Ⅳ オブジェクト指向分析・設計

学習目標
① オブジェクト指向分析の手順を理解させる。
② オブジェクト指向設計の手順を理解させる。
③ クラスの見つけ方を身につけさせる。

1 オブジェクト指向分析・設計・プログラミング

前章では、オブジェクト指向プログラミングを行うための言語として、Ｃ＋＋言語を解説した。しかし、オブジェクト指向の考え方を身につけるという視点からは、Ｃ＋＋言語のようなオブジェクト指向プログラミング言語の文法を習得するよりも、もっと大切なことがある。それは、オブジェクト指向の考え方を習得し、それに基づいて問題を分析し、設計できる力を養うことである。

問題をよく理解しないで、コンピュータに向かっていきなりプログラミングすることのできる時代は、既に終わりを迎えた。また、構造化設計に基づくソフトウェア開発手法も、ソフトウェアの規模が大きくなるにつれて、生産性及び保守性の面で限界を露呈した。

そして、現在のソフトウェア開発に求められているのは、大規模で複雑な問題を扱っても、生産性及び保守性を損なうことのない新しい手法である。

オブジェクト指向に基づくソフトウェア開発手法は、信頼性と保守性が高いプログラムを短期間で作成できる、これまでにない画期的な手法である。これまでの手法では、解決すべき問題に対して、それを解決するためには、どのような手続きをどのような順序で行えばよいかという観点から解決策を探ってきた。一方、オブジェクト指向に基づくソフトウェア開発手法では、解決すべき問題をコンピュータ上で擬似的に再現する、すなわち、シミュレーションを行うという観点から解を得る。この手法と、手続きにこだわる従来の手法との相違は大きく、最初は難しく感じるかもしれない。しかし、シミュレーションを使った解決法というのは、現実世界をそのままプログラミングし、コンピュータ上で再現するというものであり、無理のないごく自然な方法
であるといえる。

オブジェクト指向の考え方に基づく分析・設計・プログラミングは、それぞれオブジェクト指向分析（Object Oriented Analysis、ＯＯＡと略することもある。）、オブジェクト指向設計（Object Oriented Design、ＯＯＤと略することもある。）、オブジェクト指向プログラミング（ＯＯＰ）と呼ばれる。

それでは、オブジェクト指向に基づくソフトウェア開発手法のおおまかな手順について説明する。

まず、解決しようとしている問題が何なのかを分析する。これを問題領域と呼ぶことにする。　次に、その問題領域に適したクラスライブラリを用意する。もちろん、これは他人が作ったものでもよいし、自分で作ったものでもよい。

次に、問題領域のクラスの候補を抽出し、クラスを設計する。ここでは、クラスの継承関係も含めて考えることが必要となる。

最後に、設計されたクラスの振る舞いをプログラミング言語で実際にプログラミングする。

ここで特徴的なことは、問題領域のクラスを設計するときに、最善でないと思ったら何回前に戻ってやりなおしてもよいということである

従来の開発手法であったウォーターフォールモデルでは、水が上から下へと流れるように作業を行うことになっていた。しかし、現実には前段階の設計が不十分なために、前段階に戻って作業することが多かった。これは「手戻り」と呼ばれ、良くないこととされてきた。しかし、良くなかったのは手戻りではなく、手戻りを引き起こさせるようなソフトウェア開発手法にあったのである。

オブジェクト指向に基づく手法は、既に存在するものから未知のものを作りだすというものである。そのため、何でも全てわからないと気が済まないという考え方は捨てる必要がある。ソフトウェアの開発者は、とかく、ブラックボックスであるライブラリの中身が見えないと、そのブラックボックスを使おうとしないものである。それは、ブラックボックスを信用していないからである。確かに、これまでの手続き型言語によって開発されたライブラリは、結合度が高く下手に使用すると、期待した動作をしてくれないものが多かった。しかし、ソフトウェア開発の複雑さを軽減するためには、他人の作ってくれたライブラリを、中身は全てわからなくても使う勇気と度量をもつことが必要である。オブジェクト指向プログラミング言語は、信頼の高いクラスライブラリを提供することができる仕掛けを、クラスライブラリ開発者に提供しているのである。

2 クラスの抽出

(1) クラスの継承関係

オブジェクト指向分析において、問題領域からクラスを抽出するという作業が最も重要な作業である。

この作業は、クラスの継承関係というものに着目して行う。次の２種類の関係がクラスの抽出に役に立つ。

イ 「is a」の関係

「is a」の関係とは、英語の

X is a Y.

という文に対応する関係である。

XはYの一種である。

という風に訳すことができる。

これは、クラスの継承関係、すなわち、親クラスと子クラスの関係を抽出するのに利用する。XはYの一種なのだから、Xが子クラス、Yが親クラスとなる。

例えば、「くだもの」と「リンゴ」という二つのことばが存在するときに、それらの関係が上の文に当てはまれば、それらの間には継承関係があることになる。

「くだもの」は「リンゴ」の一種である。・・・・・（１）

「リンゴ」は「くだもの」の一種である。・・・・・（２）

という二つの文が正しいかを考えてみると、（２）の文の方が正しそうである。そこで、

「くだもの」という親クラスと、「リンゴ」という子クラスを抽出することができる。

□ 「has a」の関係

「has a」の関係とは、英語の

X has a Y.

という文に対応する関係である。

XはYを（その一部として）所有している。

という風に訳すことができる。

これは、クラスが保持すべきデータを抽出するのに利用する。YはクラスXの内部データとなる。

例えば、「車」と「ハンドル」という二つのことばが存在するときに、それらの関係が上の文に当てはまれば、それらの間にはクラスと内部データの関係があることになる。

「車」は「ハンドル」をその一部として所有している。・・・・・・（1）

「ハンドル」は「車」をその一部として所有している。・・・・・・（2）

という二つの文が正しいかを考えてみると、（1）の文の方が正しそうである。そこで、「車」というクラスと、「ハンドル」という内部データを抽出することができる。

3 オブジェクト指向分析

オブジェクト指向分析は、その後の設計、プログラミングの指針になるものであり、大変重要である。

分析とは、システムはクラスの集合と考え、システムの中にどのようなクラスが含まれているか、あるいは、クラスから生成されたオブジェクトがどのように動作するのかを解析するものである。

ところで、オブジェクト指向におけるクラスの分析や設計は、人によって異ってかまわない。それは、分析や設計をする人によって、問題領域の解釈が異なるからである。ただし、優れた分析ができる人は、洗練されたシステムを構築できることも確かである。

それでは、分析の手順を順次説明していく。

(1) 仕様の文書化

どのようなものを開発するのかということは、最初に考えるべきことである。開発者は、まず、開発するものの仕様を決定する。そして、それを文書にしておく。

仕様を文書にすることにより、頭の中を整理することができる。

オブジェクト指向の場合、従来のウォーターフォールモデルのように、1回仕様を決めたら、それを変更できないという硬直した考え方をもつ必要はない。後戻りが絶対できないように、仕様を完全に固める必要もない。オブジェクト指向では、システムを変更することに対して、非常に柔軟に対応することができる。例えば、あらかじめプロトタイプと呼ばれる試作品を作成し、その動作や形状を開発者と利用者の双方が見て、それから再度仕様を固めるといった、プロトタイピングという開発手法にも、オブジェクト指向は対応することができる。

(2) クラス候補の抽出

クラスをいきなり抽出せよといっても、それはオブジェクト指向にはじめて取り組む開発者には雲をつかむような話である。そこで、具体的に説明する。

(1)で決定した仕様は、文章になっている。その中から名詞を抽出する。これは、クラスの単位が名詞的であるという性質を利用したもので、この作業によりクラスの候補を仕様から抽出したことになる。

(3) 問題領域のクラス候補の抽出

次に、(2)で抽出した候補のうち、問題領域のクラスのみを取り出す。問題領域とは、システムを開発することによって、解決しようとしている問題の範囲のことである。

仕様を記述した文章には、問題領域のほかにコンピュータ自身に関する内容も含まれている。例えば、「キーボードを押す」といった文章があった場合、「キーボード」という名詞は、問題領域に関係したものではなく、コンピュータ自身に関係したものである。これは、これから解決しようとしている現状の問題には登場しないので、クラスの抽出には関係ない。

また、クラスは、それぞれただ一つの目的を持つべきであり、複数の目的をもつクラスは望ましくない。このようなときはクラスを分割する。更に、クラス間で相互に強い依存性をもたないようにすべきである。そして、クラス候補で抽出したが、システムにあまり関係のないものは、クラス候補から消してやる。これらは、実はもともとシステムの構成要素ではなかったということである。

ここで大切なことは、どうやってソフトウェアを作ろうかという気持ちをできるだけ押さえ、冷静に仕様を観察し、クラス候補を抽出することである。どうやってソフトウェアを作ろうかという気持ちは、邪念であると割り切り、邪念に邪魔されないよう、無心に抽出をするという気持ちこそが大切である。

4 オブジェクト指向設計

オブジェクト指向設計においては、問題領域のクラスを抽出し、それらに名前をつけ、更にメンバ関数、データメンバを抽出するという作業を行う。これらを行うときは、常にオブジェクト指向の考え方を意識した上で行うことが大切である。そうすることによって、クラスの設計を的確に行えるようになる。

一方、もしこれらの作業を無意識に行った場合、設計に失敗することが多い。また、失敗した場合、「オブジェクト指向というのは、導入する意味がない。」という評価をされてしまう場合が多い。

