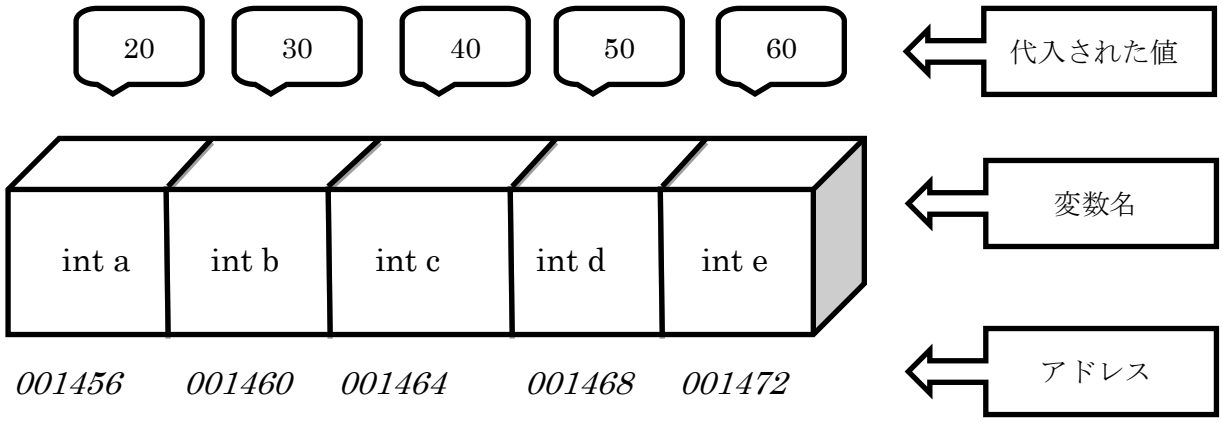
ソフトゼミ A 2015 第 7 回

ポインタ

今回でゼミ A は一応最終回です。今までお疲れ様でした。
今回はポインタというものについて学びます。
今までより少し難しい部分も多いかもしれませんが、時間をかけてゆっくりと理解していきましょう。

➢ アドレス

ポインタとは変数のアドレスを記録する変数のことです…と言われてもパッとしませんね。まずはアドレス(番地)というものについて説明しましょう。
変数とはさまざまな値を入れるための箱のようなものです。アドレスとは、その箱の位置を記した所謂住所のことです。
たとえば、int a=20,b=30,c=40,d=50,e=60 とすると、



(イメージ図)

軽く説明すると、例えば int a を例にとると「“001456”という場所に割り振られた“a”という変数に“90”という数字が代入されている」という感じです。
アドレスは任意で決めることができません。コンパイル時に自動的に各変数に割り当てられます。
変数のアドレスは&[変数名]で表すことができます。上の図で説明すると、a のアドレスは&a で表すことができます。

➢ ポインタ

それではポインタの説明に移りましょう。最初に説明したのを繰り返すとポインタとは変数のアドレスを記録する変数のことです。前の図で説明すると、「001456」、「001460」、「001464」…これらのアドレスを記録するための変数となります。
続いてはポインタの使い方を説明していきます。

■　宣言

まずはポインタを宣言してみましょう。
宣言の仕方は、

```
[データの型] *[配列名]
```

例としては「int *a」のような形になります。*(アスタリスク)を忘れてしまうと普通の変数の宣言になってしまうので気を付けましょう。また、int 型の変数は int 型、char 型の変数は char 型のポインタしか使えません。

■　設定

上で宣言したポインタに変数のアドレスを入れてみましょう。

```
int a=20;          //変数の宣言
int *p;            //ポインタの宣言
p=&a;              //変数のアドレスをポインタに代入
```

これで変数 a のアドレスがポインタ変数 p に入ることになります。ページ 1 の図で示すと p に a のアドレス「001456」が代入されました。3 行目のポインタ変数 p には*がつかないことに注意しましょう。

これらをまとめると次のようになります。
もし上の 3 行目、p=&a、ポインタ p に a の値が入っている場合は、

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

このようになります。

■　表示

そして、上の宣言、設定のまとめとしてこれらを通して表示してみましょう。

設定などは具体的にしないとわかりませんからね。1 ページ目の図に沿ってソースコードを進めてみましょう。

```
#include<stdio.h>
int main(void){
        int a=20;                       //変数 a を宣言
        int *p;                          //ポインタ p を用意
        p=&a;                           //変数 a のアドレスをポインタ p に入れる
        printf("a=%d¥n",a);             //a の値を出力
        printf("アドレス:%p¥n",&a);   //変数 a を使用しアドレスを出力
        printf("アドレス:%p¥n",p);      //ポインタ p を使用しアドレス出力
        return 0;
}
```

さて、これを実行すると 3 行にわたって出力されるはずです。

(出力例)

```
C:¥Users¥Public¥Documents¥エレ研>07.0
a=20
アドレス:0018FF50
アドレス:0018FF50
```

次からは実際の使い方について説明していきます。

■　交換

ポインタを使って変数の交換をします。まずはソースコードを見てください。

```
#include<stdio.h>
void swap1(int a,int b){                    //変数での交換
        int c;                              //変数 c を用意して
        c=a;                                //c に a を代入
        a=b; //a に b を代入
        b=c; //b に c を代入
}
void swap2(int *a,int *b){                  //アドレスでの交換
        int c;                              //変数 c を用意して
        c=*a;                               //c に a のアドレスを代入
        *a=*b;                              //a に b のアドレスを代入
        *b=c;                               //b のアドレスに c(a のアドレスの値)を代入
}
int main(void){
        int a,b;
        a=20;
        b=30;
        printf("a:%d b:%d¥n",a,b);          //a と b を出力(1 回目)
        swap1(a,b);                         //変数で交換
        printf("a:%d b:%d¥n",a,b);          //a と b を出力(2 回目)
        swap2(&a,&b);                       //アドレスで交換
        printf("a:%d b:%d¥n",a,b);//a と b を出力(3 回目)
        return 0;
}
```

