オブジェクト指向とゲームプログラミング

● 基礎編 － 第５回　型変換、分岐 ●

型変換

変数や数値の型を別の型へ変換することを型変換といいます。この型変換の特性のために、予想し得ない計算結果をもたらすことがあります。
算術式を記述していくと、整数のデータと浮動小数点のデータが混ざってしまうことがあります。

1 + 1.2　　整数値 + 浮動小数点値　→　この結果は浮動小数点値になります。

iをint型の変数、jをdouble型の変数とすると、

i + j　　→　この結果は浮動小数点値(double型)になります。

異なる型どうしの演算の場合には、より大きな型へと変換されて計算されます。

```
1   + 1.2
↓
1.0 + 1.2   浮動小数点値 + 浮動小数点値
```

このように、型変換はプログラムの文脈の中で自動的に行われることがあり、これを暗黙の型変換と呼びます。暗黙の型変換のルールは、以下のとおりです。

①char, short, enumはいずれもintに昇格されます。intとして表現できない整数型はunsigned intとなります。

②①の後、式が混合式(２つの型が一致しない)である場合は、

int → unsigend int → long → unsigend long → float → double → long double

のような階層構造によって、下位の型のデータはより高位の型のデータに昇格し、演算の結果はその型になります。

計算に使用される型がすべてshort(またはchar)だとしても、これらは混合式ではありませんが、すべてintへと昇格します。
今、以下のように定義されているとします。

```
short    s;
long     l;
float    f;
double   d;
```

このとき、以下の演算はコメントに示した型変換が行われます。

```
s + 893;         // intへ昇格
s + 8930000;     // sはintへ昇格
s + l;           // sはlongへ昇格
s + f;           // sはfloatへ昇格
l + f;           // lはfloatへ昇格
f + 3.14f;       // 型変換なし
f + 3.14;        // fはdoubleへ昇格
f + d;           // fはdoubleへ昇格
d + 3.14;        // 型変換なし
d + 3.14f;       // 3.14fがdoubleへ昇格
```

ここで、整数値が浮動小数点値に変換される場合には注意が必要です。longは１０桁程度の精度を持っていますが、floatは７桁程度の精度であるからです。したがって、longの精度が失われることがあります。

型変換は、代入動作でも発生します。

```
double   d = 893;
```

この例では、整数値をdouble型の変数dに代入しています。この場合、893が型変換されて893.0×10^{0}になってdに代入されます。逆に、浮動小数点値を整数型の変数に代入すると、浮動小数点値が整数に変換されます。

```
int    i = 89.3;
```

この場合、高位の型から下位の型への変換になります。この場合は、89.3の小数点以下が切り捨てられ、89という整数値になります。

```
int      i;
double   d, e;

e = 89.3;
i = e;
d = i;       // d = i = e = 89.3 と同じ意味
```

この例では、まずeに89.3がそのまま格納されます。次に、iにeの値89.3を代入しようとしますが、iはint型のため暗黙の型変換が起き、小数以下が切り捨てられて89となり、iに代入されます。次にdにiの値89を代入するので、dには89.0が代入されることになります。

このように、違う型どうしの計算においては、常に暗黙の型変換を意識してプログラムすることが重要です。また、代入や次で紹介するキャスト演算子では、下位の型へと変換される場合があります。このとき、変換先の型で表現できない値が強制的に代入されてしまうことがあります。このような場合は情報が失われてしまいます。浮動小数点型が整数型に降格する場合は小数以下が切り捨てられますが、intやlongでしか表現できない値をshortやcharに変換したときの動作は、CPUによって異なります。

キャスト演算子

整数どうしの除算を行ったとき、商を小数で受け取りたい場合があります。

```
int      i = 89, j = 3;
double   d = i / j;
```

このとき、dにはi / jの29.0が代入されます。なぜなら、int型どうしの演算は、int型にしかならないからです。29.666666666666666666666666666667を代入したいときには、キャストという方法を用います。キャストは演算子の一つで、以下の書式を取ります。

```
(型名)変数
(型名)定数
(型名)(式)
```

