ソフトゼミＢ　第５回 ファイル分割

■　はじめに

これまでは一つのファイルに全てプログラムを書きました。しかし、実際自分でゲームを作っていくとどうしても行数が長くなります。そうすると、プログラムが見にくくなり、間違っている部分を探しにくくなってしまいます。

よって、今回は関係のあるプログラムをある程度の集まりごとに分割していきます。一つ一つのプログラムソースの行数が少なく、行っている動作の正誤も認識しやすいので、プログラムへの理解度が上がり、作業効率も向上します。

実際にゲームを作っていく時は初めから分割してプログラムを書いていくと良いと思います。

■　コンパイラの挙動とビルド

プログラムをする上ではいくつか気をつけなければならないことがあります。
まず、何故プロトタイプ宣言が必要なのでしょうか。コンパイラは基本的にソースファイル内に記述されているソースコードを上から順に解釈するため、まだ定義されていない関数が呼び出されると、「そのような関数は定義されていない」ので、コンパイルエラーとなります。ですから、その関数のプロトタイプ宣言をすることで、関数があとで定義されていることをコンパイラに伝えるのです。

上から順にコンパイルし終わると、コンパイラはオブジェクトファイル(拡張子が「.obj」なファイル)と呼ばれるファイルを生成します。これを基にして実行ファイルやライブラリが生成されます。

また、ソースファイルは一つのプロジェクトに複数書くことができます。プロジェクトに属する複数のソースファイルは全てコンパイラがオブジェクトファイルに変換し、リンカがそれらを結合させ、はじめて実行ファイルとなります。ヘッダは、#include によってプログラム内に挿入されるだけであり、#include しなければプログラムには何も影響を及ぼしません。

実は、いままでコンパイルと呼んできた作業は実は「**ビルド**」と呼ばれる作業であり、「ビルド」は「**プリプロセス**」(#include や#define などに従って、ソースコードを書き換える)→「**コンパイル**」(ソースコードを翻訳してオブジェクトファイルをつくる) →「**リンク**」(オブジェクトファイルを連結し、実行ファイルをつくる)という一連の作業から成って

いるのです。ややこしいかもしれませんが、とても重要なので頭に叩き入れましょう。

■　extern について

ゼミ A 第 6 回でも説明しましたが、変数にはそれぞれスコープ(有効範囲)があります。関数内部で宣言された変数はその関数内でしか使えないのと同様に、ソースコード内で宣言されたグローバル変数もそのソースコード内でしか扱うことができません。よって別のファイルでもその変数を使いたい場合は文頭に「extern」を付けた宣言を行います。(extern は記憶クラス指定子と呼ばれるものです。)

このように「extern」を変数宣言の先頭につけることによって、コンパイラに変数が他のソースファイル内で宣言されていることを知らせることができます。

■　ヘッダ(.h)について

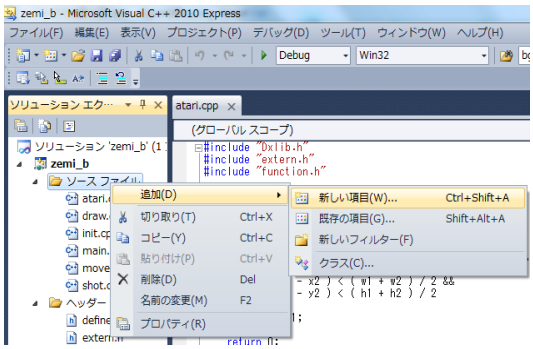
標準ライブラリ関数である printf や scanf などのプロトタイプ宣言は stdio.h というヘッダファイルの中で宣言されています。そのために「おまじない」である#include <stdio.h>という指令が必要なのです。

#include 指令によって、この一行が、そっくりそのまま<stdio.h>は、ヘッダ（header）と呼ばれ、それを#include 指令によって取り組むことをインクルードするといいます。

#include の行は、プリプロセスの段階で、そのヘッダの内容と入れ替わります。また、ヘッダは自作でき、自作のヘッダをインクルードするときは **<**ファイル名**>** ではなく **“**ファイル名**”**で記述します。例えば、標準ライブラリではない DX ライブラリをインクルードするときは、#include “DxLib.h”と書きました。