これを実行してみると次のように表示されるはずです。

```
C:¥Users¥Public¥Documents¥エレ研>07.1
a:20 b:30
a:20 b:30
a:30 b:20

C:¥Users¥Public¥Documents¥エレ研>
```

警告文がコンパイル時に出てくると思いますが、無視して大丈夫です。理由は

あとで説明します。

さて、実行されて出力された結果が 3 行あると思います。1 行目は代入された値をそのまま出力したもの、2 行目は swap1、つまり変数で直接 a と b を交換した結果、3 行目は swap2、つまりアドレスを交換した結果の出力となっています。しかし 2 行目の出力、つまり swap1 では交換が成功していません。この理由としては、swap1 内での変数 a,b,c はそれぞれローカル変数、つまり変数の有効範囲がその関数の中に限られている変数のため、変数外での交換結果の出力に失敗してしまったのです。void 型ではなく、int 型の関数では値を一つ返すことができますが、今回のように 2 つ以上になると変数を戻すことはできなくなってしまいます(この失敗が警告文の正体でもあります)。

さて、成功させるには…ということで3行目の結果、swap2 を見てみましょう。swap2 では&a,&b と変数 a,b のアドレスを渡しています。これを交換することによって変数の有効範囲でない関数外、今回だと main 文の中にまで影響を与えることができるのです。

このようにアドレスをうまく使うと関数外でも複数のローカル変数を操作することができます。このように有効範囲を超えて関数を交換するような操作をしないといけない場合にはアドレスがとてもよく役に立ちます。

➢ 配列因数

さて、少し道を外れますが、上のように複数の変数を有効範囲を超えて関数から戻す方法を 5 つほど紹介していきたいと思います。

①アドレス

今回紹介した方法です。

②グローバル変数

前回少しだけ触れた方法です。前回も説明しましたが、グローバル変数の特徴はそのファイルの中ならいつでもどこでも変数の操作ができることです。この点がメリットにもデメリットにもなりうるため、使うときは注意しましょう。

③構造体

構造体に含まれた多くの変数を一気に返すことができるようになります。

④static 指定子

前回∀に出てきた方法です。これを使えば関数が終了しても値は変わらず保存され続けます。これもどこでも操作できてしまうというグローバル変数と同様の弱点があるので注意しましょう。

⑤配列因数

配列を配列名で関数に受け渡すことで配列の値を操作することができます。
具体例を見てみましょう。

```
#include<stdio.h>
void hairetu(int a[]){
        int i;
        for(i=0;i<3;i++){
                scanf("%d",&a[i]);
        }
}
int main(void){
        int a[10],i;
        hairetu(a);
        for(i=0;i<3;i++){
                printf("a[%d]:%d¥n",i,a[i]);
        }
        return 0;
}
```

変数の交換のプログラムの swap1 のように関数内での値の操作は関数外まで及ばず、ここだと hairetu 関数の中でしか操作できないので main 文には影響を与えないはずです。しかし、上のプログラムでは関数内での操作、“値を読み込み、それぞれ代入”された値が main 文にも反映されています。これは swap2 での影響と同じようになっています。

このように、配列というのは実はポインタであり、“a”というアドレスに[]の中の数字の分だけプラスされているのです。そのため関数外からでも書き換えができます。

➢ 練習問題

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してください。
2.main 関数内で、2 つの変数 a,b を宣言した後それぞれ任意の数を読み込み、和と差を返す関数を実装してください。

➢ ゼミ B の日程について

ゼミ A は今回が実質的な最終回です。お疲れ様でした。わからなかったところは他のサークル員に聞くなり家に帰って自力で調べるなりしてみてください。

そしてゼミ A が終わったので今度はゼミ B が始まります。ゼミ A で学んだことを応用して簡単なゲームを作っていきます。日程は以下の通りです。

ソフトゼミB日程				
種別	回数	活動日	活動場所	内容
導入	1	5月26日	0306	VC++、DXライブラリの導入
シューティング	2	5月28日	0306	画像処理と移動
	3	6月2日	0306	自機・敵機の弾の描画と移動
	4	6月4日	0306	当たり判定
	5	6月9日	0306	ファイル分割
アクション	6	6月11日	0306	プレイヤーの移動と重力
	7	6月16日	0306	ブロック設定
	8	6月18日	0306	スクロールとファイル分割

ゼミ B 開始に伴い、導入等の操作が再び必要になります。余裕のある方はゼミ B 初回までに VC++と DXLib を導入しておいてください。

■ Microsoft Visual C++ 2012 Express Edition(VC++)の導入

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

上のページから「Visual Studio 2012 Express」の中の「Visual C++ 2012 Express」をインストールしてきてください。他のバージョンでの動作確認はしていないのでお勧めはできません。また、

http://www.meiji.ac.jp/isc/msca/

ここに書かれている通り、明大生は Visual Studio の有償版を無料で使うことができるみたいです。

インストールしたら VC++を起動して「ヘルプ」→「製品の登録」からユーザー登録をしておいてください。

■ DX ライブラリの導入

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

今回のゲーム制作にあたり、ゲーム制作を大幅に手助けしてくれるツールです。上のページから一番上にある「Visual C++用」をダウンロード→解凍しておいてください。解凍したフォルダは C ドライブ直下などわかりやすい場所に置いておくことをお勧めします。

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立つと思います。