オブジェクト指向とゲームプログラミング

● 基礎編 － 第４回　変数と演算子 ●

変数

コンピュータは、計算を行うときに使う数値などを記憶しておくために、メモリに数値を格納するための箱を用意します。この箱には、数値を入れたり、四則演算などで自由に内容を書き換えることができます。この箱のことを変数といいます。

```
int   a = 8, b = 9, c = 3;
```

a[8]　　b[9]　　c[3]

上の例では、a，b，cという変数名がつけられ、それぞれ８，９，３が入っています

変数名

変数に付ける名前のことを変数名といいます。コンピュータは変数名によって変数の区別を行います。C言語の変数名には、以下のような規則があります。

(１)最初の文字は、英字(AからZ，aからz)か下線(_)でなければなりません。
(２)２番目以降の文字は、英字、下線または数字(0から9)でなければなりません。
(３)大文字と小文字が区別されます。
(４)変数名の長さに制限はありませんが、実際には最初の３１文字までが有効です。
(５)途中に英字、下線、数字以外の文字があってはいけません。
(６)途中に空白があってはいけません。
(７)キーワード(予約語)であってはいけません。
(８)予約済み識別子ではってはいけません。

(３)の規則は、たとえば、abcとaBc，AbC，ABCはすべて異なる変数であることを示しています。また、(４)の条件は、abcとabcdは異なる名前ですが、次の２つの変数は、３１文字目まで(下線部分)が同じなので、同じ変数として扱われることになります。

abcdefghijklmnopqrstuvwxyz01234a
abcdefghijklmnopqrstuvwxyz01234b

(７)のキーワード(予約語)とは、あらかじめ意味の決まっている名前であって、int，double，short，longなど数10種類あります(VisualC++では、青色で表示されます)。これらの名前は、変数名に用いてはいけません。ただし、変数名の一部としては用いることができます。
いくつか変数名の正しい例を示しましょう。abc，XYZ，a123，a_123，a__1_2_3，very_long_long_nameなどは、すべて正しい名前です。

練習問題

以下の変数名は、すべて正しくない変数名です。どこが正しくないのかを説明しましょう。

誤った変数名	説明
7abc	
3.14159	
IN-data	
My Data	
@$100	
int	
モエモエ	
安藤忠	
∀	
1lI	

型

Ｃ言語では、整数のデータを入れる変数であることを決めてしまうと、その変数には整数値しか格納できません。実数を格納する変数を作ると、その変数には実数値しか格納できません。整数値を格納しようとしても、自動的に実数に変換されます。
このような変数の属性を型と呼びます。Ｃ言語では、以下の４つの基本の型があります。

型名	説明	Windows上での使用ビット数
char	文字を格納	8
int	符号付き整数	32
float	浮動小数点	32
double	倍精度浮動小数点	64

コンピュータは無限に大きな数字や、無限に大きな精度の数を記憶することができません。上の表のようにそれぞれの型には記憶容量の大きさが決まっていて、一定の範囲内の数しか格納することができないようになっています。それぞれの型の使用ビット数は、その型の変数を使うために使用する記憶容量の大きさです。ここで、Windows上と断っているのは、Ｃ言語の仕様では、これらの型で使う記憶容量の大きさが明確に規定されていないからです。Windows3.1やコンパイラによっては、int型が16ビットのものもあります。
さらに型の範囲を明確に定義するためにshortとlongと呼ばれる修飾子が用意されています。これを変数の宣言の前に付けて、その型が取りうる値の範囲を指定します。short，longを付けた型は、以下の３つがあります。

型名	説明	Windows上での使用ビット数
short int	短い符号付き整数	16
long int	長い符号付き整数	32
long double	拡張倍精度浮動小数点	Windowsではdoubleと同じ内部表現