「(型名)」がキャスト演算子です。キャスト演算子は、直後にある変数、定数、式の結果を指定された型へ変換します。

```
double   d = 8.93
int      j = (int)d;
double   e = (double)(89 / 3);
double   g = (double)89 / (double)3;     // 89.0 / 3.0と同じ意味
double   h = (double)89 / 3;             // 89.0 / 3  と同じ意味
```

最初のキャストでは、(int)がキャスト演算子で、double型の値をint型に変換しています。この場合、小数以下は切り捨てられます。次の例では注意が必要です。まず、括弧の中の89 / 3が計算されます。この数値は２つとも整数型なので、結果も整数型になります。すなわち、89 / 3→29となり、次にキャスト演算子が適用されます。すなわち「(double)29」となり、29.0がeに代入されます。残りの２つはコメントのとおりで、gとhには29.666666666666666666666666666667が代入されます。最後の89.0 / 3は異なる型どうしの演算です。このような場合は、より高位の型に暗黙的に変換されるので、 89.0 / 3.0という式と同じになります。

練習問題

以下のプログラムを実行したとき、それぞれの変数に代入される値を答えよ。

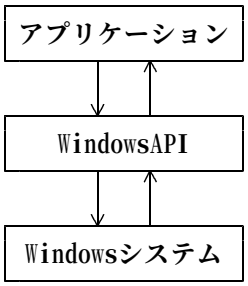
```
int      a = (int)893.893;
int      b =   5   / 1.2;
double   c = 893   / 2;
double   d = 893.0 / 2.0;
double   e = 1 / 3   + 2 * 3;
double   f = 1 / 3.0 + 2 * 3;
double   g = (double)(1 / 3) + 2 * 3;
short    h = (short)65535;
```

API

Windowsプログラミングを行うためには、「C言語とWindowsAPI」または「C++とMFCライブラリ(APIをクラス化したもの)」を用います。API(Application Programming Interface)とは、Windowsが提供する機能を呼び出すための数百種にもおよぶ関数の集まりのことです。

Windows上で動作するソフトウェアを開発する場合、ソフトウェアの持つすべての機能をプログラムするのは困難で無駄が多いため、多くのソフトウェアが共通して利用する機能は、Windowsが提供しています。プログラマは、規約に従ってAPIを「呼び出す」だけで、自分でプログラムを作成することなく、その機能を利用したソフトウェアを作成することができるのです。

Windowsプログラミングでは、膨大な種類に及ぶAPIを組み合わせてプログラムを構築していきます。




```
// APIの例…MessageBox関数(メッセージボックスを表示)
MessageBox(NULL, "続けますか？", "タイトル", MB_ICONEXCLAMATION | MB_OKCANCEL);
```

関数名　　引 数

引数

引数(ひきすう)とは、関数またはAPIを呼び出すときに、それらに引き渡す値のことをいいます。前述のMessageBox関数のカッコで括られた部分が該当します。MessageBox関数では、「NULL」「"メッセージ"」「"タイトル"」「MB_ICONEXCLAMATION | MB_OKCANCEL」という４つの引数を渡しています。

関数は、渡された引数をもとに最終的な動作を決定します。つまり、引数とは関数に与える動作指示と言えます。

戻り値

関数は、実行が終わると値を返すことがあります。これを、戻り値(返却値)と呼びます。たとえば、MessageBox関数は、メッセージボックスの処理が終わった後、押されたボタン(に割り当てられた値)を返します。

戻り値は、変数に代入することができます。以下のようにすると、変数nRetにはメッセージボックスのどのボタンが押されたが格納されます。

```
int  ret = MessageBox(NULL, "メッセージ", "タイトル", MB_ICONEXCLAMATION | MB_OKCANCEL);
```

戻り値は、関数の処理が成功したかどうかや、次の処理に必要な情報を返します。ある関数の戻り値を保存しておけば、次に実行する関数に渡したり、内容を調べて次の処理を分岐させたりというようなことができます。

戻り値のある関数は、数字のように扱われます。特別な意味はありませんが、以下のようにすることもできます。

```
ret = MessageBox(NULL, "メッセージ", "タイトル", MB_ICONEXCLAMATION | MB_OKCANCEL) + 1;
```

