ソフトゼミ B　第 5 回

ファイル分割

今までは 1 つのファイルにすべてのソースを書いていました。それではゲームを作るときソースが長すぎて見づらいです。そこで、ソースを見やすくしたり管理をよりやりやすくするためファイル分割を行いましょう。

◆　ファイル分割

ファイル分割とはソースを複数のファイルに分けて書くことです。変数の宣言や関数の役割などで分類しファイルを分割しておけば、どこにどの関数があるかわかりやすくなります。今あるソース内の judge_playerbullte_to_enemy 関数にバグがあることが分かったとします。これを直そうとしたとき 1 つのファイルのすべての関数から探し出さなければなりませんが、ファイル分割しておけばどこにあるかはわかりやすいです。さらに、ファイル分割は複数人でソースを組むときの役割分担に便利です。ファイルを分割する方法としては.c または.cpp ファイルを複数用意して分割コンパイルしてリンクする方法とヘッダーファイルを用意する方法があります。

◆　分割コンパイルとリンク

分割コンパイルの過程の説明をします。例として今回作っているゲームのソースを分解する場合を考えます。まず今までは main.cpp のみを使っていました。main.cpp の関数を,プログラム内の初期化をつかさどる関数を格納する initialize.cpp、画像のロードをする関数を格納する load.cpp、自機、敵、弾の移動をつかさどる関数を格納する move.cpp、敵と弾を生成する関数を格納する create.cpp、画面描画をつかさどる関数を格納する draw.cpp、当たり判定をつかさどる関数を格納する judge.cpp、数値的な処理をつかさどる uodate.cpp、以上 8 つのファイルに分割して分割コンパイルした場合を考えます。まず、上の 8 つのファイルを別々にコンパイルします。borland の場合 bcc32 の後に空白を置いて-c を入力してからファイル名入力します。するとコンパイルしたファイル名の拡張子が.obj になったものが出力されます。それらをまとめてコンパイルするとすべてのファイルのソースを網羅した実行ファイルが出力されます。borland の場合 bcc32 の後に空白を置いて-e を書いた後にそれぞれ空白を置いて.obj ファイル名（実は.c や.cpp でも可）を書きます。この作業をリンクといいます。※Visual C++のプロジェクトでは分割コンパイルとリンクを自動でやってくれるのでこんなめんどくさいことは考えなくて良いです。

◆ ヘッダーファイル

ヘッダーファイルを使ったファイル分割を説明します。これはあるファイルをコンパイルするときヘッダーファイル内の情報をソースファイルの情報に足し合わせることができます。足し合わせ方としてはソースファイルの先頭に#include と書いてヘッダー後ろにファイル名を書きます。(いつも書いている“stdio.h”、”math.h”などはコンパイラ内で定義されているヘッダーファイルです。またこれらを標準ライブラリといいます。) この時ファイル名を標準ライブラリは<>で囲うのに対して自作ヘッダーは””で囲みます。なので#include<math.h>に対して#include”DxLib.h”とソースには書きます。

◆ extern 宣言

分割コンパイルした場合それぞれのファイルで使用されている（ローカル変数は当たり前ですが）グローバル変数は共有されてません。たとえば main.cpp 内でグローバル変数で宣言された変数 x は main.cpp 内のグローバル変数であって、ほかのファイル、例えば move.cpp 内の関数では使えません(move.cpp をコンパイルするとエラー)。また、move.cpp 内で普通にグローバル変数 x を定義しなおしても不具合ができます（リンクをする時点で警告)。理由としてはグローバル変数を定義するときにメモリー領域を割り充てるのですが、先のようにやると x を２回定義したことにより同名の変数が２個できてしまうことにあります。(名前が一緒でも中身は違う)
どうすればいいかというと一方のファイルで変数を定義し、それ以外のファイルにそういう名前の変数があることを知らせる必要があります。(これを変数を宣言するといいます。)そこで変数を宣言だけする場合変数の形名の前に extern と書きます。(宣言なので定義ではありません。つまり extern では変数は作られません。)

```
extern 型 変数名; //宣言のみ
```

