

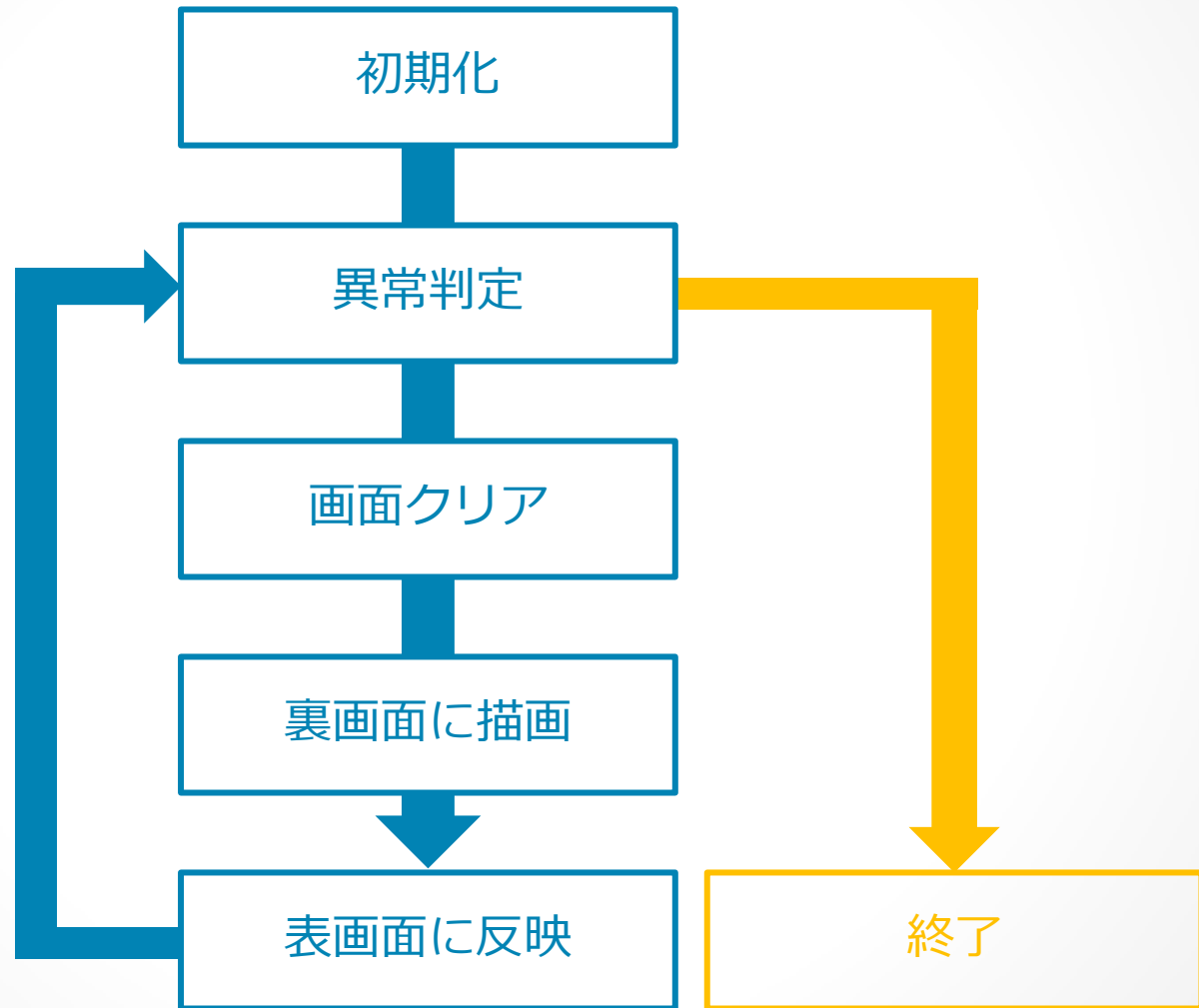
C++ゲームプログラミング講習 上級 第3回

ポインタ

復習

```
1  #include "DxLib.h"
2
3  int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
4                      LPSTR lpCmdLine, int nCmdShow ){
5
6      ChangeWindowMode(TRUE);
7      SetDrawScreen(DX_SCREEN_BACK);
8      DxLib_Init();
9
10     int handle = LoadGraph("Resource/picture.png");
11
12     int x = 0;
13
14     while(true) {
15         if(ProcessMessage() == -1) break;
16         if(CheckHitKey(KEY_INPUT_ESCAPE) == 1) break;
17
18         ClearDrawScreen();
19
20         DrawGraph(x, 30, handle, TRUE);
21         x += 1;
22
23         ScreenFlip();
24     }
25
26     return 0;
27
28 }
```

復習



敵のクラス化

- 前回、3種類の敵を作った
- 変数が多くなったり、メインループがごちゃごちゃになった
- 敵を**クラス化**しよう！

クラスの前に・・・

- C/C++でとても大事な**ポインタ**

(プログラミング初心者が挫折するところ。Javaでは使(わ)ない|え(な)ない)でも良いようになっている)