短い符号付き整数、長い符号付き整数のビット数も、コンピュータやＯＳなどによって異なります。このように同じプログラムでも使うコンピュータによって精度が異なるということは、当然プログラミングのさまざまな場面に影響を及ぼします。そこでＣ言語では、それぞれの型での最低限保証される精度が定められています。short intとintは少なくとも16ビットあり、long intは少なくとも32ビットであることが保証されています。
また、符号付き整数に対して符号なし整数というものもあります。int，short int，long int，charは、符号付きの整数型ですが、これらのすべての型に対して、符号なしの整数として扱うようにすることもできます。そのためには、unsignedを各型の前に付ければよいのです。
たとえば、char型は文字を格納しますが、その範囲は-128から127までです(注：0～255という環境もあります)。しかし、unsigned charにすると、0から255になります。なお、unsignedの反対の意味としてsignedがあります。実は今までの定義ではこのsignedが省略されていたのです。これらをまとめると以下のようになります。

組み合わせた型	慣例的な型	Windows上での型	Windows上でのサイズ(バイト)	範　囲
signed char	char	char	1	-128～127
unsigned char	unsigned char	BYTE		0～255
signed short int	short	short	2	-32768～32767
unsigned short int	unsigned short	WORD		0～65535
signed int	int	int	4	-2147483684～2147483647
unsigned int	unsigned int	UINT		0～4294967295
signed long int	long	long		intと同じ
unsigned long int	unsigned long	DWORD		unsigned intと同じ
float	float	float		±10の±38乗(有効桁数7桁)
double	double	double	8	±10の±308乗(有効桁数15桁)
long double	long double	long double		Windowsではdoubleと同じ

以上のように、たくさんの型がありますが、整数を扱う場合は、たとえ小さな数値しか必要としない場合でもint型かunsigned int型、実数の場合はdouble型を使います。これらの型は、ＯＳ(ＣＰＵ)がもっとも高速に計算できるようになっています。そのほかの型は、特別な場合だけ使うようにします。たとえばDirect3Dでは、ほとんどの数値をfloat型で扱うため、double型を使っても数値が切り捨てられたり、コンパイル時に「警告」が発生することがあります。このような場合はfloat型を使います。

定数

123や893のように直接値を指定する数値を定数(constant)といいます。定数も実は型を持ちます。123や893と記述すると、これらの定数はintとして扱われます。intの範囲を超えるとunsigned intとして扱われます。
また、定数だけからなる式を定数式と呼びます。たとえば、次の式はすべて定数式です。

```
893
1 + 2 + 3 + 4 / 5
123 * (89 - 3)
```

小数点の付いた数値はdouble型として扱われます。

```
3.14159      1.41421356      8.93
```

小数点の付いた数値の後ろにfまたはFを付けるとfloat型として扱われます。

```
3.14159f     1.414213f       8.93f
```

変数の定義と初期化

変数を定義するためには以下のように記述します。

```
型　変数名;
```

たとえば、int型の変数を定義するには、

```
int   h;
int   m;
int   s;
int   t;
```

とします。これでh, m, s, tという名前の箱が用意されるわけです。また、カンマ記号を用いて、次のようにまとめて書くこともできます。

```
int   h, m, s, t;
```

このように、C言語では使用する変数をすべて定義しなくてはいけません。また、変数は使用する前に定義しておく必要があります。たとえば、以下のようなプログラムはエラーになります。

```
int   h, m, s;

h = 8;
m = 9;
s = 3;
t = 0;  // エラー！

int   t;
```

変数は定義されることにより、その箱がメモリに確保されます。しかし、その際に自動的に初期値として0が代入されるわけではありません。つまり、どんな値が入っているかわからない状態(不定)になっています。

```
int   x, y;

y = x + 100;
```

このプログラムを実行すると、yの値は意味のないものとなります。このように、初期化(最初に値をセットしておくこと)をしていないと意図しない結果になってしまいます。

const修飾子

円周率や光速度のような定数を用いて計算を行うことは多々あります。しかし、円周率や光速度のように常に、一定の値を変数に格納して計算に用いるというのは問題があります。なぜなら、変数の値は、その名の示すとおり、その内容を変更できるので、ミスや勘違いによってその定数の値が別のものに書き換えられてしまうことがよくあるからです。プログラムの実行中、ずっと同じ値である場合には、const修飾子を付けてその値を変えることができないようにします。

```
const double   pi = 3.14159265358979;    // 円周率
pi = 2.99792458;                          // エラー！
```