これをそのまま読むと、「メッセージボックスに１を足す」という意味不明なプログラムになりますが、MessageBox関数は戻り値を返すため、実際には「メッセージボックスの戻り値に１を足す＝押されたボタン(に割り当てられた値)に１を足す」という動作になります。つまり、OKボタンを押した場合は、

```
ret = IDOK + 1;
```

という意味になり、IDOKに割り当てられた値(１)に１が足されて２となり、変数retに代入されます。

練習問題

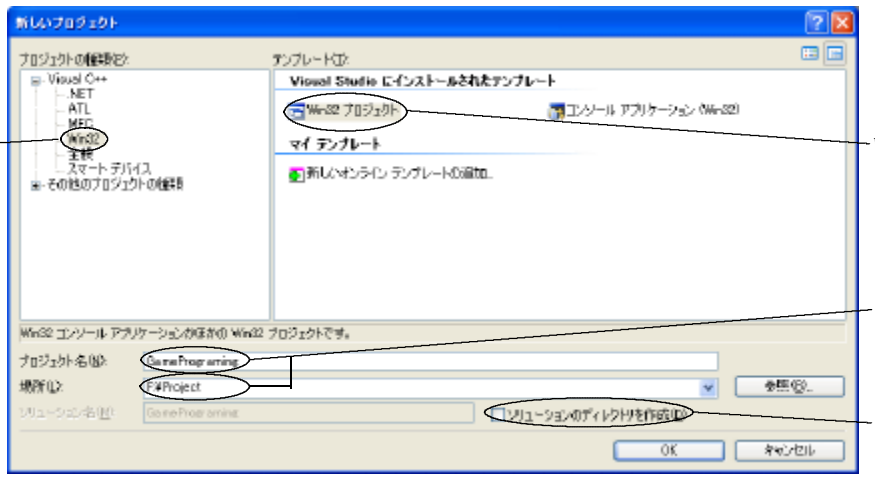
VisualC++で簡単なプログラムを作成し、実行してみましょう。

― VisualC++ キー操作一覧 ―

←, →	カーソルの１文字移動
↑, ↓	カーソルの１行移動
Enter	改行を挿入
Tab	タブを挿入
Insert	上書き/挿入モード切替
Delete	削除
Backspace	後方の１文字を削除
Shift + ←↑↓→	選択範囲の指定
Ctrl + A	すべて選択
Ctrl + C	コピー
Ctrl + V	貼り付け
Ctrl + X	切り取り
Ctrl + Y	やり直し
Ctrl + Z	元に戻す
Ctrl + S	ファイルの保存
F7	ビルド
Ctrl + Break	ビルドの中止
Ctrl + F5	プログラムの実行
F4	エラーがある行に移動

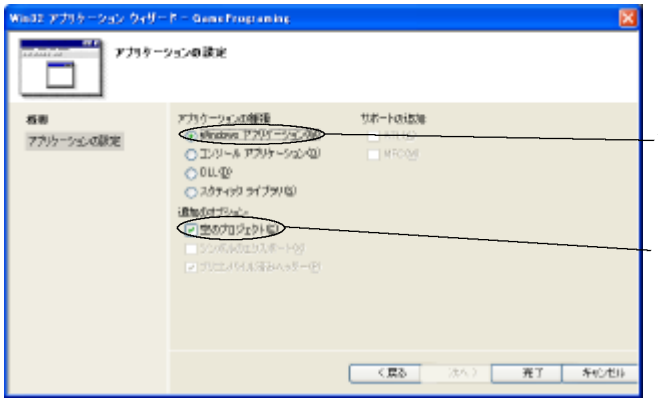
(１)アプリケーションのプロジェクトを作成しましょう。

①メニューから「ﾌｧｲﾙ(F)→新規作成(N)→ﾌﾟﾛｼﾞｪｸﾄ(P)」を選択し、新しいプロジェクトダイアログを開きます。



ﾌﾟﾛｼﾞｪｸﾄの種類の"Win32"、ﾃﾝﾌﾟﾚｰﾄの"Win32 ﾌﾟﾛｼﾞｪｸﾄ"を選択した状態で、「ソリューションのディレクトリを作成(D)」のチェックをはずしたあと、プロジェクト名(N)と場所(L)を入力し、OKボタンをクリックします。