オブジェクト指向におけるクラスの設計は、訓練を重ね経験を積むことによってかなり上達する。

また、クラスは、ただ存在するだけでは意味がない。他の人がメッセージを送ってはじめて意味をもつものである。オブジェクトにメッセージを送るのは、メインルーチンと呼ばれるプログラムであったり、それ以外のクラスであったりする。

オブジェクト指向設計は、以下の手順で行う。これは、これまでの設計法よりも容易であり、かつ見落としの少ない手順である。

(1) クラスの命名

ここまでで抽出した問題領域のクラス候補は、クラスとしての地位を得たものとし、それらをクラスとすることにする。

そこで、クラスを識別するための名前をつける。一つのクラスに、一つの名前とする。このとき、クラスの本質を言い当てたわかりやすい名前をつけることが望ましい。例えば、Class1といった意味のない名前は望ましくない。Television、AirConditionerといったそのものずばりの名前がわかりやすい。わかりやすい名前にすることによって、ソフトウェアの再利用を促進することができる。また、プログラム上では、クラスをアルファベット順に並べ、すぐに位置がわかるようにする工夫も必要である。

命名のときの規則については、後に説明することにする。

(2) クラスごとの仕様の文書化

各々のクラスの名前が決まったところで、次はそれらのクラスごとの仕様を文書化する。この文書は、分析作業を行っているときから保守作業を行うときまで、長い期間使用される重要なものである。

また、この作業中に新たにクラス候補が見つかることもある。そのような場合は、それもクラスとして新たに追加してやる。このあたりは、後戻りをしながら前進をするという余裕をもった気持ちが必要である。ここに時間をかけ、開発しようとしているシステムの詳細を、頭の中でより具体化していくと、当然、更に良いと思われる仕様が登場してくる。

(3) クラス間の関連抽出

データベースの世界では、現在リレーショナルデータベースが主流を占めている。これは、データをテーブルと呼ばれる容器に入れ、その容器から必要なデータを取り出すという考え方でできている。そして、テーブルとテーブルは、リレーション（関係）と呼ばれる概念で結びつけられている。

オブジェクト指向の場合も、クラスとクラスが「関連」と呼ばれる概念で結び付けられている。それぞれのクラスの間でメッセージのやりとりという関係がある場合、「これらのクラスの間には関連がある」と呼ぶ。オブジェクト指向に基づいた設計を行う場合、どのクラスとどのクラスの間に関連があるか、ないかを明確にしておくことは重要である。

クラスは、ただ単体でぽつんと存在しても、あまり役には立たない。クラスは、それぞれ関連をもちあって、はじめて役に立つ。また、各クラスの関連には、１対１の場合、１対多の場合、多対多の場合という３種類の場合がある。関連を調べるときには、どの種類なのかを同時に考えておく必要がある。

また、関連とデータメンバは別物ではあるが、よく間違えやすい。この二つを見分けるには経験が必要である。

(4) クラスごとのメンバ関数抽出

クラスに対して、メッセージを送るときの受け手となるメンバ関数を抽出する。メンバ関数は、クラスの外部からのメッセージを受け取る受付窓口である。受付はふつう複数個ある。

ここで注意しなければならないのは、クラス内部のデータメンバを先に設計するという

ことである。従来の設計法では、まずデータ構造を設計し、次に、そのデータ構造を利用して関数を設計するという手順であった。しかし、オブジェクト指向では、クラス内部のデータ構造よりも、クラス外部の受付窓口となるメンバ関数のほうがはるかに重要な意味を持っており、こちらをさきに設計するべきである。

(5) クラスごとのデータメンバ抽出

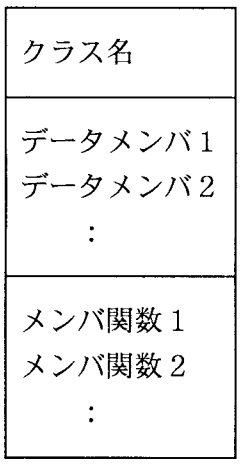
クラスの内部のデータメンバを抽出する。これは、最も優先順位が低い項目である。例えば、メンバ関数はそのままにして、内部の処理は変更するといったことが、自由にできるのが、オブジェクト指向設計の長所である。そこで、とりあえずデータメンバを設計しておけばよい。

ただし、データメンバが一つしかないクラスは要注意である。もし、データメンバが一つで、しかも、そのデータメンバに対する値の設定と値の取得というメンバ関数しか存在しないクラスがあったら、これは他のクラスのデータメンバとして吸収させる方がよい。

(6) クラスの図示

ここまでで固まった各クラスの内容を次は図の形で表現する。このような図を「クラス図」と呼ことがある。クラス図は、クラスの内容や関連を視覚に訴える効果がある。これによって、システムの開発者とユーザとの間でスムースな意思の疎通が行える。

クラス図の記述方法には、さまざまな流儀があるが、ここでは長方形を横に3段に区切る流儀について説明する。それは、1段目に「クラス名」、2段目に「データメンバ」、3段目に「メンバ関数」を記述するというものである。



5 クラス設計の指針

クラスを設計する手順は、既に説明したとおりである。ここでは、いったんクラス設計を終えた後、再度クラス設計が最適かを見直すときの指針について説明する。

(1) クラスの継承関係の見つけ方

クラスを設計するとき、クラスの再利用が可能なように一般化する。もし、必要ならば複数のクラスに共通の部分を見つけて、親クラスを新たに作成する。

(2) クラス設計の指針

クラスの継承の関係は適当か否かを調べる。一つのクラスに課せられた責任が重すぎはしないか、あるいは軽すぎはしないか、クラスやデータメンバ・メンバ関数の名前は適切か、機能は複数のクラスで重なっていないか、他のクラスのデータを直接参照・変更したりしていないか、クラスの外部からクラス内のデータメンバを参照・更新してはいないか等をチェックする。

6 オブジェクトの分析・設計例

これまで説明してきたオブジェクト指向分析・オブジェクト指向設計の例として、「ブロックゲーム」を取り上げる。これはパソコン上で動作するゲームである。これを分析し、設計する手順に従って詳しく説明していく。

(1) クラスライブラリの構築

クラスライブラリは、ふつう、コンパイラを開発しているメーカが提供することが多い。しかし、今回は、自分でクラスライブラリを作成することにする。

それは、以下の三つのクラスからなる、クラスライブラリである。そして、こららのクラスを基本クラスとして、継承してソフトウェアに使用する。

イ 基本関数ラップクラス：ＣＷｒａｐ

基本的な関数をラップ（包み込む）したクラスである。これは、機種やコンパイラ依存の関数などを集めたものであり、異なる機種やＯＳに移植するときに、変更を容易にするためのものである。

ロ 画面表示クラス　　：ＣＳｃｒｅｅｎ

ＣＷｒａｐクラスから派生させたクラスである。

- キーボード入力：メッセージループを抽象化したもの
- ディスプレイ表示：整数（int、long）や文字列（char）の表示を抽象化したもの
- 一時停止：入力があるまで一時的に処理を停止するもの
- カーソル管理：カーソル位置を２バイト系の画面配列で管理するもの

ハ ファイル入出力クラス：ＣＤａｔｕｍ

ＣＷｒａｐクラスから派生させたクラスである。

- 文字列をファイルに書き込み、また、読み出す

(2) ブロックゲームの仕様の文書化

以下に、ブロックゲームの仕様を示した文書を示す。

『コンピュータの画面上に、長方形の枠がおかれている。その枠内の上部から、様々な形をしたブロックが生成され、徐々に落ちてくる。そして、枠内の床か、あるいは他のブロックに接触し、移動できなくなると、そのブロックは停止し、得点が加算される。停止したブロックは、背景の一部になる。そして、再度上部から、新たなブロックが生成され、

落ちてくる。もし、背景のブロックが高く積み重なり、新たなブロックが生成された瞬間からブロックが移動できなくなっていたら、ゲームは終了する。

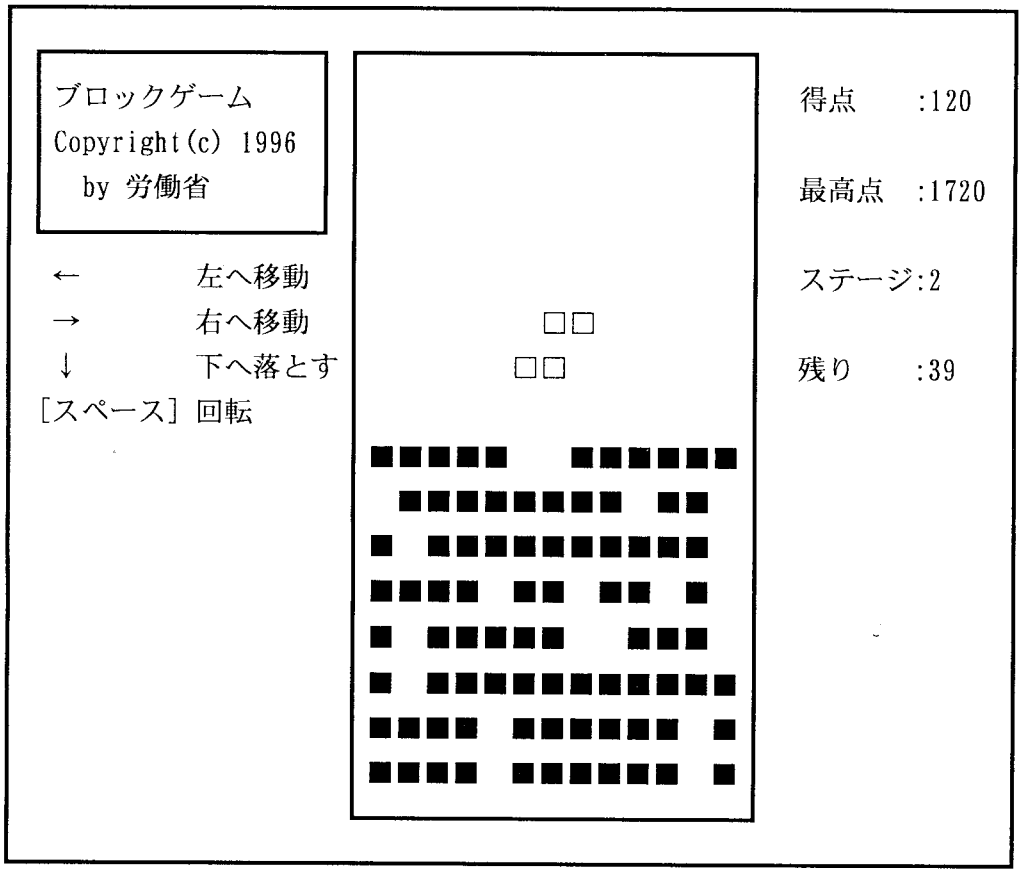
ユーザは、落ちてくるブロックを、「←」「→」キーを押して左右に移動させたり、「　」（スペース）キーを押して回転させたりすることができる。

