第３回ａ　ファイル分割

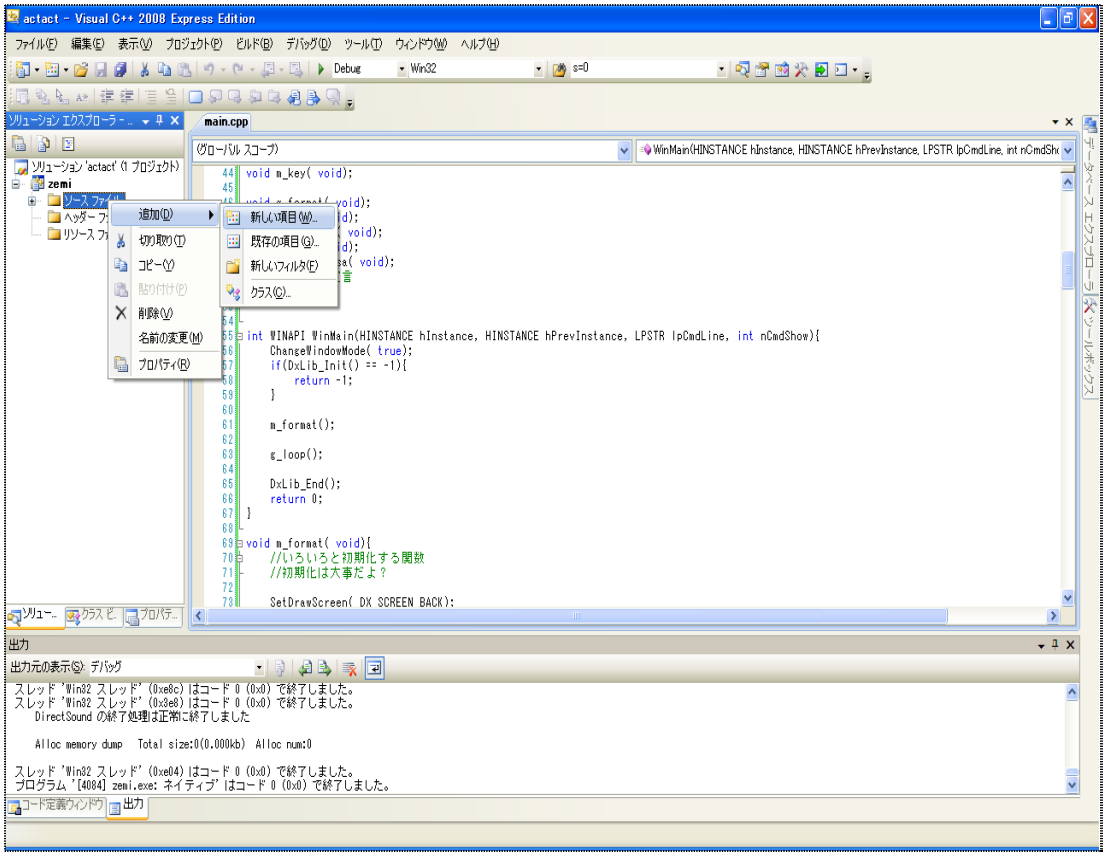
ゼミ A で習った extern を使いこれまでのソースを分割したいと思います。

extern の使い方はゼミ A の資料を見てください。

ファイル作成

Visual studio の左にあるソリューションエクスプローラーのソースファイルの所で右クリック→追加→新しい項目を押してファイル名の所に game と入力し”追加”を押してください。ソースファイルの中に game.cpp が追加されているはずです。確認してください。

次にソリューションエクスプローラーのヘッダーファイルの所でも同じことをして今度はファイル名を header と入力してください。そうすると今度はヘッダーファイルの中に header.h が追加されているはずです。



ヘッダーファイルとは、拡張子が「.h」のファイルのことを指します。このファイルをインクルードすると、ヘッダーファイルに書かれたプロトタイプ宣言や構造体、定数の定義などをまとめて利用できるようになります。

ファイル分割

main.cpp の中の以下の文を header.h にコピーして struct mono{}を main.cpp から消してください。

```
#include "DxLib.h"

void m_format( void);
int m_escape( void);
void m_key( void);

void g_loop( void);

int time;

int key_up;
int key_down;
int key_left;
int key_right;
int key_a;
int key_b;

struct mono{
	int haba;
	int takasa;

	int x;
	int y;
	int lr;
	int sokudo_x;
	int sokudo_y;
	int kasoku_x;
	int kasoku_y;
	int flag;

	int handle;
};
```

次に game.cpp の始めに#include “header.h”と書き、#define TIP_MAX_X 128 から#define JIKI_SOKUDO 8 までと、プロトタイプ宣言全部、int tip[TIP_MAX_X + 1][TIP_MAX_Y + 1];から struct mono jiki;、void g_loop(void){}、void g_format(void){}、void g_tip_draw(void){}、void g_jiki(void){}、void g_jiki_sousa(void){}の部分を全部 game.cpp にコピーし main.cpp から消します。

始めに書いた#include “header.h”の意味は、上で書かれているように header.h で書いた

構造体や宣言を利用できるようにする宣言です。

ちなみに、DxLib.h をインクルードした header.h をさらにインクルードした game.cpp は DxLib.h をインクルードしたことになるので game.cpp に改めて書く必要はありません。

現在 game.cpp はこのようになっていると思います。

```
#include "header.h"

#define TIP_MAX_X 128
#define TIP_MAX_Y 32
#define TIP_HABA 32
#define TIP_TAKASA 32

#define JIKI_HABA 32
#define JIKI_TAKASA 64

#define JIKI_SOKUDO 8

void g_format( void);
void g_loop( void);
void g_tip_draw( void);
void g_jiki( void);
void g_jiki_sousa( void);
//プロトタイプ宣言

int tip[TIP_MAX_X + 1][TIP_MAX_Y + 1];
int tip_handle;

struct mono jiki;

void g_loop( void){〜〜〜
}

void g_format( void){〜〜〜
}
void g_tip_draw( void){〜〜〜
}

void g_jiki( void){〜〜〜
}

void g_jiki_sousa( void){〜〜〜
}
```

main.cpp でも header.h で書いた宣言や構造体を使いたいのでインクルードできるように書きましょう。

ここでひとまずコンパイルをしてみましょう・・・。

おそらくコンパイルは通らないと思います。なぜなら、header.h で定義されているものが main.cpp で再定義されてしまっているからです。

この問題を解決するためには再定義を無くさなければなりません。そこで、片方の定義を宣言にするために extern を使います。

実際には、extern を両方に共通している宣言の片方の頭に付けてあげます。今回はインクルードする関係で header.h の以下の部分に付けます。

```
void m_format( void);
int m_escape( void);
void m_key( void);

void g_loop( void);

int time;

int key_up;
int key_down;
int key_left;
int key_right;
int key_a;
int key_b;
```

⇒

```
extern void m_format( void);
extern int m_escape( void);
extern void m_key( void);

extern void g_loop( void);

extern int time;

extern int key_up;
extern int key_down;
extern int key_left;
extern int key_right;
extern int key_a;
extern int key_b;
```

これでコンパイルは通り前回と同じ画面が出ると思います

次に第 3 回 b に移りたいと思います。