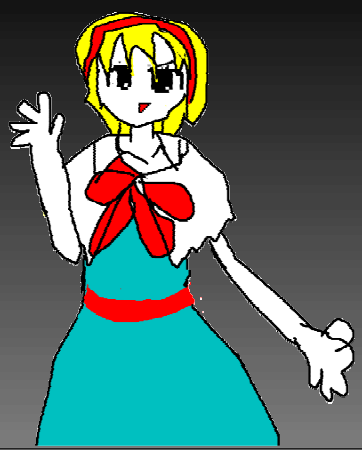
明治大学エレクトロニクス研究部
ソフト班
ソフトゼミC　一日目



ソフトゼミCの概要

ソフトゼミAの延長線上です。
ゼミBはDXライブラリを使ってのゲーム制作が主でしたが、まだプログラミング自体について、またやります。
またゼミAと同じく、CUI(これなんてUNIXインターフェイス？)で動くプログラムを使って学びます。

一日目と二日目の前半は文字列について、二日目の後半と三日目はC++に駒を進めて、オブジェクト指向について学びます。

だいたい、そんなとこ。

ソフトゼミCの効能

プログラミングについて、理解と意味不明さが増します。

文字列について学ぶことにより、ゲーム上で文字が表示することができたり、ノベルゲームのエンジン等も作ろうと思えば作れるようになります。ようはゼミAで全く触れなかった文字全般についてです。

オブジェクト指向というのは、今まで関数主体だったC言語に対し、モノを主体としたプログラミングの方法です。今のプログラミング言語では大抵はオブジェクト指向になっているので、学んでも損はありません。また情報学科の方は2年になると、なんでもかんでもオブジェクト指向にするJavaを学ぶそうなので、今のうちにオブジェクト指向を理解しておくといいかもしれません。今回はC++を使いますが……

なら、始めようか

そうだね。

章前問題

テキストにある問題をやってくれ。

補足:
フィボナッチ数列について
1, 1, 2, 3, 5, 8, 13, 21, 34, ...
前の項と前の前の項を足した値がなるとかってそんなやつ。

第一章　ポインタの前に

型にはそれぞれ役割がある。

int型:整数を扱う程度の能力
float型:単精度小数点を扱う程度の能力
double型:倍精度小数点を扱う程度の能力
char型:文字を扱う程度の能力

•double型では小数点も整数も扱えるが、int型では小数点は扱えない。
•float型使うよりかは、double型使っておけ
•charは「きゃら」って読んだり「ちゃー」って読んだり流派がある

第一章　ポインタの前に

sizeof演算子

•一見、関数のように扱える(気がする)
ex) printf(“int型は %d byte使っています¥n”, sizeof(int));

•別に型以外にも、とりあえずなんか入れてみると、なんか返ってくる
ex) printf(“⊂二二二(^ω^)二⊃ = %d byte“,
sizeof(“⊂二二二(^ω^)二⊃”));

•あと配列に対して使ってみるのも便利かも

第一章　ポインタの前に

表現範囲

•リミッターを超えてもパソコンが壊れることはない
•double型は内部で指数で保持されている
•でも、1.000000000001とかっていう数字は苦手

•RPGなんかで、チート使って、攻撃力あげすぎた状態で攻撃すると、何故か敵が回復したりするバグはここが原因。

第一章　ポインタの前に

章末問題

多分、簡単。

補足:
巫女ぐにょ•りぬくすというのはLinuxのディストリビューションの一つ。いつのまにかUbuntuベースになってた。起動画面が巫女さん。起動しても巫女さん。PCクラスタが世界一簡単に構築できるらしい。
CDブートできるので、暇な人はお試しを。

第二章　ポインタ

間接演算子

•遠く彼方にあるものを指すもの。
•値の実態は保持していないが、指しているものの値を変えることができる。

•ちなみに何も指していないポインタは、ものすごく危険なので、せめて*p = NULL;とでもしておきましょう
•いわゆる「ぬるぽ」です。

第二章　ポインタ

ポインタと配列

•配列の正体は実はポインタだった！！
a[i] == *(&a[0] + i)　←きっとこれテストでる

•もしかしたら、ポインタの正体は配列なのかもしれない！！

•というか、上の例、&つけたり、アスタリスクつけたり、わかるづらいな

•さらに新事実、&a[0]の正体はaだった！！
&a[0] == a　←きっとこれテストでる

第二章　ポインタ

関数と配列

•全部引数として渡すの面倒くせぇなぁ……
•ポインタって連番だから、最初に位置と、個数さえわかればよくね？
•うはっwwwww、俺天才wwwwwwww

```
int max( int *array, int num){
        int i, max = 0;
        for( i = 0; i < num ; i++){
                if( max < array[i]) max = array[i];
        }
        return max;
}
```

第二章　ポインタ

構造体

・構造体って、いろんな変数をまとめられるよ！！

```
struct point{
        int x;
        int y;
};
```

・最後のセミコロンは忘れやすいです。
・C++ではクラスっていう違うものに化けるよ

第二章　ポインタ

構造体へのポインタ

```
void swap( struct point *date){
        int buf;
        buf = (*data).a;
        (*data).a = (*data).b;
        (*data).b = buf;
}
```

アスタリスク＋括弧＋ドットを使うのは面倒じゃね？

第二章　ポインタ

構造体へのポインタ

```
void swap( struct point *data){
        int buf;
        buf = data->a;
        data->a = data->b;
        data->b = buf;
}
```

・「->」をアロー演算子と呼ぶ。
・構造体へのポインタを使うときはこれをまず使う
・C++だとクラスをnewして使うことが多々あります

第二章　ポインタ

章末問題

がんばれ！！

補足:
構造体を戻り値の型とする場合。

```
struct point swap( struct point data){
        int buf;
        buf = data.a;
        data.a = data.b;
        data.b = buf;
        return data;
}
```

第三章　文字列

文字列と文字

・文字　→　一文字だけ
ex）a, b, 1, ¥n, ¥t
・文字列　→　一文字以上
ex）wish you happiness, えれけん！, ⊂ニニニ(^ω^)ニつ

・文字列は終端に「¥0」ヌル文字が入る。

第三章　文字列

char型配列

さぁ、charに文字を代入してみよう！！

```
int main( void){
        char str[256];

        str[0] = 'w';
        str[1] = 'i';
        str[2] = 's';
        str[3] = 'h';
        str[4] = '¥0';
}
```

・すげー面倒じゃね？

第三章　文字列

文字列に関する関数

•ならば関数を使おう
•<string.h>のなかに文字列に関する関数が入っているよ！！
•主に使うのは
strcmp関数:文字列が同一か比較する
strcpy関数:文字列をコピーする
strcat関数:文字列をもう片方につなげる
strlen関数:文字列の長さを得る

第三章　文字列

char型ポインタ

•配列の長さって指定するの面倒じゃね？
•ならばポインタ使えばよくね？

```
int main( void){
        char *str;
        str = “wish you happiness“;

        puts( str);
        return 0;
}
```