そして、もし、停止したブロックによって、枠内の背景のブロックの一段をいっぱいにすることができると、その段の表示は消え、その段より上にあった部分がすべて一段、下に移動する。そして、得点が加算される。また、一度に多くの段を同時に消すと、より高い得点が加算される。そこでユーザは、段が消えるようにブロックを移動、回転させながら、なるべく長くゲームを続け、高い得点を競う。最高点は、ファイルに保存される。

また、ゲームは９つのステージに分かれており、一定の個数のブロックが生成されると、次のステージに進むことになる。ステージが進むごとに、ブロックが速く落ちてくる。また、それに応じて高い得点が得られる。第９ステージが終了すると、また第１ステージに戻るが、ブロックが生成される位置は、以前よりも低くなる。

ユーザは、「↓」キーを押して、ブロックを急いで下に落とすこともできる。各ブロックには、色がついている。また、ブロックの形をあらかじめ決めておくことができる。』

次に、どのような画面のイメージになるかを図Ⅳ－１に示す。

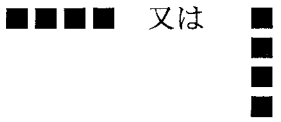


図Ⅳ－１　ブロックゲームの画面イメージ

また、あらかじめ決めておくブロックの形は次の６種類とする。

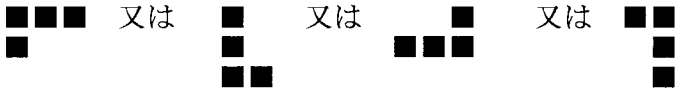
イ　Ｉ型

英語のＩに相当するように４個のブロックが並ぶもの。



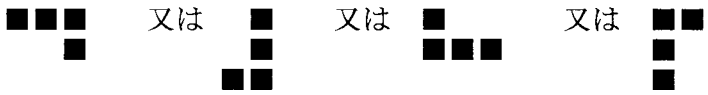
ロ　Ｌ型

英語のＬに相当するように４個のブロックが並ぶもの。



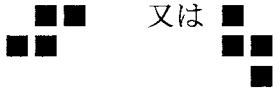
ハ　逆Ｌ型

英語のＬの逆位相に相当するように４個のブロックが並ぶもの。



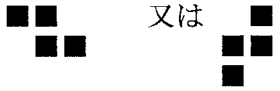
ニ　Ｓ型

英語のＳに相当するように４個のブロックが並ぶもの。



ホ　逆Ｓ型

英語のＳの逆位相に相当するように４個のブロックが並ぶもの。

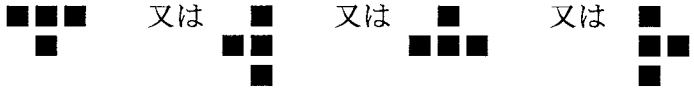


ヘ　Ｏ型

英語のＯに相当するように４個のブロックが並ぶもの。

ト　Ｔ型

英語のＴに相当するように４個のブロックが並ぶもの。



(3) クラス候補の抽出

仕様の文章からクラス候補として名詞を抽出する。それは、図Ⅳ－２に示すような以下の名詞である。

「コンピュータの画面」「枠」「床」「ブロック」「得点」「背景」「ゲーム」「ユーザ」
「「←」キー」「「→」キー」「「 」（スペース）キー」「一段」「最高点」
「ファイル」「ステージ」「「↓」キー」「ブロックの形」

図Ⅳ－２　クラス候補の名詞

(4) 問題領域に属しているクラス候補の抽出

図Ⅳ－２から、問題領域に属しているクラス候補のみを抽出する。また、他のクラスに含めたほうがよいものは含める。その結果、残ったクラス候補を図Ⅳ－３に示す。

・「背景」（「枠」と「床」と「一段」は、これに含めた。）
・「ブロック」
・「ブロック原形」（「ブロックの形」を改めた。）
・「得点」
・「最高点」
・「ステージ」

図Ⅳ－３　問題領域に属しているクラス候補

(5) クラスの命名

図Ⅳ－３で抽出した問題領域に属しているクラス候補を大きく２種類に分類し、それぞれにクラスの名前をつける。

ｲ　アプリケーション

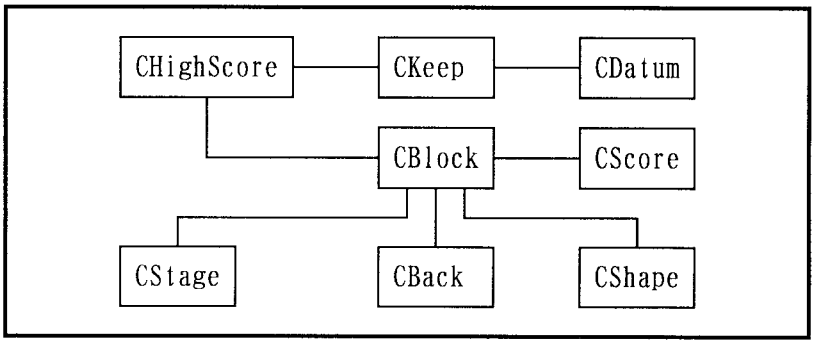
背景クラス	：ＣＢａｃｋ
ブロッククラス	：ＣＢｌｏｃｋ
ブロック原形クラス	：ＣＳｈａｐｅ
得点クラス	：ＣＳｃｏｒｅ
最高点クラス	：ＣＨｉｇｈＳｃｏｒｅ
ステージクラス	：ＣＳｔａｇｅ

ﾛ　クラスライブラリ

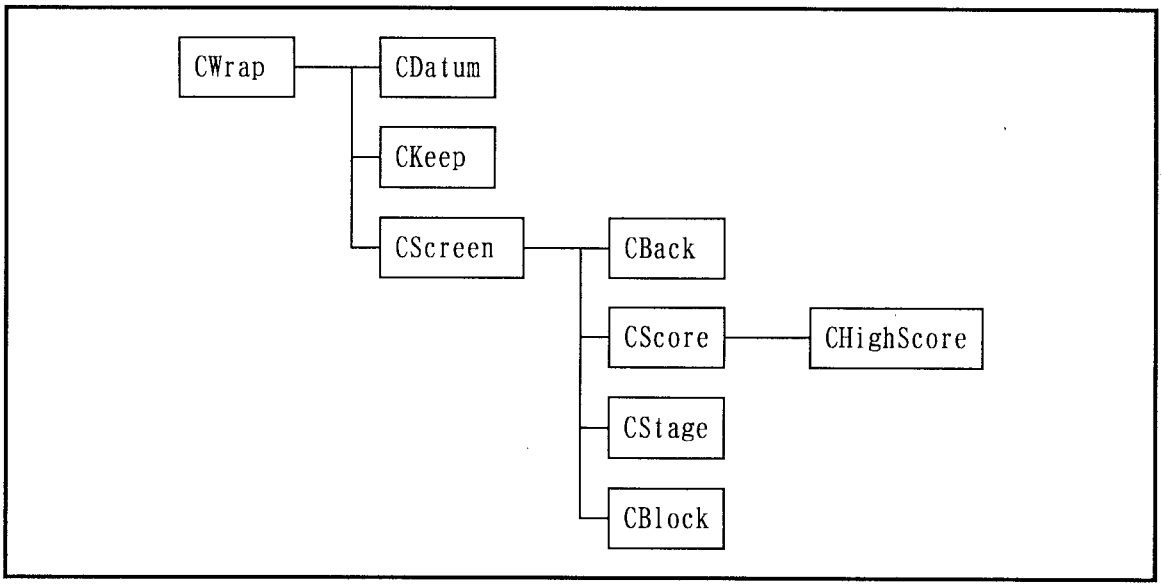
基本関数ラップクラス	：ＣＷｒａｐ
画面表示クラス	：ＣＳｃｒｅｅｎ
ファイル入出力クラス	：ＣＤａｔｕｍ
最高点保存クラス	：ＣＫｅｅｐ