第1回、第2回はDxLibの使い方について講習したが、
今回はDxLibから離れて**C++の構文**を講習する

ポインタとは

- ・ 変数のアドレスを格納できる変数

ポインタとは

- ・ 変数のアドレスを格納できる変数
- ・ 変数のアドレスを格納できる変数

ポインタとは

- ・ 変数のアドレスを格納できる変数
- ・ 変数のアドレスを格納できる変数
- ・ 変数のアドレスを格納できる変数

ポインタとは

・ 変数のアドレスを格納できる変数

アドレスを記録する変数のデータ型 *ポインタ名;

```
int a = 123;
int *p;      // ポインタ
p = &a;      // aのアドレスを代入

printf("%d¥n", a) // aの値
printf("%d¥n", &a) // aのアドレス
printf("%d¥n", p); // pの値
printf("%d¥n", *p);
// pの値(アドレス)のところにある値
```

メモリ

アドレス	値
0000	
...	
1000	123
...	
1100	

p 1000

デバッグ用コンソールの準備

```
25 //デバッグ用コンソール
26 AllocConsole();
27 freopen("CONOUT$", "w", stdout);
28 freopen("CONIN$", "r", stdin);
29
30 ChangeWindowMode(TRUE);
31 SetDrawScreen(DX_SCREEN_BACK);
32 DxLib Init();
```

標準入力、標準出力をコンソールに割り当てている。
覚えなくて良い。

これはWindows用の関数なので、DxLib以外でも使用できる

標準出力

- printfを使用。(JavaのSystem.out.printfに相当)

```
34 int a = 100;
35 int *p = &a;
36 printf("%d\n", a); // aの値
37 printf("%d\n", &a); // aのアドレス
38 printf("%d\n", p); // pの値
39 printf("%d\n", *p); // pの値のところにある値
40
```

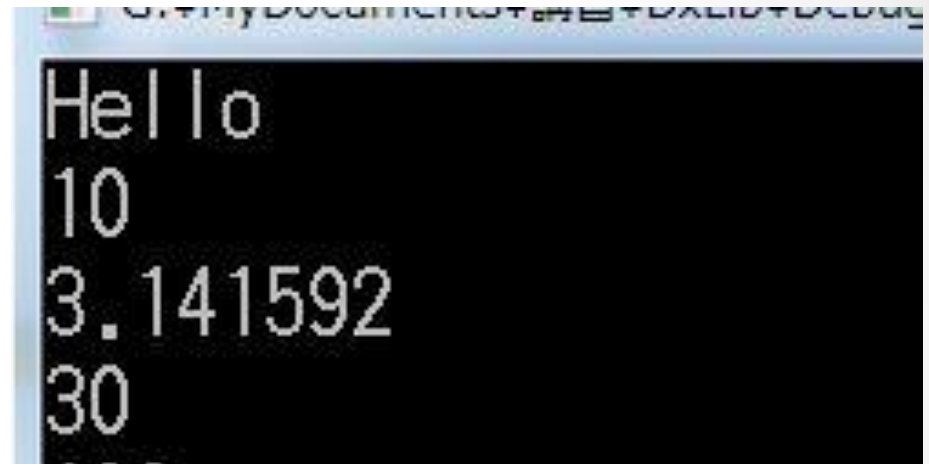
関数

```
22 void func() {  
23     printf("Hello\n");  
24 }  
25  
26 void func(int a) {  
27     printf("%d\n", a);  
28 }  
29  
30 float pi() {  
31     return 3.141592;  
32 }  
33  
34 int add(int a, int b) {  
35     return a + b;  
36 }  
37
```

```
49  
50 func();  
51 func(10);  
52 printf("%f\n", pi());  
53 printf("%d\n", add(10, 20));  
54
```

関数

```
22 void func() {  
23     printf("Hello\n");  
24 }  
25  
26 void func(int a) {  
27     printf("%d\n", a);  
28 }  
29  
30 float pi() {  
31     return 3.141592;  
32 }  
33  
34 int add(int a, int b) {  
35     return a + b;  
36 }  
37
```



```
Hello  
10  
3.141592  
30
```

```
49  
50 func();  
51 func(10);  
52 printf("%f\n", pi());  
53 printf("%d\n", add(10, 20));  
54
```

関数

```
22 void func1(int a) {  
23     a = 10;  
24 }  
25  
26 void func2(int *a) {  
27     *a = 10;  
28 }  
  
42     int a = 5;  
43     func1(a);  
44     printf("%d\n", a);  
45     func2(&a);  
46     printf("%d\n", a);  
47
```

関数

```
22 void func1(int a) {  
23     a = 10;  
24 }  
25  
26 void func2(int *a) {  
27     *a = 10;  
28 }
```

```
42     int a = 5;  
43     func1(a);  
44     printf("%d\n", a);  
45     func2(&a);  
46     printf("%d\n", a);  
47
```