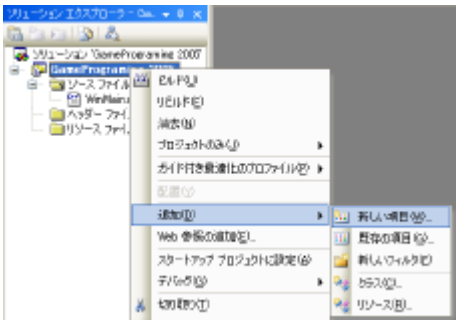
②「Win32 アプリケーション ウィザード」ダイアログの「概要」が表示されるので、「次へ」をクリックします。「アプリケーションの設定」になるので、「アプリケーションの種類」の「Windows ｱﾌﾟﾘｹｰｼｮﾝ」を選択して「追加オプション」の「空のプロジェクト」をチェックし、「完了」をクリックします。



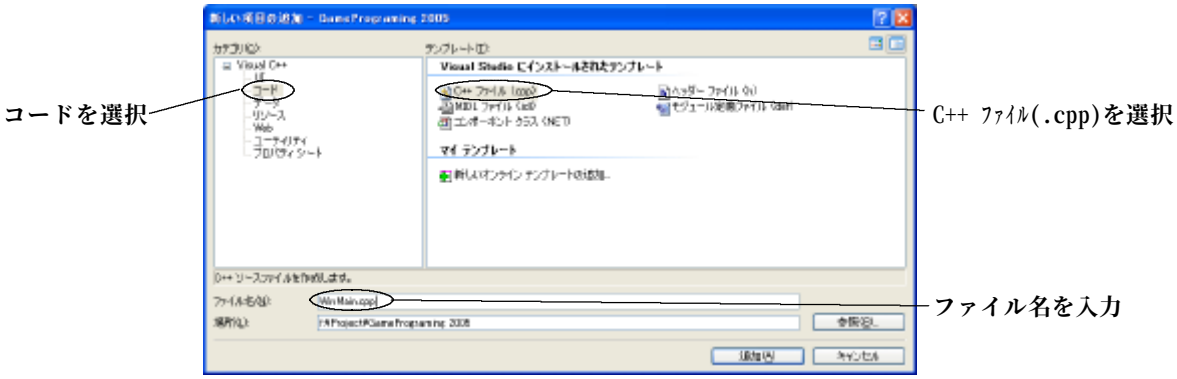
指定した場所に、プロジェクトのフォルダが作成され、その中にプロジェクトとソリューションのファイルが作成されます。

(２)WinMain関数を記述するソースファイルを作成しましょう。

①WinMain関数を記述するソースファイルを作成します。ソリューションエクスプローラのプロジェクト名が選択された状態で右クリックし、「追加▶新しい項目(W)」を選択します。



②新しい項目の追加ダイアログが開かれるので、「ｶﾃｺﾞﾘ(C)」の"ｺｰﾄﾞ"、「ﾃﾝﾌﾟﾚｰﾄ(T)」の"C++ ﾌｧｲﾙ(.cpp)"を選択した状態でファイル名に"WinMain.cpp"と入力し、追加ボタンをクリックします。



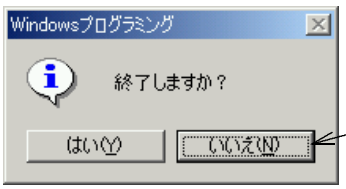
(３)以下のプログラムを入力し、実行結果を確認しましょう。

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int nShowCmd)
{
    MessageBox(NULL, "メッセージ", "タイトル", MB_ICONEXCLAMATION | MB_OKCANCEL);