(6) クラス間の関連抽出

各クラス間の関連は、図Ⅳ－４のように表すことができる。また、各クラスの継承関係は、図Ⅳ－５のように表すことができる。



図Ⅳ－４　各クラス間の関連



図Ⅳ－５　各クラスの継承関係

(7) クラスごとのメンバ関数抽出

各クラスごとに、どのようなメソッドが必要かを考え、メンバ関数として定義する。

イ　背景クラス

・ 空白の段が何段かを調べる。
・ ある一段がいっぱいかを調べる。
・ 枠内のある位置に積み重なったブロックの破片があるかを調べる。
・ 枠内の表示を消す。
・ 一段を消し、その段より上の部分を一段下に移動させる。
・ ブロックのモードを獲得する。
・ 枠の縦横の大きさを得る。
・ 枠内のある位置に積み重なったブロックの色を得る。
・ 背景を全て表示する。
・ 「↓」キーでブロックを下に落とす様子を表示する。
・ 枠を表示する。

・ ゲーム終了の表示をする。
・ タイトルと説明を表示する。
・ ブロックのモードを設定する。
・ 枠の位置を指定する。
・ 枠内のある位置に置くブロックの色を指定する。
・ タイトルと説明を表示する位置を指定する。
・ 入力キー待ちをする。

□ ブロッククラス

このクラスには、公開する関数群と公開しない関数群を設ける。

（公開メンバ関数）

・ ブロックが障害物にぶつかったかどうかを調べる。
・ ブロックの得点を表示する。
・ ブロックを１段づつ落下させる。
・ ブロックを床まで一気に落下させる。
・ 床まで一気に落下させることを示すフラグの状態を獲得する。
・ ブロックを左に移動する。
・ ブロックを移動し、得点を表示する。
・ 新しいブロックを作成する。
・ ブロックを背景の上に置く。
・ ブロックを表示する。
・ ブロックを右に移動する。
・ ブロックを構成する各四角形の位置を設定する。
・ ブロックが落下することを示すフラグを設定する。
・ ブロックが回転できるかを調べ、回転させる。
・ ゲームオーバの状態かを調べる。
・ 得点と最高得点を上げる。

（非公開メンバ関数）

・ ブロックを回転できるかを調べる。
・ ブロックの左上の位置から右下の位置を調べる。
・ ブロックの状態を調べ、得点を増やす。
・ １段にブロックが全くないか否かを調べる。
・ ブロックを消去する。
・ ブロックの色を獲得する。
・ ブロックの端点の位置を獲得する。
・ ブロックのパターンを獲得する。
・ ブロックの位相を獲得する。
・ ブロックのタイプを獲得する。
・ ブロックのパターンを初期化する。
・ ブロックのパターンを作成する。

・ ブロックを移動する。
・ ブロックを回転する。
・ ブロックの位相を設定する。
・ ブロックの何個目で新しいステージに代わるかを設定する。
・ ブロックの位置をシフトする。

ハ ブロック原形クラス
・ ブロックの原形を獲得する。
・ ブロック原形のX方向の大きさを知る。
・ ブロック原形のY方向の大きさを知る。
・ ブロックのタイプを知る。
・ タイプからブロックの原形を作成する。

ニ 得点クラス
・ 得点をクリアする。
・ 得点の情報を獲得する。
・ 得点を初期化する。
・ 得点を表示する。
・ 得点を設定する。
・ 得点を表示する位置を設定する。
・ 得点を上げる。

ホ 最高点クラス
・ 最高点をファイルから読み込む。
・ 最高点を表示する。
・ 最高点をファイルに書き出す。
・ 最高点を表示する位置を設定する。

ヘ ステージクラス
このクラスには、公開する関数群と公開しない関数群を設ける。
(公開メンバ関数)
・そのステージでブロックが落ち始める位置を獲得する。
・そのステージでのブロックの落下速度を獲得する。
・現在のステージ数を獲得する。
・新しいステージを始める。
・そのステージでのブロックの残り数を表示する。
・現在のステージ数を表示する。
・ステージの情報を表示する位置を指定する。
・そのステージでのブロックの落下速度を指定する。
・ブロックの数を増やす。

（非公開メンバ関数）
・ 現在何個目のブロックかを獲得する。
・ そのステージにおけるブロックのデフォルトの速度を獲得する。
・ 残りのブロック数を獲得する。
・ 現在のブロック数を設定する。
・ そのステージにおけるブロックのデフォルトの速度を設定する。

ト 基本関数ラップクラス
・ printf関数（表示関数）をラップする。
・ delay関数（遅延関数）をラップする。
・ getch関数（文字読み込み関数）をラップする。
・ gotoxy関数（カーソル移動関数）をラップする。
・ kbhit関数（キーボード入力検知関数）をラップする。
・ screenheight関数（画面のＹ方向の長さを知る関数）をラップする。
・ screenwidth関数（画面のＸ方向の長さを知る関数）をラップする。
・ sprintf関数（フォーマット付き変換関数）をラップする。
・ textcolor関数（文字色設定関数）をラップする。

チ 画面表示クラス
・ キーボード入力検知をする。
・ 画面をクリアする。
・ キー入力データを獲得する。
・ カーソルのＸ座標を獲得する。
・ カーソルのＹ座標を獲得する。
・ printf関数（表示関数）をラップする。
・ カーソルの位置を１増やす。
・ 処理を一旦停止する。
・ LONG型の数を表示する。
・ INTEGER型の数を表示する。
・ INTEGER型の数を桁数指定で表示する。
・ 文字列を表示する。
・ 長方形を表示する。
・ 表示する色を設定する。
・ キー入力データを強制的にシミュレートする。
・ カーソルの相対座標のオフセット値を設定する。
・ カーソルの位置を相対座標で設定する。
・ カーソルの位置を絶対座標で設定する。

リ ファイル入出力クラス
・ ファイルからデータを読み込む。
・ データの長さを獲得する。

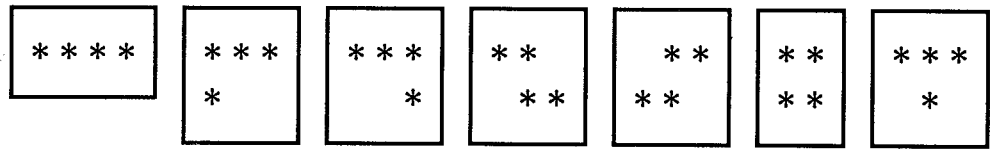
・ ファイルにデータを書き出す。

ﾇ 最高点保存クラス
・ 最高点をファイルから読み込む。
・ 最高点をファイルに書き出す。
・ ファイルの名前を指定する。

(8) クラスごとのデータメンバ抽出
各クラスごとに、どのような内部データが必要かを考え、データメンバとして定義する。

ｲ 背景クラス
・ 背景の各位置ごとの色情報のマトリックス
・ 背景枠（フレーム）の左上のＸ座標
・ 背景枠（フレーム）の左上のＹ座標
・ 背景枠（フレーム）の右下のＸ座標
・ 背景枠（フレーム）の右下のＹ座標
・ タイトルを表示する部分のＸ座標
・ タイトルを表示する部分のＹ座標
・ ブロックの一気落下の指定を示すフラグ

ﾛ ブロッククラス
・ ブロックのパターンを格納するためのマトリックス
・ ブロックの左上のＸ座標
・ ブロックの左上のＹ座標
・ ブロックの右下のＸ座標
・ ブロックの右下のＹ座標
・ ブロックの色
・ ブロックの回転角度（０、９０、１８０、２７０度）
・ ブロックのタイプ



・ 背景クラスとの関係
・ ブロック原形クラスとの関係
・ ステージクラスとの関係
・ 得点クラスとの関係
・ 最高点クラスとの関係

ハ　ブロック原形クラス
・ 原形ブロックの総数
・ ブロック原形のパターンを格納するためのマトリックス
・ ブロック原形のＸ方向の大きさ
・ ブロック原形のＹ方向の大きさ

ニ　得点クラス
・ 現在の得点
・ 現在の得点を表示するＸ座標
・ 現在の得点を表示するＹ座標

ホ　最高点クラス
・ 最高点保存クラスとの関係

ヘ　ステージクラス
・ そのステージにおけるブロックの落下速度
・ そのステージにおけるブロックのデフォルトの速度
・ 何個目のブロックか
・ ブロックの何個目で新しいステージになるか
・ ステージを表示するＸ座標
・ ステージを表示するＹ座標

ト　基本関数ラップクラス
　このクラスには、データメンバは存在しない

チ　画面表示クラス
・ キー入力データを強制的にシミュレートするか否かのフラグ
・ キー入力データを強制的にシミュレートするときのデータ
・ キーボード入力データ
・ 現在のカーソル位置
・ 画面のＸ方向の大きさ
・ 画面のＹ方向の大きさ
・ 画面のカーソル移動最大位置
・ カーソルのＸ方向のオフセット値
・ カーソルのＹ方向のオフセット値
・ 数値を変換するための作業領域