5
10

注意

- 関数は前に書かなければならない(エラーになる)

```
22
23 int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,
24 LPSTR lpCmdLine, int nCmdShow ){
25
26     //デバッグ用コンソール
27     AllocConsole();
28     freopen("CONOUT$", "w", stdout);
29     freopen("CONIN$", "r", stdin);
30
31     ChangeWindowMode(TRUE);
32     SetDrawScreen(DX_SCREEN_BACK);
33     DxLib_Init();
34
35     func(10, 20);
36
37     while(true) {
38         if(ProcessMessage() == -1) break;
39         if(CheckHitKey(KEY_INPUT_ESCAPE) == 1) break;
40
41         ClearDrawScreen();
42
43
44         ScreenFlip();
45     }
46
47     return 0;
48 }
49
50 void func(int a, int b) {
51     printf("%d\n", a + b);
52 }
```

注意

- 関数は前に書かなければならない(エラーになる)

```
22 void func(int a, int b) {  
23     printf("%d\n", a + b);  
24 }  
25 |  
26 int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,  
27 LPSTR lpCmdLine, int nCmdShow ){  
28     //デバッグ用コンソール  
29     AllocConsole();  
30     freopen("CONOUT$", "w", stdout);  
31     freopen("CONIN$", "r", stdin);  
32  
33     ChangeWindowMode(TRUE);  
34     SetDrawScreen(DX_SCREEN_BACK);  
35     DxLib_Init();  
36  
37     func(10, 20);  
38  
39     while(true) {  
40         if(ProcessMessage() == -1) break;  
41         if(CheckHitKey(KEY_INPUT_ESCAPE) == 1) break;  
42  
43         ClearDrawScreen();  
44  
45         ScreenFlip();  
46     }  
47  
48     return 0;  
49 }  
50  
51 }
```

注意

- もしくは関数の定義だけ先に書く

```
22 void func(int a, int b);  
23  
24 int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,  
25 LPSTR lpCmdLine, int nCmdShow ){  
26  
27     //デバッグ用コンソール  
28     AllocConsole();  
29     freopen("CONOUT$", "w", stdout);  
30     freopen("CONIN$", "r", stdin);  
31  
32     ChangeWindowMode(TRUE);  
33     SetDrawScreen(DX_SCREEN_BACK);  
34     DxDLib_Init();  
35  
36     func(10, 20);  
37  
38     while(true) {  
39         if(ProcessMessage() == -1) break;  
40         if(CheckHitKey(KEY_INPUT_ESCAPE) == 1) break;  
41  
42         ClearDrawScreen();  
43  
44  
45         ScreenFlip();  
46     }  
47  
48     return 0;  
49 }  
50  
51 void func(int a, int b) {  
52     printf("%d\n", a + b);  
53 }
```

注意

- もしくは関数の定義だけ先に書く

```
22 void func(int a, int b);  
23  
24 int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance,  
25 LPSTR lpCmdLine, int nCmdShow ){  
26  
27     //デバッグ用コンソール  
28     AllocConsole();  
29     freopen("CONOUT$", "w", stdout);  
30     freopen("CONIN$", "r", stdin);  
31  
32     ChangeWindowMode(TRUE);  
33     SetDrawScreen(DX_SCREEN_BACK);  
34     DxDLib_Init();  
35  
36     func(10, 20);  
37  
38     while(true) {  
39         if(ProcessMessage() == -1) break;  
40         if(CheckHitKey(KEY_INPUT_ESCAPE) == 1) break;  
41  
42         ClearDrawScreen();  
43  
44  
45         ScreenFlip();  
46     }  
47  
48     return 0;  
49 }  
50  
51 void func(int a, int b) {  
52     printf("%d\n", a + b);  
53 }
```

ポインタを身につけよう！

```
35 // xに直接代入せずに 12を出力する|
36 int x;
37 int *p;
38
39 printf("%d\n", x);
40
```

ポインタを身につけよう！

```
35 // xに直接代入せずに 12を出力する
36 int x;
37 int *p;
38
39 p = &x; // pにxの番地を代入
40 *p = 12; // p番地の値を12にする
41
42 printf("%d\n", x); // 12が出力
43
```

ポインタを身につけよう！

```
22 // aとbの中身を入れ替える関数
23 void swap(int *a, int *b) {
24
25 }
26
```

ポインタを身につけよう！

```
22 // aとbの中身を入れ替える関数
23 void swap(int *a, int *b) {
24     int c;
25     c = *a; // cにa番地の値を代入
26     *a = *b; // a番地にb番地の値を代入
27     *b = c; // b番地にcを代入
28 }
29
```

```
43 int a = 10, b = 5;
44 swap(&a, &b);
45 printf("%d %d\n", a, b);
46
```


まとめ

- ポインタとは、
変数のアドレスを格納できる変数
- 関数の引数をポインタにしたら
参照渡しのようになる
- 関数は先に定義するか、
後の方に定義するなら先に宣言を書く

次回

- 配列の静的確保と動的確保について(要:ポインタ)
- new演算子とdelete演算子
- クラスの定義
- クラスの静的確保と動的確保(要:ポインタ)
- (もしかしたら配列で終わるかも)