練習問題

(1)～(7)には、不適切な部分があります。不適切な部分を説明しましょう。

```
int    a = 8.93;                  // (1)
int    b = 10000000000;           // (2)

int    c;
int    d = c + 893;               // (3)

unsigned int   e = -1;            // (4)

double   f = 1.234567890123456; // (5)

float    g = 3.14159265358979;  // (6)
float    h = 0.0;                 // (7)
```

算術演算子

コンピュータの演算回路は、整数に対するものと浮動小数に対するものがあります。このため、C言語では、演算に関する規則が整数に関するものと浮動小数に関するものとに分かれています。

C言語の算術式では、加算と減算はそれぞれ+と-を用いますが、乗算は×ではなく*を用います。また、キーボードに÷のキーがないので、除算は/(スラッシュ)で代用します。たとえば、a × bはa * bと記述し、a ÷ bはa / bと記述します。括弧があれば、その中が先に計算されます。演算に関する代表的な演算子を紹介します。

算術演算子	Cでの記号	意味	備考
＋	+	加算	
－	-	減算	
×	*	乗算	
÷	/	除算	
余り	%	余り	整数のみ使用可

整数どうしの除算については注意が必要です。7 / 2は、3.5ではなく3となります。つまり、整数どうしの除算のときにはその答えも整数となるので、小数点以下は切り捨てられます。

%演算子は余りを求める演算子です。a % bは、aをbで割ったときの余りになります。たとえば10 % 3は、1になります。ただし、割る数と割られる数がどちらも整数でないと使用できません。

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, b, c, d, e, f, g;

a = 10;
b =  3;
```

```
c = a + b;
d = a - b;
e = a * b;
f = a / b;
g = a % b;

double   h, i, j, k, l, m, n, o;

h = 10.3;
i =  3.3;
j = h + i;
k = h - i;
l = h * i;
m = h / i;
n = h % i;
o = 3.0 % 2.0;
```

代入演算子

代入演算子は、代入動作をする演算子であり、

左辺値　代入演算子　右辺値

の形式を取り、「右辺値の値を計算し、その結果を演算子の規則に従って、左辺値に適用する」演算子のことをいいます。a = 89やb = aという式に示されているように、左辺値は変数、右辺値は数値や変数のことをいいます。

代入演算子には、以下に示すものがあります。

代入演算子	説明	動作
=	代入する	op1 = op2
+=	加算する	op1 = op1 + op2
-=	減算する	op1 = op1 - op2
*=	乗算する	op1 = op1 * op2
/=	除算する	op1 = op1 / op2
%=	剰余する	op1 = op1 % op2
<<=	左シフトする	op1 = op1 << op2
>>=	右シフトする	op1 = op1 >> op2
&=	ANDでビット演算する	op1 = op1 & op2
^=	排他的ORでビット演算する	op1 = op1 ^ op2
|=	ORでビット演算する	op1 = op1 | op2

上の演算子を用いた文で、たとえば、

```
x += 89 + 3;
```

は、以下の文と同じ効果を示すことを意味しています。

```
x = x + (89 + 3);
```

一方、

```
1 = 2;       a = 1 += 2;
```

などはエラーになります。代入動作において、右辺値は左辺の型に変換されます。

```
int      pi = 3.141592654;
double   qi = 1;
```

この場合、piへの代入動作の際、3.141592654はint型に変換され、その結果、小数点以下が切り捨てられ、piへは3が代入されます。qiへの代入では、int型の1がdouble型の1.0に変換されて代入されます。

練習問題

int型の変数aに89，bに3，double型の変数hに8.9，iに0.03が格納されているとき、以下の(１)から(10)を単独で実行したときの値を応えよ。