リ　ファイル入出力クラス
・ ファイルの名前
・ ファイル構造体へのポインタ
・ 現在のデータの長さ

・ ファイルをオープンするタイプ（読み込み、書き出し）

ｽ 最高点保存クラス
・ 最高点を格納するファイルの名前

(9) ヘッダファイルの作成

各クラスのメンバ関数とデータメンバの概要が決まったら、それらに名前をつけ、クラスを定義するヘッダとしてまとめる。そして、それらをヘッダファイルにする。

また、ヘッダファイルを一括して読み込むファイルを作成する。これは、ソース上で複数のヘッダファイルを記述する手間を省くためのものである。

更に、定数部分等をまとめたヘッダファイルも同時に作成する。

ｲ 背景クラス・ヘッダファイル

```
/*************************************************************************/
/*  CBACK.H                                                              */
/*     HEADER FOR CBACK CLASS                                            */
/*     CBACK CLASS IS A DERIVED CLASS FROM CSCREEN                       */
/*     THIS IS FOR MANAGING BACKGROUND OF BLOCK GAME                     */
/*************************************************************************/
#if !defined(__CBACK_H)
#define __CBACK_H

#include "BLOCKCOM.H"
#include "CSCREEN.H"

class CBack:public CScreen
    {
    private:
      int  *iMap;                 /* MAP DATA (COLOR DATA)          */
      int  iFrameX1;              /* FRAME X POSITION 1             */
      int  iFrameY1;              /* FRAME Y POSITION 1             */
      int  iFrameX2;              /* FRAME X POSITION 2             */
      int  iFrameY2;              /* FRAME Y POSITION 2             */
      int  iTitleX;               /* TITLE X POSITION               */
      int  iTitleY;               /* TITLE Y POSITION               */
      BOOL  bFallBlockSpecified;  /* FALL BLOCK OR NOT              */
    public:
      CBack(int iX1, int iY1, int iX2, int iY2);
                                  /* CONSTRUCTOR                    */
      int  CheckBlankLine(void);  /* CHECK THE NUMBER OF BLANK LINE */
```

```
    BOOL CheckLineFull(int iY); /* CHECK THE LINE IS FULL          */
    BOOL CheckMap(int iY, int iX);
                                /* CHECK MAP DATA EXIST            */
    void ClearFrame(void);      /* CLEAR FRAME                     */
    void DeleteLine(int iY);    /* DELETE 1 LINE                   */
    EDGE GetEdge(void);         /* GET EDGE POINTER                */
    BOOL GetFallBlock(void);    /* GET FALL BLOCK MODE             */
    EDGE GetFrameSize(void);    /* GET FRAME SIZE (X2, Y2)         */
    int  GetMapColor(int iY, int iX);
                                /* GET MAP COLOR DATA              */
    void PrintBack(void);       /* PRINT BACK DATA                 */
    void PrintFallDown(int iSpeed);
                                /* PRINT FALL DOWN                 */
    void PrintFrame(void);      /* PRINT FRAME                     */
    void PrintGameOver(void);   /* PRINT GAME OVER SCREEN          */
    void PrintTitle(void);      /* PRINT TITLE                     */
    void SetFallBlock(BOOL bFall);
                                /* SET FALL BLOCK MODE             */
    void SetFrame(int iX1, int iY1, int iX2, int iY2);
                                /* SET FRAME DATA                  */
    void SetMap(int iY, int iX, int iColor);
                                /* SET MAP DATA                    */
    void SetTitleDisplayPos(int iX, int iY);
                                /* SET TITLE DISPLAY POSITION      */
    void WaitKey(int iSpeed);   /* WAIT KEY INPUT                  */
};

#endif /* __CBACK_H */
```

□ ブロッククラス・ヘッダファイル

```
/*************************************************************************/
/*  CBLOCK.H                                                             */
/*     HEADER FOR CBLOCK CLASS                                           */
/*     CBLOCK CLASS IS A DERIVED CLASS FROM CSCREEN                      */
/*     THIS IS FOR MANAGING EACH BLOCK DATA                              */
/*************************************************************************/
#if !defined(__CBLOCK_H)
#define __CBLOCK_H

#include "BLOCKCOM.H"
#include "CSCREEN.H"
#include "CBACK.H"
#include "CSHAPE.H"
#include "CSCORE.H"
#include "CHISCORE.H"
#include "CSTAGE.H"

class CBlock: public CScreen
    {
    private:
      BOOL bPattern[4][4];        /* PATTERN AREA                  */
      int  iBlockX1;              /* BLOCK X POSITION 1            */
      int  iBlockY1;              /* BLOCK Y POSITION 1            */
      int  iBlockX2;              /* BLOCK X POSITION 2            */
      int  iBlockY2;              /* BLOCK Y POSITION 2            */
      int  iBlockColor;           /* COLOR OF BLOCK (0 to 7)       */
      int  iBlockPhase;           /* DIRECTION OF BLOCK (0 to 3)   */
      /*              0:0deg  1:90deg  2:180deg  3:270deg          */
      int  iBlockType;            /* TYPE OF BLOCK (0 to 5)        */
      /*              0:####  1:###  2:###  3:##   4: ##  5:##     */
      /*                      #       #     ##    ##     ##       */
      CBack *back;                /* RELATION TO BACK CLASS        */
      CShape *shape;              /* RELATION TO SHAPE CLASS       */
      CStage *stage;              /* RELATION TO STAGE CLASS       */
      CScore *score;              /* RELATION TO SCORE CLASS       */
      CHighScore *highscore;      /* RELATION TO HIGHSCORE CLASS   */
      int  AnswerRollBlock(void);
                                  /* ANSWER IF ROLL BLOCK HIT OR NOT*/
      EDGE CalcBottom(int iX, int iY, int iType, int iPhase);
```

```
                              /* CALCULATE EDGE FROM TOP        */
    void CheckAndUpScore(void);
                              /* CHECK BLOCK STATE AND UP SCORE */
    int  CheckBlankLine(void); /* CHECK THE NUMBER OF BLANK LINE */
    void EraseBlock(void);    /* ERASE BLOCK                    */
    int  GetColor(void);      /* GET BLOCK COLOR                */
    EDGE GetEdge(void);       /* GET EDGE POINTER               */
    BOOL GetPattern(int iY, int iX);
                              /* GET PATTERN DATA               */
    int  GetPhase(void);      /* GET BLOCK PHASE                */
    int  GetType(void);       /* GET BLOCK TYPE                 */
    void InitPattern(void);   /* INITIALIZE PATTERN             */
    void MakePattern(void);
                              /* MAKE PATTERN                   */
    void MoveBlock(int iDir);
                              /* MOVE BLOCK TO DIRECTION        */
    void RollBlock(void);
                              /* ROLL BLOCK TO RIGHT            */
    void RollPhase(void);     /* ROLL PHASE                     */
    void SetBlock(EDGE edge); /* SET BLOCK                      */
    void SetBlockStageUnit(int iData);
                              /* SET BLOCK STAGE UNIT           */
    void ShiftBlock(int iData); /* SHIFT BLOCK POSITION         */
  public:
    CBlock(CBack *pBack, CShape *pShape, CStage *pStage,
                          CScore *pScore, CHighScore *pHighScore);
                              /* CONSTRUCTOR                    */
    BOOL CheckHitBlock(int iDir, int iLength);
                              /* CHECK IF BLOCK HIT OR NOT      */
    void DisplayScores(void);
                              /* DISPLAY SCORE AND HIGHSCORE    */
    void DownBlock(void);     /* DOWN BLOCK                     */
    void FallBlock(void);     /* FALL BLOCK                     */
    BOOL GetFallBlock(void);  /* GET FALL BLOCK FLAG            */
    void LeftBlock(void);     /* MOVE BLOCK TO LEFT             */
    void MoveAndPrintBlock(int iKeyType);
                              /* MOVE BLOCK AND PRINT DATA      */
    void PlaceAndNewBlock(void);/* PLACE AND MAKE NEW BLOCK     */
    void PlaceBlock(void);
                              /* PLACE BLOCK ON BACK            */
    void PrintBlock(void);
```

```
                                /* PRINT BLOCK                    */
    void RightBlock(void);      /* MOVE BLOCK TO RIGHT            */
    void SetEachPosition(void); /* SET EACH RECTANGLE POSITION    */
    void SetFallBlock(BOOL iFall);
                                /* SET BLOCK FALL FLAG            */
    void TestAndRollBlock(void);/* TEST AND ROLL BLOCK            */
    void TestGameOver(void);    /* TEST GAME OVER                 */
    void UpScorePair(LONG lData);
                                /* UP SCORE AND HIGHSCORE         */
};

#endif /* __CBLOCK_H */
```

ハ　ブロック原形クラス・ヘッダファイル

```
/*****************************************************************************/
/*  CSHAPE.H                                                                 */
/*     HEADER FOR CSHAPE CLASS                                               */
/*     CSHAPE CLASS IS A BASE CLASS                                          */
/*     THIS IS FOR DEFINING THE SHAPE OF EACH BLOCKS                         */
/*     (*) MAXIMUM ENTRY NO IS 20                                            */
/*****************************************************************************/
#if !defined(__CSHAPE_H)
#define __CSHAPE_H

#include "BLOCKCOM.H"
#include "CWRAP.H"

class CShape: public CWrap
    {
    private:
      int iEntryNo;              /* SHAPE ENTRY TOTAL NO          */
      struct stShape
          {
          BOOL bPattern[4][4];
          int  iSizeX;
          int  iSizeY;
          } *pShape[20];         /* POINTER TO SHAPE STRUCT       */
    public:
      CShape(int iShapeNo);      /* CONSTRUCTOR                   */
      ~CShape(void);             /* DESTRUCTOR                    */
      BOOL GetPattern(int iType, int iY, int iX);
                                 /* GET PATTERN DATA              */
      int  GetSizeX(int iType);  /* GET X SIZE                    */
      int  GetSizeY(int iType);  /* GET Y SIZE                    */
      int  GetTypeNo(void);      /* GET TYPE NO                   */
      void SetPattern(int iType, char *sBitPattern);
                                 /* SET PATTERN DATA TO MATRIX    */
    };

#endif /* __CSHAPE_H */
```