    return 0;
}
```

(４)Win32API プログラマーズ リファレンス(API32JPN.HLP)またはMSDNライブラリでMessageBox関数を調べ、以下のメッセージボックスと同じものを作成しましょう。

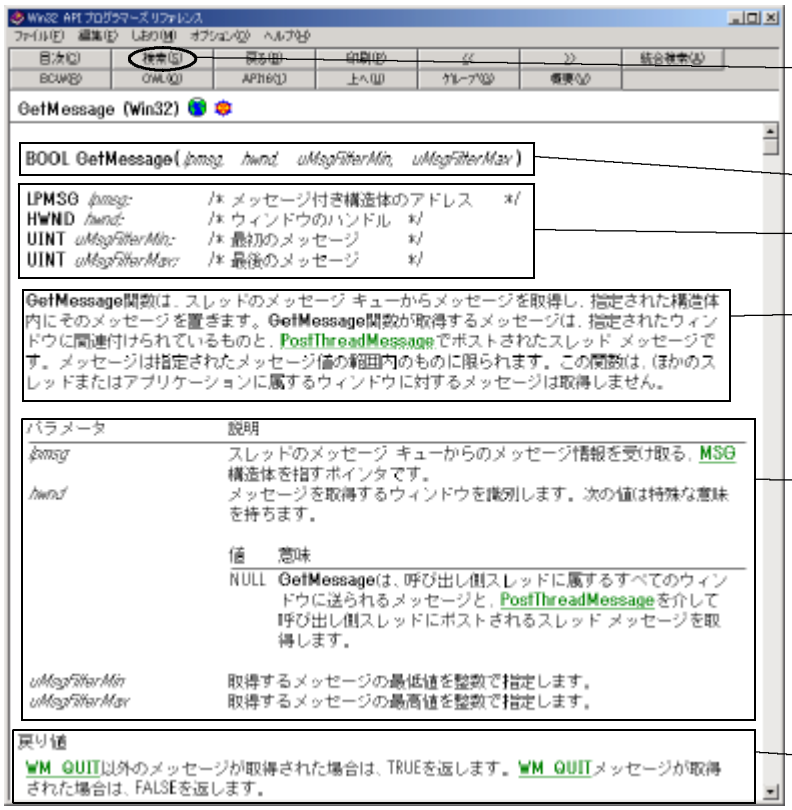


※「いいえ(N)」が選択されている状態にしましょう。

ヒント：複数の値を組み合わせるときは、"|"を使います。(例：MB_ICONEXCLAMATION | MB_OKCANCEL)

－APIリファレンス－

APIリファレンスは以下のようになっています。独特の表現が使われていますが、APIやDirectXのメソッドに関する有益な情報が記述されているので、読みこなせるようにしましょう。



検索(S)をクリックし、関数名を入力します

API(関数)のプロトタイプ

パラメータの型とその説明

APIの詳細な説明

パラメータについての解説。渡すべき値が詳細に記述されています。

関数の戻り値の説明

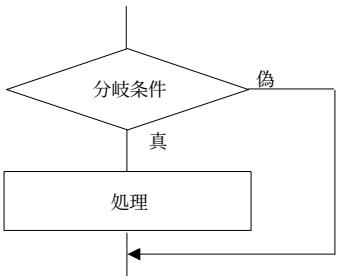
条件分岐

C／C++は、上から順番にプログラムが実行されますが、プログラムの流れを何らかの条件によって分岐させることができます。分岐は、if文やswitch文で行います。

if文は、

もし～ならば「処理」を実行し、
そうでなければ(「処理」を実行しないで)次の処理を実行する

という分岐をさせることができます。



if文の書式は以下のようになります。

```
if(条件) {
    条件を満たしたときに実行する処理
}
```

「条件」にはどのような条件でも記述できるわけではなく、基本的に２つの値を比較することしかできません。比較に使用できる演算子も決められており、以下のものしか使用できません。

```
==      !=      <       >       <=      >=
```

「==」は等しいかどうかを調べます。「==」の左側と右側が等しいときは真(成り立つ)、等しくないときは偽(成り立たない)となります。C／C++では、「=」が代入に割り当てられているので、比較は「==」を用いるようになっています。

「!=」は「==」の逆で、「!=」の左側と右側が異なるときは真(成り立つ)、等しいときは偽(成り立たない)となります。キーボードに≠がないため、このような記述になっています。

「<」「>」は大小比較に用います。「<=」「>=」は以上と以下(キーボードに≦と≧がないためにこのように記述します)を調べる場合に用います。

練習問題

以下の条件に従ってプログラムが動作するように空欄を埋めましょう。

(１)変数aと変数bが等しい場合に「等しい」と表示する

