

第3回a ファイル分割

これまでは、1つのファイルに全てプログラムを書いていましたが、本格的にゲームとして作っていく過程でどうしても行数が長くなり、プログラムが見にくくなったり、間違っている部分を探しにくくなります。従って、関係のあるプログラムをある程度のまとまりごとに分割しておくとな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。第3回aではそのファイル分割について学びたいと思います。
※Visual C++内でしばしば見られる語句の末尾等の長音符は積極的に省略していきます。

○コンパイラの挙動

今までにプロトタイプ宣言等が登場したように、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。それを覚えるためには、コンパイラの挙動を覚えることが近道です。

まず、何故プロトタイプ宣言が必要なのかを思い出してみましょう。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(Visual C++ 2010では「ソース ファイル」内に表示される)を上から順に解釈し、プログラムの塊である、拡張子が「.obj」になっているファイルを生成します。上から順に解釈しているため、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」ため、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を宣言することで、関数があると定義されていることをコンパイラに伝えるのです。

さて、「.obj」ファイルが生成された時点では、まだWindowsはプログラムを実行することができません。実はソースファイルは一つのプロジェクトにつき、複数書くことができます。コンパイラが、プロジェクトに属するソースファイルを全て「.obj」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。つまり、「ヘッダファイル」はコンパイルしません。リンカという言葉が初めて登場しました。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「コンパイル」と「リンク」の2つの作業から成っている(実際にはその前にプリプロセスと呼ばれる作業がありますが、今回は省略します)のです。ややこしいかもしれませんが、激烈に重要なので頭に叩き込んでください。

一般にソースファイルには関数の定義を記述し、ヘッダファイルに構造体の定義やプロトタイプ宣言、defineやグローバル変数や、後述するexternを記述します。

最後になりますが、コンパイラはソースファイルを一つコンパイルする毎に、それまで覚えていたヘッダファイルの情報を全て忘れ去りますので、includeはそれぞれのソースファイルに考えながら記述してやる必要があります。

○extern

ゼミAの第六回『関数について』でやったと思いますが、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事ができません。よって別のファイルでもその変数を使いたい場合は「extern」という記憶クラス指定子をつけ、コンパイラに別のソースファイル内でグローバル変数として宣言されていることを知らせてやらねばなりません。

「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出来ます。関数のプロトタイプ宣言のようなものです。

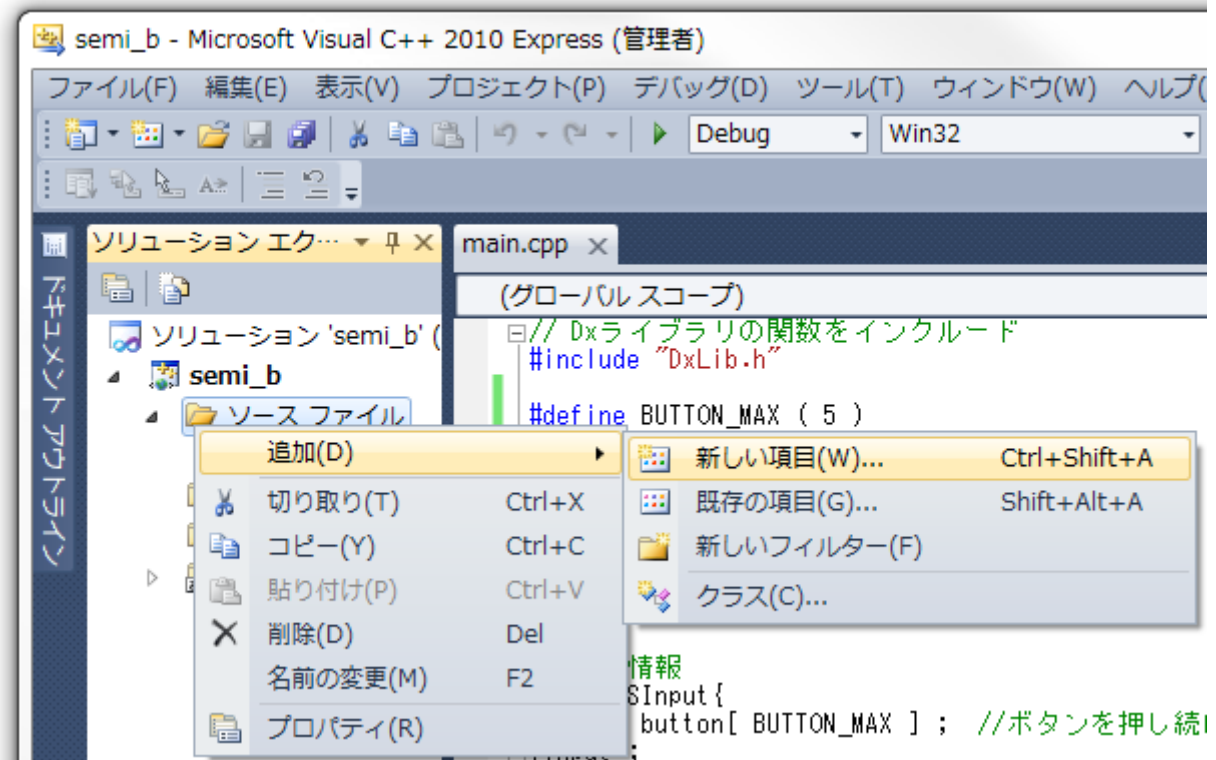
```
//source01.c
int num = 0;
int main(void) {
    function();
    printf(" num = %d ", num);
    return(0);
}
```

```
//source02.c
extern int num;
void function(void){
    num =5;
}
```