```
a += b;      // (１)
a -= b;      // (２)
a *= b;      // (３)
a /= b;      // (４)
a %= b;      // (５)
h += i;      // (６)
h -= i;      // (７)
h *= i;      // (８)
h /= i;      // (９)
h %= i;      // (10)
```

課　　題

メインループに関する以下の文章を読み、課題をクリアせよ。

ゲームプログラムは、「ものの状態の集まり」と考えることができます。キャラクタの座標やパラメータ、背景など、それぞれ「もの」の状態です。ゲームプログラムとは、このいろいろな「もの」の状態を時間に沿って更新し、それを画面に描画するプログラムと考えることができます。
つまり、ゲームプログラムの処理を非常に単純化すると、

・「もの」の状態を更新(内部処理)
・「もの」を画面に描画(描画処理)

という２つにまとめることができます。
この２つの処理の繰り返しがメインループになります。このループ１回ぶんを「１フレーム」と呼びます。１フレームごとに、キャラクタの位置や状態、背景やそのほかの表示状態などを更新し、その状態を反映して描画を行う、という処理を繰り返していきます。

(１)War'Dのプログラムを変更し、プレイヤーのＨＰがつねにＭＡＸになるようにしましょう。

－操作方法－
↑，↓，←，→…移動
Ｚ…ショット
Ｘ…スラッシュ
Ｃ…ボム

－ヒント－
１．War'Dのメインループは、GameFunc2.cppのStageProc関数です。
２．lMaterWidthという変数が、プレイヤーのＨＰを保存しています。
３．ＨＰの最大値は１００です。
４．メインループは、定期的に実行されます(１秒間に６０回)。
５．メインループの○○で、lMaterWidth変数に○○○を代入すれば、ＨＰが減ってもすぐに戻ります。

(２)War'Dのプログラムを変更し、ボムゲージがつねにＭＡＸになるようにしましょう。

－ヒント－
１．MaterWidthという変数が、ボムゲージの量を保存しています。
２．ボムゲージの最大値は１００です。

(３)ザコ敵の無敵時間をなくしましょう。

－ヒント－
１．bMuteki[i]という変数が、ザコ敵の無敵状態を管理しています。
２．１の変数には、trueかfalseを代入することができます。
３．敵の設定は、EnemyData4.cppのEnemyInit関数で行っています。

(４)いきなりボスが登場するようにしましょう。

－ヒント－
１．すべてのザコ敵を倒すと、ボスが登場します。
２．倒されたザコ敵は、「消えた」という状態になります。
３．g_chEnemy[i].eStateという変数が、敵の状態を保存しています。
４．３の変数は、「DERU」「KESU」「MADA」「SRASHED」という状態を設定することができます。

(５)プレイヤーを無敵にしましょう。

－ヒント－
１．プレイヤーの衝突領域を０にするのがもっとも簡単です。
２．GameFunc2.cppのStage関数でプレイヤーの設定をしています。
３．「// 自機当たり判定」でプレイヤーの衝突領域を設定しています。

(６)ザコ敵を一撃で破壊できるようにしましょう。

(７)DivineJailのプログラムを変更し、プレイヤーの初期ステータスを任意の数値にしましょう。

－操作方法－
・通常時
　↑，↓，←，→…移動
　Ｚ…マップ表示
　Ｘ…メニュー画面
　Ｃ…周辺マップ表示

・戦闘時
　↑，←，→…移動
　↓…ガード
　Ｚ…攻撃
　Ｘ…攻撃
　アルファベット…呪文入力(Enterで確定)

－ヒント－
　Command.cppファイルのPlayerSet関数

(８)ボーナスポイントを９９にしましょう。

－ヒント－
　Command.cppファイルのInputStetas関数

(９)戦闘終了後にもらえる経験値を１０００にしましょう。

－ヒント－
１．GameFunc.cppファイルのRtnFight関数
２．ALLExp変数がもえらる経験値

(10)戦闘中無敵になるようにしましょう。

－ヒント－
１．GameFunc.cppファイルのRtnFight関数
２．プレイヤーの状態を「ダメージを受けた直後の無敵」状態にする
３．戦闘中の状態は、g_chHuma.Fightmoveという変数