```
if(a （１） b) {
    MessageBox(NULL, "等しい", "情報", MB_ICONINFORMATION | MB_OK);
}
```

(２)変数xと変数yが等しくない場合に「等しくない」と表示する

```
if(x （２） y) {
    MessageBox(NULL, "等しくない", "情報", MB_ICONINFORMATION | MB_OK);
}
```

(３)変数zが18より大きい場合に「ＯＫ」と表示する

```
if(z （３） ) {
    MessageBox(NULL, "ＯＫ", "情報", MB_ICONINFORMATION | MB_OK);
}
```

(４)変数zが18より小さい場合に「ＮＧ」と表示する

```
if(z （４） ) {
    MessageBox(NULL, "ＮＧ", "情報", MB_ICONEXCLAMATION | MB_OK);;
}
```

(５)変数iが20以上の場合に「ＯＫ」と表示する

```
if(i （５） ) {
    MessageBox(NULL, "ＯＫ", "情報", MB_ICONINFORMATION | MB_OK);
}
```

(６)変数iが20以下の場合に「ＮＧ」と表示する

```
if(i （６） ) {
    MessageBox(NULL, "ＮＧ", "情報", MB_ICONEXCLAMATION | MB_OK);;
}
```

else節

メッセージボックスのOKボタンが押された場合は「OKボタンが押されました」と表示し、キャンセルボタンが押された場合は「ｷｬﾝｾﾙボタンが押されました」と表示するプログラムは、以下のようになります。

```
int  ret = MessageBox(NULL, "メッセージ", "タイトル", MB_ICONEXCLAMATION | MB_OKCANCEL);
if(ret == IDOK) {
    MessageBox(NULL, "OKボタンが押されました", "情報", MB_ICONINFORMATION | MB_OK);
}
if(ret == IDCANCEL) {
    MessageBox(NULL, "ｷｬﾝｾﾙボタンが押されました", "情報", MB_ICONINFORMATION | MB_OK);
}
```

このプログラムで使用しているメッセージボックスは、OKボタンとキャンセルボタンしかないので、２回目の分岐条件は「OKボタン以外が押されたとき」は「キャンセルボタンが押されたとき」と同じ意味になります。つまり、以下のように記述することもできます。

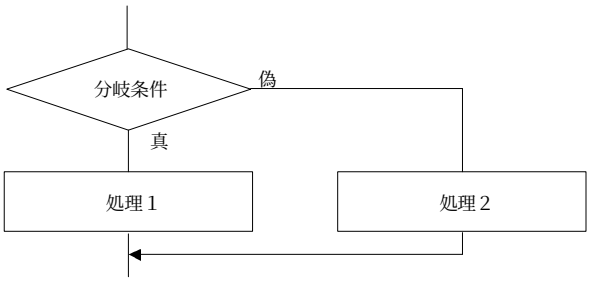
```
int  ret = MessageBox(NULL, "メッセージ", "タイトル", MB_ICONEXCLAMATION | MB_OKCANCEL);
if(ret == IDOK) {
    MessageBox(NULL, "OKボタンが押されました", "情報", MB_ICONINFORMATION | MB_OK);
}
if(ret != IDOK) {
    MessageBox(NULL, "ｷｬﾝｾﾙボタンが押されました", "情報", MB_ICONINFORMATION | MB_OK);
}
```

if文は、このような場合、else節を用いることにより、「そうでなければ」という分岐をさせることができます。つまり、判定の結果によって実行する処理を２つに分けることができるのです。これをif-else文と呼び、if-else文では、

もし～ならば「処理１」を実行し、
そうでなければ「処理２」を実行する

という分岐をさせることができます。if-else文の書式は以下のようになります。

```
if(条件) {
    条件を満たしたときに実行する処理
} else {
    条件を満たさないときに実行する処理
}
```



前述のプログラムをif-else文を用いて記述すると、