ニ　得点クラス・ヘッダファイル

```
/*****************************************************************************/
/*  CSCORE.H                                                                 */
/*     HEADER FOR CSCORE CLASS                                               */
/*     CSCORE CLASS IS A DERIVED CLASS FROM CSCREEN                          */
/*     THIS IS FOR MANAGING SCORE DATA                                       */
/*****************************************************************************/
#if !defined(__CSCORE_H)
#define __CSCORE_H

#include "BLOCKCOM.H"
#include "CSCREEN.H"

class CScore:public CScreen
    {
    protected:
      LONG lScore;              /* CURRENT SCORE                 */
      int  iScoreX;             /* SCORE POSITION                */
      int  iScoreY;             /* SCORE POSITION                */
    public:
      void ClearScore(void);    /* CLEAR CURRENT SCORE           */
      LONG GetScore(void);      /* GET CURRENT SCORE             */
      void InitScore(int iX,int iY);
                                /* INITIALIZER                   */
      void PrintScore(void);    /* DISPLAY CURRENT SCORE         */
      void SetScore(LONG lData); /* SET CURRENT SCORE            */
      void SetScorePos(int iX,int iY);
                                /* SET CURRENT SCORE POSTION     */
      LONG UpScore(LONG lIinc); /* INCREASE SCORE                */
    };

#endif /* __CSCORE_H */
```

ホ　最高点クラス・ヘッダファイル

```
/*************************************************************************/
/*  CHISCORE.H                                                           */
/*     HEADER FOR CHIGHSCORE CLASS                                       */
/*     CHIGHSCORE CLASS IS A DERIVED CLASS FROM CSCORE                   */
/*     THIS IS FOR MANAGING HIGH SCORE DATA                              */
/*************************************************************************/
#if !defined(__CHIGHSCORE_H)
#define __CHIGHSCORE_H

#include "BLOCKCOM.H"
#include "CSCORE.H"
#include "CKEEP.H"

class CHighScore:public CScore
    {
    private:
      CKeep *keep;                /* RELATION TO KEEP              */
    public:
      CHighScore(void);           /* CONSTRUCTOR                   */
       CHighScore(void);          /* DESTRUCTOR                    */
      void LoadScore(void);       /* LOAD HIGH SCORE               */
      void PrintScore(void);      /* DISPLAY HIGH SCORE            */
      void SaveScore(void);       /* SAVE HIGH SCORE               */
      void SetScorePos(int iX,int iY);
                                  /* SET POSITION OF HIGH SCORE    */
    };

#endif /* __CHIGHSCORE_H */
```

ヘ ステージクラス・ヘッダファイル

```
/*****************************************************************************/
/*  CSTAGE.H                                                                 */
/*     HEADER FOR CSTAGE CLASS                                               */
/*     CSTAGE CLASS IS A DERIVED CLASS FROM CSCREEN                          */
/*     THIS IS FOR MANAGING STAGE INFORMATION                                */
/*****************************************************************************/
#if !defined(__CSTAGE_H)
#define __CSTAGE_H

#include "BLOCKCOM.H"
#include "CSCREEN.H"

class CStage:public CScreen
    {
    private:
      int  iSpeed;                /* SPEED OF BLOCK                 */
      int  iDefaultSpeed;         /* DEFAULT SPEED OF BLOCK         */
      int  iBlockNo;              /* BLOCK COUNTER                  */
      int  iBlockStageUnit;       /* STAGE UNIT FOR NEW BLOCK       */
        int  iDisplayX;             /* X POS OF DISPLAYING STAGE      */
      int  iDisplayY;             /* Y POS OF DISPLAYING STAGE      */
      int  GetBlockNo(void);      /* GET CURRENT BLOCK NO           */
      int  GetDefaultSpeed(void); /* GET DEFAULT SPEED              */
      int  GetRemainBlock(void);  /* GET REMAINING BLOCK NO         */
      void SetBlockNo(int iData); /* SET BLOCK NO                   */
      void SetBlockStageUnit(int iData);
                                  /* SET BLOCK UNIT FOR NEW STAGE   */
    public:
      CStage(int iStartSpeed,int iBlockInStage);
                                  /* CONSTRUCTOR                    */
      int  GetFallPos(void);      /* GET BLOCK FALL POSITION        */
      int  GetSpeed(void);        /* GET SPEED                      */
      int  GetStage(void);        /* GET STAGE NO                   */
      void NewStage(void);        /* NEW STAGE                      */
      void PrintRemainBlock(void);/* PRINT REMAINING BLOCK NO       */
      void PrintStage(void);      /* PRINT STAGE NO                 */
      void SetDisplayPosition(int iX,int iY);
                                  /* SET DISPLAY POSITION           */
      void SetSpeed(int iData);   /* SET SPEED                      */
      BOOL UpBlockNo(void);       /* UP BLOCK NO                    */
    };

#endif /* __CSTAGE_H */
```

ト　基本関数ラップクラス・ヘッダファイル

```
/*************************************************************************/
/*  CWRAP.H                                                              */
/*     HEADER FOR CWRAP CLASS                                            */
/*     CWRAP CLASS IS A BASE CLASS                                       */
/*     FUNCTIONS DEPEND ON COMPILER                                      */
/*************************************************************************/
#if !defined(__CWRAP_H)
#define __CWRAP_H

#include "BLOCKCOM.H"

class CWrap
    {
    public:
      void Clrscr(void);
      int  Cprintf(const char *s) {return cprintf(s);}
      int  Cprintf(const char *s,char *s2) {return cprintf(s,s2);}
      int  Cprintf(const char *s,int i) {return cprintf(s,i);}
      int  Cprintf(const char *s,LONG l) {return cprintf(s,l);}
      void Delay(unsigned i);
      int  Getch(void) {return getch();}
      void Gotoxy(int x,int y);
      int  Kbhit(void) {return kbhit();}
      int  ScreenHeight(void);
      int  ScreenWidth(void);
      int  Sprintf(char *b,const char *s) {return sprintf(b,s);}
      int  Sprintf(char *b,const char *s,char *s2) {return sprintf(b,s,s2);}
      int  Sprintf(char *b,const char *s,int i) {return sprintf(b,s,i);}
      int  Sprintf(char *b,const char *s,LONG l) {return sprintf(b,s,l);}
      void Textcolor(int i);
    };

#endif  /* __CWRAP_H */
```

チ　画面表示クラス・ヘッダファイル

```
/*************************************************************************/
/*  CSCREEN.H                                                            */
/*     HEADER FOR CSCREEN CLASS                                          */
/*     CSCREEN CLASS IS A DERIVED CLASS FROM CWRAP                       */
/*     THIS IS FOR:                                                      */
/*     (1) KEYBOARD INPUT                                                */
/*     (2) DISPLAY OUTPUT                                                */
/*     (3) PAUSE                                                         */
/*     (4) CURSOR MANAGEMENT                                             */
/*************************************************************************/
#if !defined(__CSCREEN_H)
#define __CSCREEN_H

#include "BLOCKCOM.H"
#include "CWRAP.H"

class CScreen:public CWrap
    {
    private:
      BOOL bKeyForce;             /* INPUT KEY FORCE FLAG (T:FORCE) */
      int  iKeyForce;             /* INPUT KEY FORCE DATA           */
      int  iKey;                  /* INPUT KEY DATA                 */
      int  iCurPos;               /* CURSOR X+Y*YLEN                */
      int  iSizeX;                /* X LENGTH OF SCREEN             */
      int  iSizeY;                /* Y LENGTH OF SCREEN             */
      int  iSizeMax;              /* MAX LENGTH OF SCREEN           */
      int  iCurXoffset;           /* CURSOR X OFFSET                */
      int  iCurYoffset;           /* CURSOR Y OFFSET                */
      char sNumberWork[20];       /* STRING FOR NUMBER TRANSLATE    */
    public:
      CScreen(void);              /* CONSTRUCTOR                    */
      int  CheckKey(void);        /* CHECK KEY HIT                  */
      void ClearScreen(void);     /* CLEAR ALL SCREEN               */
      int  GetKey(void);          /* GET KEY DATA                   */
      int  GetX(void);            /* GET CURSOR X POSITION          */
      int  GetY(void);            /* GET CURSOR Y POSITION          */
      void IncCursor(int iData);  /* INCREASE CURSOR POSITION       */
      void Pause(void);           /* PAUSE ONLY                     */
      void PrintNumber(LONG l);   /* PRINT NUMBER (LONG)            */
      void PrintNumber(int i);    /* PRINT NUMBER (INT)             */
      void PrintNumber(int i, int iCol);
                                  /* PRINT NUMBER (INT)             */
      void PrintText(char *sText);/* PRINT STRING                   */
      void Rectangle(int iX1, int iY1, int iX2, int iY2);
                                  /* DRAW RECTANGLE                 */
      void SetColor(int iData);   /* SET COLOR                      */
      void SetKeyForce(int iKey); /* SET KEY FORCE FOR GETKEY       */
      void SetOffset(int iX, int iY);
                                  /* SET CURSOR POSITION OFFSET     */
      void SetRelXY(int iX, int iY);
                                  /* SET RELATIVE  X,Y (0,0)-       */
      void SetXY(int iX, int iY); /* SET CURSOR ON X,Y (0,0)-       */
    };

#endif /* __CSCREEN_H */
```