◆ 実際に分割をやってみる

ゼミＢで使ったサンプルプログラムをファイル分割してみましょう。Visual C++は分割コンパイルとリンクを自動でやってくれます。どのようにしてファイル分割をするのか以下に書いていきます。

- 左上の新しい項目をクリック（もしくは Ctrl+Shift+A 覚えとくと便利）

- ソースファイルを作る場合 C++ファイルを、ヘッダーファイルを作る場合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。作るファイルは以下の通りです。
  - 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に打ち込みます。（コピペすると楽です。）main.cpp も改変します。関数の中身は変えないので関数名だけ書いておきます。

◆ ヘッダーファイル変更

- define.h

  マクロを格納するファイル

```
#define WIDTH 640
#define HEIGHT 480
#define ENEMY 20
#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{
        int x,y;
        int vx,vy;
        int hp;
        int score;
        int size_x;
        int size_y;
};
struct S_enemy{
        int x,y;
        int vx,vy;
        int hp;
        int t;
        int size_x;
        int size_y;
};
struct S_bullet{
        double x,y;
        double vx,vy;
        int hp;
        int atk;
        int size_x;
        int size_y;
```

```
};
struct S_item{
        int x,y;
        int vx,vy;
        int hp;
        int size_x;
        int size_y;
};
struct S_background{
        int x,y;
        int vy;
        int size_x;
        int size_y;
};
struct S_image{
        int img_player;
        int img_enemy;
        int img_bullet;
        int img_item;
        int img_gameover;
        int img_title;
        int img_background;
};
enum e_flag{
        int item;
        int tokuten_tama;
        int haikei;
};
```

- 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;
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 white=GetColor(255,255,255);
```

- 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];
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 white;
```

- function.h

  関数のプロトタイプ宣言を格納するファイル

```
int initialize();
void load();
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_playerbullet_to_enemy();
void judge_enemybullet_to_player();
void judge_player_to_enemy();
void judge_player_to_item();
void update();
void draw();
```

◆ ソースファイル内容

- main.cpp

  ゲームの仕組みを格納するファイル

```
#include"DxLib.h"
#include"global.h"
#include"function.h"
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
nShowCmd ){
    ChangeWindowMode(TRUE);
    SetGraphMode( WIDTH, HEIGHT, 32 );
    if(DxLib_Init()!=0) return 0;
    SetDrawScreen( DX_SCREEN_BACK );
    load() ;
    while(ProcessMessage()==0){
        if(key[KEY_INPUT_ESCAPE]) break;
        update();
```

```
        draw();
    }
    DxLib_End();
    return 0;
}
```

- init.cpp

  プログラムの初期化関係の関数を格納するファイル

```
#include "extern.h"
#include "function.h"
//関数名だけ書くので中身はコピペでもしてください。
int initialize(){
…
}
```

- Load.cpp

  画像のロードの関数を格納するファイル

```
#include"extern.h"
#include"DxLib.h"
#include" function.h"
void load(){
…
}
```

- draw.cpp

  描画関係の関数を格納するファイル

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

- judge.cpp

  当たり判定に関係のある関数を格納するファイル

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

- move.cpp

  移動に関数を格納するファイル

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"
void move_player(){
…

}
void move_enemy(){
…
}
void move_playerbullet(){
…
}
void move_enemybullet(){
…
```

```
}
void move_item(){
…
}
void move_background(){
…
}
```

- update.cpp

  数値的な処理をする関数を格納するファイル

```
#include "DxLib.h"
#include "extern.h"
#include "function.h"
void update(){
…
}
```

- 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(){
…
}
void create_playerbullet(){
…
}
void create_enemybullet(){
…
}
void create_item(int x,int y,int size_x,int size_y){
…
}
```

以上です。お疲れ様でした。できたら Debug で動くことを確認しましょう。