```
int  ret = MessageBox(NULL, "メッセージ", "タイトル", MB_ICONEXCLAMATION | MB_OKCANCEL);
if(ret == IDOK) {
    MessageBox(NULL, "OKボタンが押されました", "情報", MB_ICONINFORMATION | MB_OK);
} else {
    MessageBox(NULL, "ｷｬﾝｾﾙボタンが押されました", "情報", MB_ICONINFORMATION | MB_OK);
}
```

となります。

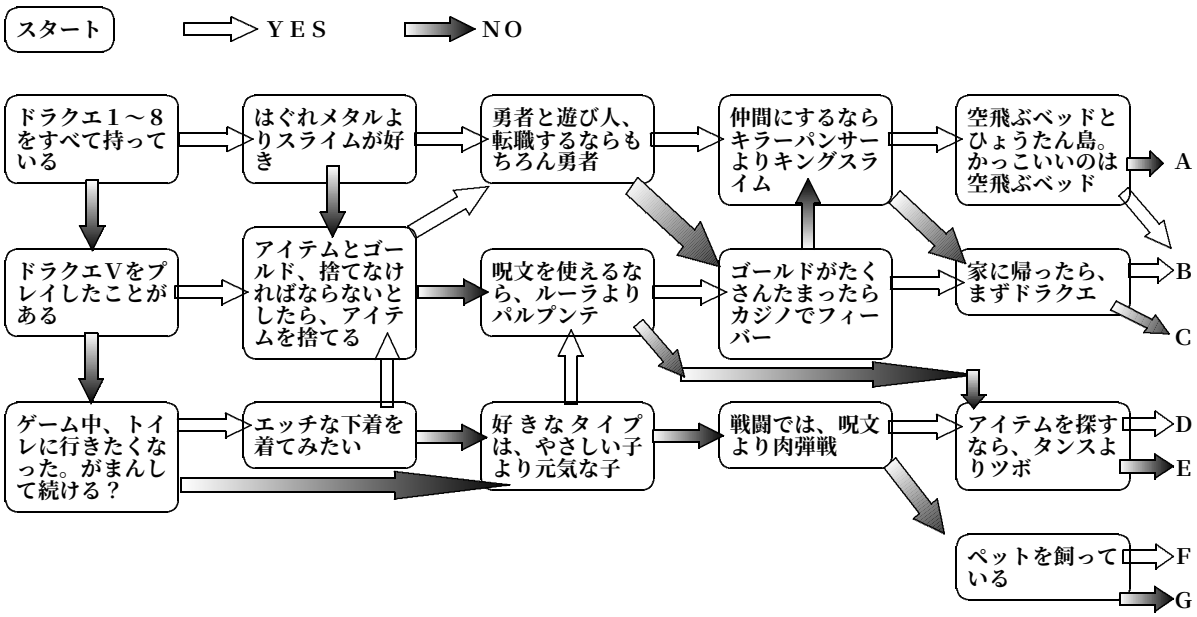
練習問題

1　以下のメッセージボックスと同じものを作成しましょう。

2　１のプログラムを変更し、はいボタンが押された場合は、以下のようなメッセージボックスを表示するようにしましょう。

3　２のプログラムを変更し、はいボタンが押された場合は「YES」、いいえボタンが押された場合は「NO」と表示するようにしましょう。

4　メッセージボックスとif文を用いて、以下のような相性診断を行うプログラムを作成しましょう。



A…主人公	主人公にたどり着いたあたなは、行動力大。ひとりでがんばれ
B…バーバラ	元気で明るいバーバラが、気弱なあなたををひっぱってくれるでしょう
C…チャモロ	いつもまじめなあなたは、チャモロと一緒に呪文を研究しましょう
D…ハッサン	自由気ままなハッサンが、あなたの力強い仲間になってくれるでしょう
E…テリー	クールなテリーと良い友達になれるでしょう
F…ミレーユ	やさしいお姉さんのミレーユに甘えてみたい、それが本心のようです
G…ファルシオン	決して戦わない、白馬のファルシオンと心が通じそう