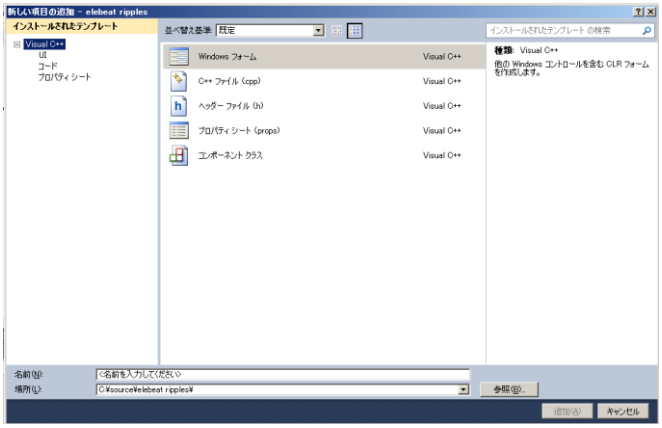
■　ファイル作成

新しくファイルを作るには Visual C++の左にある
ソリューションクスプローラーのソースファイルの所を右クリックします。
次に、「追加」をクリックします。



「新しい項目」を左クリックします。

すると、次の画面が出ます。



ここで、

「ソースファイル(.cpp)」を 3 つ、

- init.cpp
- draw.cpp
- shot.cpp

「ヘッダーファイル(.h)」を 5 つ、

- define.h
- extern.h
- function.h
- global.h
- struct.h

という名前でそれぞれ作成してください。

■　ファイル分割

main.cpp から抜き出して分割していきます。main.cpp のソースコードを参照しながらやって行きましょう。

➢ 移動

まず、

```
/**************#define ここから**************/
/*窓関連*/
//ウィンドウサイズ
#define WINDOW_WIDTH    640
                    中略
#define SHOT_BUTTON KEY_INPUT_Z
/**************#define ここまで**************/
```

main.cpp の上の部分をコピーして **define.h** に貼りつけて下さい。次に、

```
/*プロトタイプ宣言*/
void movePlayer( void );
中略
void moveEnemyShots( void );
```

上の部分を **function.h** に貼付け、コメント「/*プロトタイプ宣言*/」の直下に新たに

```
int init ( void );
void draw( void );
```

を足して下さい。

次に、

```
/*構造体の宣言*/
//ゲームデータ
struct Sgame{
                    中略
// 敵の弾
struct SEnemyShot{
   int x, y;
```

```
    int vx, vy;
    int life;
};
```

上の部分を **stcuct.h** に貼りつけて下さい。

次に、

```
/*グローバル変数の宣言*/
char keyState[ 256 ];
struct SGame game;
                    中略
struct SEnemyShot    enemyShot[ ENEMY_SHOT_MAX ];
int enemyShotNum;
```

上の部分を **global.h** に貼りつけて下さい。

次に、**init.cpp** に

```
int init ( void ){
   int i;
   // ウィンドウモードにする
   ChangeWindowMode( TRUE ) ;
   // 解像度とカラービット数を設定
   SetGraphMode( WINDOW_WIDTH, WINDOW_HEIGHT, 32 );
   // DX ライブラリの初期化に失敗すると、即終了
   if( DxLib_Init() == -1 ){ return -1; }
                        中略
   imgList.sidebar    = LoadGraph( "img/sidebar.png" );
   // 描画先の指定
   SetDrawScreen( DX_SCREEN_BACK );
   return 0;
}
```

と、打ち込みましょう。中略の部分は、main.cpp内のWinMain関数の先頭部分を参考に書いてみてください。初期化絡みの処理をinit関数で行うようにするための移動なのですが、**DXライブラリの初期化に失敗した時に戻り値-1 を返したい**ためにint型の関数となっております。元になったWinMain関数の先頭部分は消して、以下の関数呼び出しを追加してください。

```
if( init() == -1 ){ return -1; }
```

これで、init関数で異常があった時(DXライブラリ初期化失敗時)にプログラムを止めることができるようになりました。

次に、**draw.cppに**

```
void draw ( void ){
   int i;
   //画面に描かれたものを消去する
   ClearDrawScreen();
   //自機を描画
   DrawGraph( player.x, player.y, imgList.player, TRUE );
                        中略
```

```
        中略
  //表画面に反映
  ScreenFlip();
}
```

と、打ち込みましょう。こちらも、main.cppの描画部分を参考にしてみてください。この時点で、WinMain関数冒頭部にあった変数宣言「int i;」は不要になったので削除して構いません。