リ　ファイル入出力クラス・ヘッダファイル

```
/*****************************************************************************/
/*  CDATUM.H                                                                 */
/*     HEADER FOR CDATUM CLASS                                               */
/*     CDATUM CLASS IS A DERIVED CLASS FROM CWRAP                            */
/*     THIS IS FOR MANAGING FILE INPUT/OUTPUT                                */
/*****************************************************************************/
#if !defined(__CDATUM_H)
#define __CDATUM_H

#include "BLOCKCOM.H"
#include "CWRAP.H"

class CDatum:public CWrap
    {
    private:
      char sFileName[256];        /* FILE NAME                          */
      FILE *fileStatic;           /* FILE POINTER                       */
      int  iLength;               /* CURRENT DATA LENGTH                */
      BOOL bOpenType;             /* OPEN TYPE (OPN_READ/OPN_WRITE)     */
    public:
      CDatum(char *sName,BOOL bType);
                                  /* DATA OPEN FILE                     */
      ~CDatum(void);              /* DATA CLOSE FILE                    */
      void GetData(char *sData);  /* GET DATA FROM FILE                 */
      int  GetLength(void);       /* GET DATA LENGTH                    */
      int  PutData(char *sData);  /* PUT DATA TO FILE                   */
    };

#endif /* __CDATUM_H */
```

ヌ　最高点保存クラス・ヘッダファイル

```
/*****************************************************************************/
/*  CKEEP.H                                                                  */
/*     HEADER FOR CKEEP CLASS                                                */
/*     CKEEP CLASS IS A DERIVED CLASS FROM CWRAP                             */
/*     THIS IS FOR KEEPING HIGH SCORE INTO FILE                              */
/*****************************************************************************/
#if !defined(__CKEEP_H)
#define __CKEEP_H

#include "BLOCKCOM.H"
#include "CWRAP.H"
#include "CDATUM.H"

class CKeep:public CWrap
    {
    private:
      char  sHighScore[10];       /* HIGH SCORE FILE NAME          */
    public:
      LONG  PullScore(void);      /* PULL HIGH SCORE FROM FILE     */
      void  PushScore(LONG lScore);
                                  /* PUSH HIGH SCORE TO FILE       */
      void  SetName(char *sName); /* SET FILE NAME                 */
    };

#endif  /* __CKEEP_H */
```

ル　総ヘッダファイル

```
#include "BLOCKCOM.H"
#include "CSCORE.H"
#include "CHISCORE.H"
#include "CBLOCK.H"
#include "CKEEP.H"
#include "CSHAPE.H"
#include "CSTAGE.H"
```

ヲ　共通ヘッダファイル

```
/*****************************************************************************/
/*  BLOCKCOM.H                                                               */
/*     HEADER FOR BLOCK COMMON                                               */
/*     IT INCLUDES #DEFINE DATA                                              */
/*****************************************************************************/
#if !defined(__BLOCKCOM_H)
#define __BLOCKCOM_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <dos.h>

#define      BOOL        int
#define      LONG        long

#define      FALSE       0
#define      TRUE        1
#define      SPECIAL     -1

#define      CHK_NOW     0
#define      CHK_DOWN    1
#define      CHK_LEFT    2
#define      CHK_RIGHT   3
#define      CHK_ROLL    4

#define      CLR_BLUE          0x01
#define      CLR_CYAN          0x03
#define      CLR_GREEN         0x02
#define      CLR_LIGHTGRAY     0x07
#define      CLR_RED           0x04
#define      CLR_WHITE         0x0f
#define      CLR_YELLOW        0x0e

#define      KEY_DOWN          0x50
#define      KEY_ESC           0x1b
#define      KEY_LEFT          0x4b
#define      KEY_RIGHT         0x4d
#define      KEY_SPACE         ' '
#define      MES_NEWGAME       0x101
#define      MES_FALLBLOCK     0x102
#define      MES_NEWBLOCK      0x103
#define      MES_NEWSTAGE      0x104
#define      MES_RESTARTGAME   0x105
#define      MES_PAUSE         'p'
#define      MES_REDRAW        'r'
#define      MES_SPEEDDOWN     'D'
#define      MES_SPEEDUP       'd'

#define      OPN_READ    1
#define      OPN_WRITE   2

struct EDGE
    {
    int iX1;                          /* X DATA 1                        */
    int iY1;                          /* Y DATA 1                        */
    int iX2;                          /* X DATA 2                        */
    int iY2;                          /* Y DATA 2                        */
    };

#endif  /* __BLOCKCOM_H */
```

(10) クラス実装

ヘッダファイルが完成したら、各クラスの実際の動作を実装する。

イ 背景クラス

```
#include "CBACK.H"

CBack::CBack(int iX1=14,int iY1=1,int iX2=26,int iY2=23)
    {                                   /* CONSTRUCTOR                    */
    int  i,j;

    /* SET PARAMETER */
    if(iX1<11)
      iX1=11;
    if(iY1<0)
      iY1=0;
    if(iY2>ScreenHeight()-1)
      iY2=ScreenHeight()-1;
    iMap=(int *)calloc(sizeof(int),(iY2-iY1+2)*(iX2-iX1+1));
    iFrameX1=iX1;
    iFrameY1=iY1;
    iFrameX2=iX2;
    iFrameY2=iY2;
    bFallBlockSpecified=FALSE;
    SetColor(CLR_WHITE);
    SetOffset(iFrameX1,iFrameY1);
    SetFrame(iFrameX1,iFrameY1,iFrameX2,iFrameY2);
    for(j=0;j<=iFrameY2-iFrameY1;j++)
      {
      for(i=0;i<=iFrameX2-iFrameX1;i++)
          SetMap(j,i,FALSE);
      }
    j=iFrameY2-iFrameY1+1;
    for(i=0;i<=iFrameX2-iFrameX1;i++)
      SetMap(j,i,TRUE);
    ClearScreen();
    }

int  CBack::CheckBlankLine(void)  /* CHECK THE NUMBER OF BLANK LINE   */
    {
    int  i,j,k;

    for(j=0;j<=iFrameY2-iFrameY1;j++)
      {
      k=0;
      for(i=0;i<=iFrameX2-iFrameX1;i++)
          {
          k+=GetMapColor(j,i);
          }
      if(k>0)
          return(j);
      }
    return(iFrameY2-iFrameY1);
    }

BOOL CBack::CheckMap(int iY,int iX)  /* CHECK MAP DATA                   */
    {
    if(0!=GetMapColor(iY,iX))
      return TRUE;
```

```
    else
      return FALSE;
    }

BOOL CBack::CheckLineFull(int iY)  /* CHECK THE LINE IS FULL          */
    {
    int  i;

    if(iY>=iFrameY2-iFrameY1+1)
      return FALSE;
    for(i=iFrameX1;i<=iFrameX2;i++)
      {
      if(0==GetMapColor(iY, i-iFrameX1))
          return FALSE;
      }
    return TRUE;
    }

void CBack::ClearFrame(void)
    {                       /* CLEAR FRAME                    */
    int i,j;

    SetColor(CLR_WHITE);
    for(j=iFrameY1;j<=iFrameY2;j++)
      for(i=iFrameX1;i<=iFrameX2;i++)
          {
          SetRelXY(i-iFrameX1, j-iFrameY1);
          SetMap(j-iFrameY1, i-iFrameX1, 0);
          PrintText("  ");
          }
    }

void CBack::DeleteLine(int iY)  /* DELETE 1 LINE                  */
    {
    int  i,j;

    for(j=iY-1;j>=0;j--)
      {
      for(i=iFrameX1;i<=iFrameX2;i++)
          SetMap(j+1, i-iFrameX1, GetMapColor(j, i-iFrameX1));
      }
    for(i=iFrameX1;i<=iFrameX2;i++)
      SetMap(0, i-iFrameX1, 0);
    }

EDGE CBack::GetEdge(void)            /* GET EDGE POINTER          */
    {
    EDGE edge;

    edge.iX1=iFrameX1;
    edge.iY1=iFrameY1;
    edge.iX2=iFrameX2;
    edge.iY2=iFrameY2;
    return edge;
    }

BOOL CBack::GetFallBlock(void)    /* GET FALL BLOCK MODE          */
    {
    return bFallBlockSpecified;
    }
```

```
EDGE CBack::GetFrameSize(void)    /* GET FRAME SIZE (X2, Y2)          */
    {
    EDGE edge;

    edge.iX1=0;
    edge.iY1=0;
    edge.iX2=iFrameX2-iFrameX1;
    edge.iY2=iFrameY2-iFrameY1;
    return edge;
    }

int CBack::GetMapColor(int iY, int iX) /* GET MAP COLOR DATA          */
    {
    return iMap[(iFrameX2-iFrameX1+1)*iY+iX];
    }

void CBack::PrintBack(void)
    {                       /* PRINT FRAME                      */
    int i, j;

    for(j=iFrameY1;j<=iFrameY2;j++)
      for(i=iFrameX1;i<=iFrameX2;i++)
          {
          SetRelXY(i-iFrameX1, j-iFrameY1);
          if(FALSE==GetMapColor(j-iFrameY1, i-iFrameX1))
              {
              SetColor(CLR_WHITE);
              PrintText("  ");
              }
          else
              {
              SetColor(GetMapColor(j-iFrameY1, i-iFrameX1));
              PrintText("□");
              }
          }
    }

void CBack::PrintFallDown(int iSpeed)/* PRINT FALL DOWN                */
    {
    int  i, j;
    EDGE edge;

    edge=GetFrameSize();
    Delay(iSpeed);
    i=CheckBlankLine();
    for(j=edge.iY2;j>=i;j--)
      {
      DeleteLine(edge.iY2);
      PrintBack();
      Delay(iSpeed);
      }
    }

void CBack::PrintFrame(void)  /* PRINT FRAME                      */
    {
    Rectangle(iFrameX1-1, iFrameY1-1, iFrameX2+1, iFrameY2+1);
    }

void CBack::PrintGameOver(void)  /* PRINT GAME OVER SCREEN          */
    {
    EDGE edge;
```

```
    SetColor(CLR_RED);
    edge=GetEdge();
    Rectangle(edge.iX1+(edge.iX2-edge.iX1)/2-5,
              edge.iY1+(edge.iY2-edge.iY1)/2-2,
              edge.iX1+(edge.iX2-edge.iX1)/2+5,
              edge.iY1+(edge.iY2-edge.iY1)/2+1);
    SetColor(CLR_YELLOW);
    SetXY(edge.iX1+(edge.iX2-edge.iX1)/2-5+1,edge.iY1+(edge.iY2-edge.iY1)/2-1);
    PrintText("    ゲーム終了    ");
    SetXY(edge.iX1+(edge.iX2-edge.iX1)/2-5+1,edge.iY1+(edge.iY2-edge.iY1)/2);
    PrintText("続けますか？(y/n)");
    SetColor(CLR_WHITE);
    }

void CBack::PrintTitle(void)  /* PRINT TITLE                    */
    {
    SetColor(CLR_BLUE);
    Rectangle(iTitleX,iTitleY,iTitleX+10,iTitleY+4);
    SetColor(CLR_CYAN);
    SetXY(iTitleX+2,iTitleY+1);
    PrintText("ブロックゲーム");
    SetColor(CLR_GREEN);
    SetXY(iTitleX+1,iTitleY+2);
    PrintText("Copyright(c) 1996");
    SetXY(iTitleX+2,iTitleY+3);
    PrintText("by A. Yamashita");
    SetColor(CLR_LIGHTGRAY);
    SetXY(iTitleX+1,iTitleY+5);
    PrintText("←        左へ移動");
    SetXY(iTitleX+1,iTitleY+6);
    PrintText("→        右へ移動");
    SetXY(iTitleX+1,iTitleY+7);
    PrintText("↓        下へ落とす");
    SetXY(iTitleX,iTitleY+8);
    PrintText(" [スペース] 回転");
    SetColor(CLR_WHITE);
    }

void CBack::SetFallBlock(BOOL bFall)/* SET FALL BLOCK MODE        */
    {
    bFallBlockSpecified=bFall;
    }

void CBack::SetFrame(int iX1,int iY1,int iX2,int iY2)
    {                    /* SET FRAME DATA              */
    iFrameX1=iX1+1;
    iFrameY1=iY1+1;
    iFrameX2=iX2-1;
    iFrameY2=iY2-1;
    }

void CBack::SetMap(int iY,int iX,int iColor) /* SET MAP DATA        */
    {
    iMap[(iFrameX2-iFrameX1+1)*iY+iX]=iColor;
    }

void CBack::SetTitleDisplayPos(int iX,int iY)
                              /* SET TITLE DISPLAY POSITION   */
    {
    if(iX<0)
      iX=0;
```

```
    iTitleX=iX;
    iTitleY=iY;
    }

void CBack::WaitKey(int iSpeed)
    {
    int iRepeatCounter=0;
    /* LOOP UNTIL KEY HIT */
    while(FALSE==CheckKey())
      {
      if(GetFallBlock()==FALSE)
      Delay(iSpeed);
      if(iRepeatCounter==3)
      {
      SetKeyForce(MES_FALLBLOCK);
      iRepeatCounter=0;
      return;
      }
      else
      iRepeatCounter++;
      }
    return;
    }
```

