ソフトゼミ B 2015 第 5 回

ファイル分割

今まではすべての処理を main.cpp に記述してきましたが、今回はそれぞれの役割ごとに分割させていく「ファイル分割」を行っていきます。

➢ ファイル分割

ファイル分割をすることで、今までは長くて見えづらかったソースファイルが役割ごとにまとまっているうえに分割されたことにより短くなっているので非常に見やすいものになります。

例えば、create_enemybullet 関数にエラーがあったとします。今まではソースファイルのはじめからスクロールしていってその関数を探さなければならず、手間のかかる作業でした。しかし分割して、例えば create_となっている、敵、弾などをつくる関数のみを集めた create.cpp というファイルに上記の関数も入れておけば、どこにあるかはわかりやすくなります。書類等をフォルダにまとめておく、といった作業に酷似しているといえます。

また、judge に関する関数をつくる人、create に関する関数をつくる人、move に関する関数をつくる人・・・といったように役割分担をする際にも便利です。

➢ ヘッダーファイルを用いたファイル分割

ヘッダーファイルを使った分割をします。ヘッダーファイルに define や struct といった全てのソースコードに共通する情報を書き込んだのち、それをソースコードに足し合わせることができます。

#include <stdio.h>や、#include<math.h>はコンパイラの中で既に定義されている、すなわち行うべき処理がすでに記述されているヘッダーファイルなので<>で囲みます。これらを標準ライブラリと言います。

しかしこれから自分たちで作成する define.h や struct.h といったヘッダーファイルはコン　パイルの中で定義されていません。これらは<>で囲まずに""で囲みます。上記のヘッダーファイルは#include “define.h”や#include “struct.h”といったように記述します。

➢ extern 宣言

分割してそれぞれをコンパイルした場合、ローカル変数はもちろんですが、グローバル変数も共有されていません。グローバル変数も、そのファイル内でしか通用しないからです。

たとえば、create.cpp でグローバル宣言した変数 x は、そのファイル内のどの関数でも使用することができますが、ほかのファイル、例えば judge.cpp では使用することができません。つまり playerbullet などを共通の変数として扱うことができないのです。かといってそれぞれのファイルで playerbullet を宣言したとしても、create.cpp 内の playerbullet と judge.cpp の playerbullet は名前こそ同じであるものの、全く別物の変数として捉えられてしまいます。

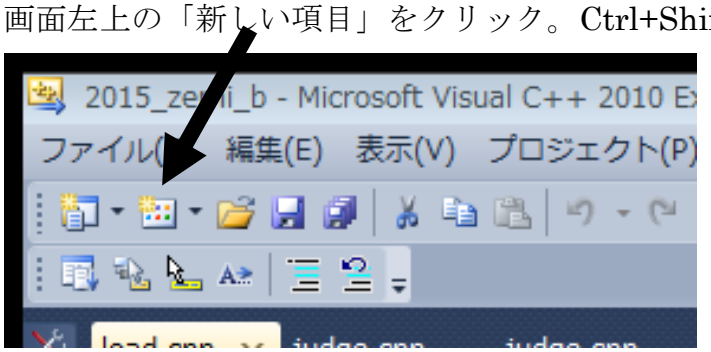
一方のファイルで変数を宣言し、ほかのファイルにその変数が存在することを教える作業をします。そこで使用するのがこの extern 宣言です。変数を宣言する前に extern を記述します。あくまでも宣言なので、新しく変数を定義するわけではないので注意してください。

```
extern 型 変数名;      //宣言のみ
```

➢ 実際に分割してみる

実際に今回のプログラムを分割してみましょう。

- 画面左上の「新しい項目」をクリック。Ctrl+Shift+A でもできます。



- ソースファイルをつくる場合はC++を、ヘッダーファイルをつくる場合はヘッダーファイルを選択して、名前を入力して追加をクリック。

  ソースファイル
  - initialize.cpp
  - load.cpp
  - move.cpp
  - create.cpp
  - judge.cpp
  - draw.cpp
  - update.cpp

  ヘッダーファイル
  - define.h
  - struct.h
  - grobal.h
  - extern.h
  - function.h

- それぞれのファイルを記述していきます。該当する範囲を「切り取り」→「貼り付け」をしていくと楽です。関数の中身は変えないので関数名だけ書いておきます。

➢ ヘッダーファイル

- define.h

  マクロを格納するファイル

```
#define WIDTH 800
#define HEIGHT 600
#define ENEMY 100
#define ITEM 30
#define BULLET 200
```

- struct.h

  構造体の宣言を格納するファイル

```
#include “define.h”
struct S_player
{
 //中身
};
struct S_enemy
{
 //中身
};
struct S_bullet
{
 //中身
};
struct S_item
{
 //中身
};
struct S_background
{
 //中身
};
struct S_image
{
 //中身
```

```
};
enum e_flag
{
 //          中身
};
```

- grobal.h

  グローバル変数の定義を格納するファイル

```
#include "struct.h"
#include "DxLib.h"
S_player player;                              //自機の struct をインスタンス化
S_enemy enemy[ENEMY];
S_bullet playerbullet[BULLET];
S_bullet enemybullet[ENEMY][BULLET];
S_item item[ITEM];
S_background background;
S_image imglist;
e_flag flag;
char key[256];
int black = GetColor(0,0,0);
```

- extern.h

  グローバル変数の宣言を格納するファイル

```
#include "struct.h"
extern S_player player;
extern S_enemy enemy[ENEMY];                  //敵の struct を
extern S_bullet playerbullet[BULLET];
extern S_bullet enemybullet[ENEMY][BULLET];
extern S_item item[ITEM];
extern S_background background;
extern S_image imglist;
extern e_flag flag;
extern char key[256];
extern int black;
```

- function.h

  関数のプロトタイプ宣言を格納するファイル

```
void load();
int initialize();
void move_player();
void move_enemy();
void move_playerbullet();
void move_enemybullet();
void move_item();
void move_background();
void create_enemy();
void create_playerbullet();
void create_enemybullet();
void create_item(int x,int y,int size_x,int size_y);
void judge_player_to_enemy();
void judge_playerbullet_to_enemy();
void judge_enemybullet_to_player();
void judge_player_to_item();
void update();
void draw();
```

- move.cpp

  移動に関係する関数を格納するファイル

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"

void move_player()
{
 //中身
}
void move_enemy()
{
 //中身
}
void move_playerbullet()
{
 //中身
}
void move_enemybullet()
{
 //中身
}
void move_item()
{
 //中身
}
void move_background()
{
 //中身
}
```

- create.cpp

  いろいろと生成する関数を格納するファイル

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"
```

```
#include <math.h>


void create_enemy()//敵を作る関数
{
 //中身
}
void create_playerbullet()
{
 //中身
}
void create_enemybullet()
{
 //中身
}
void create_item(int x,int y,int size_x,int size_y)
{
 //中身
}
```

- judge.cpp

  当たり判定に関する関数を格納するファイル

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"
void judge_player_to_enemy()
{
 //中身
}
void judge_playerbullet_to_enemy()
{
 //中身
}
void judge_enemybullet_to_player()
{
 //中身
}
```

```
void judge_player_to_item()
{
 //中身
}
```

- Draw.cpp

  描画に関する関数を格納するファイル

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"
void draw()
{
 //中身
}
```

- Update.cpp

  数値的な処理を行う関数を格納するファイル

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"

void update()
{
 //中身
}
```

以上です。お疲れ様でした。デバッグしてきちんと動くかどうかを確認しましょう。