上に示した例は source01.c と source02.c の二つのファイルがあり、source01.c でグローバル変数として宣言した int num を source02.c でも使えるようにしたプログラムです。この extern は実際のゲーム制作時にもかなりお世話になるものですが、同名の変数が複数存在した場合などに意図しない結果となる事があるので気をつけて使うようにしましょう。なお、同じ変数を複数回同じソースファイル内で extern した場合や、グローバル変数としてどこにも宣言されていないのに extern した場合、また、プロトタイプ宣言とは異なりグローバル変数が宣言されているソースファイル内のどこかで extern した場合はエラーとなりますので、注意してください。

○ファイル作成

今回は、今まで main.c で宣言していた構造体の定義を struct.h に、define による定数定義を define.h に、グローバル変数を global.h に、その extern を extern.h に、プロトタイプ宣言を function.h にまとめていきます。また、関係する物体ごとに、cpp ファイルを作成し、そこに関数をまとめていきます。Visual C++の左にあるソリューションエクスプローラのソースファイルの所で右クリック->追加->新しい項目を左クリックし、C++ファイルを選択後、ファイル名の所に player と入力し「追加」を押してください。ソースファイルの中に player.cpp が追加されているはずなので、それを確認してください。次にヘッダファイルの所でも同様に操作を行います。今度はヘッダファイルを選択後、ファイル名を extern と入力し「追加」を押してください。ヘッダファイルの中に extern.h が追加されているはずです。ヘッダファイルというのは拡張子が「.h」のファイルの事です。このファイルを include することで、そこに書かれているプロトタイプ宣言や構造体、define による定数の定義などをソースコード内に挿入出来ます。同様に後 4 つ、define.h、struct.h、global.h、function.h というヘッダファイルも作ってください。次のページの画像は作業中のものです。



○ファイル分割

まず main.cpp 中の記号定数の宣言部を define.h にコピーし、main.cpp から消し、消した部分で define.h をインクルードしてください。この時点でのソースファイルはフォルダ「a」内のファイル郡を参照してください。

次に、構造体の定義の末尾でグローバル変数として宣言していた構造体変数の宣言を、global.h に記述し、構造体の定義の末尾での宣言を消去、構造体の定義部の後の部分で、global.h をインクルードしてください。以下は global.h と main.cpp の内容です。この時点でのソースコードはフォルダ「b」を参照してください。

```

/*****
    global.h      グローバル変数の宣言
*****/
// 構造体変数の宣言
struct SInput input;      // 入力情報
struct SPlayer player;    // プレイヤー
struct SImg img;          // 画像

前略
#define.h

                                中略

struct SInput {
                                中略
};
                                中略

#include "global.h"
                                以下略

```

次に struct.h に現在までに作った構造体(SInput、SPlayer、SImg)を全てコピーし、main.cpp から消してください。また、消した部分で struct.h をインクルードしてください。この時点でのソースコードはフォルダ「c」を参照してください。

```

/*****
    struct.h      構造体定義
*****/
// 構造体
                                以下略

```

次に main.cpp に記述が残っているグローバル変数を global.h にコピーし、main.cpp から消します。

```

/*****
    global.h      グローバル変数の宣言
*****/
// グローバル変数
char KeyBuf[ 256 ];        // キーボードの入力を保存する変数

// 構造体変数の宣言
                                以下略

```

MovePlayer 関数と DrawPlayer 関数を player.cpp にコピーし main.cpp から消し、player.cpp の先頭で DxDlib.h を include 後、その直後の部分で、define.h、struct.h、global.h を include してください。最後に、WinMain 関数を除く全ての関数のプロトタイプ宣言を function.h 内に記述し、function.h の include を main.cpp と player.cpp の global.h の include の記述部分の後に追記してください。

ここまでやったら一回コンパイルしてみましょう。下記のエラーメッセージが出てくるはずですが。

```
1>player.obj : error LNK2005: "struct SPlayer player" (?player@@@3USPlayer@@A) は既に
```

main.obj で定義されています。

1>player.obj : error LNK2005: "struct SImg img" (?img@@3USImg@@A) は既に main.obj で定義されています。

1>player.obj : error LNK2005: "char * KeyBuf" (?KeyBuf@@3PADA) は既に main.obj で定義されています。

1>player.obj : error LNK2005: "struct SInput input" (?input@@3USInput@@A) は既に main.obj で定義されています。

1>player.obj : error LNK2005: "struct SPlayer player" (?player@@3USPlayer@@A) は既に main.obj で定義されています。

1>player.obj : error LNK2005: "struct SImg img" (?img@@3USImg@@A) は既に main.obj で定義されています。

1>player.obj : error LNK2005: "char * KeyBuf" (?KeyBuf@@3PADA) は既に main.obj で定義されています。

1>player.obj : error LNK2005: "struct SInput input" (?input@@3USInput@@A) は既に main.obj で定義されています。

1>C:\Users\Agate\Desktop\test_dx\Debug\test_dx.exe : fatal error LNK1169: 1 つ以上の複数回定義されているシンボルが見つかりました。

これはグローバル変数を宣言しているヘッダファイルを複数のソースファイルで include しているためにおこる現象です。この現象を解消するため、冒頭で解説した extern を使用します。global.h の全文を extern.h にコピーし、下記のようにグローバル変数の宣言の先頭に extern をつけてください。

```
#include "struct.h"
```

```
extern char KeyBuf[ 256 ] ;
```

```
extern struct SImg img ;
```

```
extern struct SInput input ;
```

```
extern struct SPlayer player ;
```

これだけではインクルードしていないので意味がありません。player.cpp の global.h の include を extern.h の include に置き換えて下さい。これでビルドに成功するはずです。以後、main.cpp 以外のソースファイルでは、global.h は include せず、extern.h を include します。この次点でのソースファイルはフォルダ「d」を参照してください。