次に、**move.cpp**に

```
void movePlayer( void ){
  //プレイヤーがこのフレームで動くx座標の距離
  int vx = 0;
  //プレイヤーがこのフレームで動くy座標の距離
                        中略
      enemy[ i ].life = 0;
      enemyNum --;
      }
  }
}
```

の部分(movePlayer, setEnemy, moveEnemy各関数)を貼り付けて下さい。

次に、**shot.cppに**

```
void setPlayerShot( void ){
  int i;
  if( keyState[ SHOT_BUTTON ] && playerShotNum < PLAYER_SHOT_MAX ){
                        中略
  // 弾が完全に画面外に出たら、その弾は死ぬ。
     if( enemyShot[ i ].y > WINDOW_HEIGHT + BULLET_SIZE ){
     enemyShot[ i ].life = 0;
     enemyShotNum --;
  }
}
```

の部分(setPlayerShot, movePlayerShots, setEnemyShot, moveEnemyShots各関数)を貼り付けて下さい。

- 微調整

define.h , function.h

define.h , function.h内にコピーして貼ったらmain.cpp文内のコピーした文章を消してください。

stcuct.h

一番上に、**#include "define.h"**を足して下さい。終わったら、main.cpp文内のコピーした文章を消してください。

global.h

一番上に**#include "struct.h"**を足して下さい。終わったら、main.cpp文内のコピーした文章を消してください。

extern.h

一番上に**#include "struct.h"** と書き、下に

```
extern char keyState[ 256 ];
extern struct SGame game;
extern struct SImgList imgList;
extern struct SPlayer player;

extern struct SEnemy enemy[ ENEMY_MAX ];
extern int enemyNum;
extern int enemyTrigger;
extern struct SPlayerShot playerShot[ PLAYER_SHOT_MAX ];
extern int playerShotNum;
extern struct SEnemyShot    enemyShot[ ENEMY_SHOT_MAX ];
extern int enemyShotNum;
```

と打ち込みましょう。

shot.cpp , init.cpp , draw.cpp

一番上に

```
// DX ライブラリのヘッダをインクルード
#include "DxLib.h"

// 自作ヘッダのインクルード
#include "extern.h"
#include "function.h
```

を足して下さい。

#include "DxLib.h"

shot.cpp , init.cpp , draw.cpp には DX ライブラリでしか使われない関数を使っているので、これがないとエラーが起こります。

#include "extern.h"

#include "function.h

shot.cpp , init.cpp , draw.cpp には extern.h、function.h で宣言されたものが使われているので、これがないとエラーが起こります。

終わったら、main.cpp 文内のコピーした文章を消してください。

move.cpp

一番上に

```
// DX ライブラリのヘッダをインクルード
#include "DxLib.h"

// 自作ヘッダのインクルード
#include "extern.h"
#include "function.h
//数学的な部分があるので必要
#include <math.h>
```

を書き足して下さい。

#include "extern.h"

move.cpp には extern.h で宣言されたものが使われているので、これがないとエラーが起こります。

#include <math.h>

move.cppには数学的な関数（平方根を求める関数sqrt()）が使われているので、これがないとエラーが起こります。

また、math.hはあらかじめ用意されているインクルードファイルなので、""（ダブルクォーテーション形式）ではなく、<>（角括弧形式）で定義されます。

（自作のヘッダをインクルードするときは <> ではなく “”で記述）

終わったら、main 文内のコピーした文章を消してください。

main.cpp

最後に、main.cppの、#include "DxLib.h"の下に、

```
// 自作ヘッダのインクルード
#include "global.h"

#include "function.h"
```

を書き足してください。

これを足さなければそれぞれのファイルとのつながりがないため、結果的に定義されていない識別子ばかりになり、実行させるとたくさんのエラーが発生します。

逆に言うと、この二つのヘッダファイルをインクルードするだけで識別子はすべて定義されるのです。

#include "global.h”

global.h はすべてのグローバル変数がまとめられており、これをインクルードすることですべてのグローバル変数を定義することができます。これができるのは、extern.h が存在しているからです（「extern」を変数宣言の先頭につけることによって、コンパイラに変数が他のソースファイル内で宣言されていることを知らせているため）。

#include "function.h”

main.cpp には function.h で宣言されたものが使われているので、これがないとエラーが起こります。

終わったらビルドしてみてください。ちゃんと動作しますか？もしビルド エラーが出る場合にはエラーメッセージを頼りに、もう一度過不足が無いか確かめてみてください。