った C 言語の知識を生かしてゲーム制作を行っていきます。

ソフトゼミＢ日程			
種別	回数	活動日	活動内容
導入	第1回	5/28(火)	VC++, DX ライブラリの導入
シューティングゲーム	第2回	5/30(木)	画像処理と移動
	第3回	6/4(火)	自機・敵機の弾の描画と移動
	第4回	6/6(木)	あたり判定
	第5回	6/11(火)	ファイル分割
アクションゲーム	第6回	6/13(木)	プレイヤーの移動と重力
	第7回	6/18(火)	ブロック判定
	第8回	6/20(木)	スクロールとファイル分割
（予備）	（第9回）	6/25(火)	（予備日）
	（第10回）	6/27(木)	

- **Microsoft Visual C++ 2010 Express Edition の導入**

  http://www.microsoft.com/visualstudio/jpn/downloads

  から「Visual Studio 2010 Express」の中の「Visual C++ 2010 Express」をインストールしてきてください。最新版として「Visual Studio Express 2012」が出ておりますが、動作確認をしておりませんので今回は「Visual Studio 2010 Express」で統一するようお願いします。また間違えて「Visual C# 2010 Express」をインストールしないよう気をつけてください。

  またインストール終了後、Visual C++を起動し、「ヘルプ」→「製品の登録」からユーザー登録をお願いします。

- **DX ライブラリの導入**

  http://homepage2.nifty.com/natupaji/DxLib/dxdload.html

  から、一番上にある「Visual C++用」をダウンロード→解凍してください。
  解凍したフォルダは C ドライブ直下などのわかりやすいところに置いておくことをオススメします。

  またデスクトップに解凍したフォルダ「DxLib_VC/help」の中にある index.html のショートカットを作成しておくと、今後のゲーム制作で役に立つと思います。