□ ブロッククラス

```
#include "CBLOCK.H"

CBlock::CBlock(CBack *pBack, CShape *pShape, CStage *pStage,
                              CScore *pScore, CHighScore *pHighScore)
    {                             /* CONSTRUCTOR                    */
    EDGE edge;
    EDGE edgeBack;
    int iLoop;

    back=pBack;
    shape=pShape;
    stage=pStage;
    score=pScore;
    highscore=pHighScore;
    iBlockType=rand()%(shape->GetTypeNo());
    iBlockColor=rand()%7+1;
    iBlockPhase=0;
    edgeBack=back->GetFrameSize();
    if(edgeBack.iX2<=6)
      iBlockX1=0;
    else
      iBlockX1=edgeBack.iX2/2-1;
    iBlockY1=stage->GetFallPos();
    edge=CalcBottom(iBlockX1, iBlockY1, iBlockType, iBlockPhase);
    SetBlock(edge);
    MakePattern();
    iLoop=rand()%4;
    for(int k=0;k<iLoop;k++)
      {
      RollPhase();
      RollBlock();
      }
    }
```

```
int CBlock::AnswerRollBlock(void)
    {                                   /* ANSWER IF ROLL BLOCK HIT OR NOT */
    EDGE edge;
    EDGE edgeBack;

    edge=GetEdge();
    edgeBack=back->GetFrameSize();
    if(edge.iX2-edge.iX1<edge.iY2-edge.iY1)
      {
      if(edge.iX1+(edge.iY2-edge.iY1)>edgeBack.iX2)
          return(edge.iX1+(edge.iY2-edge.iY1)-edgeBack.iX2);
      else
          return(0);
      }
    else
      {
      if(edge.iY1+(edge.iX2-edge.iX1)>=edgeBack.iY2)
          return(edge.iY1+(edge.iX2-edge.iX1)-edgeBack.iY2);
      else
          return(0);
      }
    }

EDGE CBlock::CalcBottom(int iX,int iY,int iType,int iPhase)
    {
    EDGE edge;

    edge.iX1=iX;
    edge.iY1=iY;
    if((iPhase % 2)==1) /* PHASE IS 1 OR 3 */
      {
      edge.iX2=iX+shape->GetSizeY(iType);
      edge.iY2=iY+shape->GetSizeX(iType);
      }
    else
      {
      edge.iX2=iX+shape->GetSizeX(iType);
      edge.iY2=iY+shape->GetSizeY(iType);
      }
    return edge;
    }

void CBlock::CheckAndUpScore(void)
    {                                   /* CHECK BLOCK STATE AND UP SCORE */
    int  j;
    BOOL bFull[4];
    EDGE edge;
    int  iScore;

    edge=GetEdge();
    for(j=edge.iY1;j<=edge.iY2;j++)
      {
      if(TRUE==(back->CheckLineFull(j)))
          bFull[j-edge.iY1]=TRUE;
      else
          bFull[j-edge.iY1]=FALSE;
      }
    iScore=0;
    for(j=edge.iY1;j<=edge.iY2;j++)
      {
```

```
      if(bFull[j-edge.iY1]==TRUE)
          {
          back->DeleteLine(j);
          back->PrintBack();
          if(iScore==0)
              iScore=100*(stage->GetStage());
          else
              iScore=iScore*2;
          }
      }
    score->UpScore(iScore);
    }

BOOL CBlock::CheckHitBlock(int iDir,int iLength)
    {                              /* CHECK IF BLOCK HIT OR NOT      */
    EDGE edge;
    EDGE edgeBack;

    edge=GetEdge();
    edgeBack=back->GetFrameSize();
    switch(iDir)
      {
      case CHK_NOW:   /* NOW   */
          break;
      case CHK_DOWN:  /* DOWN  */
          if(edge.iY2>=edgeBack.iY2+iLength)
              return(TRUE);
          if(edge.iY2==edgeBack.iY2+iLength)
              return(SPECIAL);
          edge.iY1+=iLength;
          edge.iY2+=iLength;
          break;
      case CHK_LEFT:  /* LEFT  */
          if(edge.iX1<iLength)
              return(TRUE);
          edge.iX1-=iLength;
          edge.iX2-=iLength;
          break;
      case CHK_RIGHT: /* RIGHT */
          if(edge.iX2+iLength>edgeBack.iX2)
              return(TRUE);
          edge.iX1+=iLength;
          edge.iX2+=iLength;
          break;
      }
    for(int j=edge.iY2;j>=edge.iY1;j--)
      for(int i=edge.iX1;i<=edge.iX2;i++)
          {
          if(TRUE==GetPattern(j-edge.iY1,i-edge.iX1))
              { /* PATTERN EXIST */
              if(TRUE==(back->CheckMap(j,i)))
                  return TRUE;
              }
          }
    return FALSE;
    }

void CBlock::DisplayScores(void)
    {                   /* DISPLAY SCORE AND HIGHSCORE       */
    SetColor(CLR_WHITE);
    score->PrintScore();
```

```
    highscore->PrintScore();
    }


void CBlock::DownBlock(void)        /* DOWN BLOCK                          */
    {
    while(1==1)
      {
      if(FALSE!=CheckHitBlock(CHK_DOWN,1))
          {
          /* PLACE BLOCK AND UP SCORE */
          PlaceAndNewBlock();
          return;
          }
      /* DOWN BLOCK */
      MoveAndPrintBlock(KEY_DOWN);
      }
    }

void CBlock::EraseBlock(void) /* ERASE BLOCK                          */
    {
    int  i,j;

    SetColor(CLR_WHITE);
    for(j=iBlockY1;j<=iBlockY2;j++)
      {
      for(i=iBlockX1;i<=iBlockX2;i++)
          {
          if(FALSE==(back->CheckMap(j,i)))
              {
              back->SetRelXY(i,j);
              PrintText("  ");
              }
          }
      }
    }

void CBlock::FallBlock(void)
    {
    int bCheckResult=CheckHitBlock(CHK_DOWN,1);
    if(bCheckResult==FALSE)
      {
      /* DOWN BLOCK */
      SetFallBlock(FALSE);
      MoveAndPrintBlock(KEY_DOWN);
      }
    else if(bCheckResult==TRUE)
      {
      /* PLACE BLOCK AND UP SCORE */
      SetFallBlock(FALSE);
      PlaceAndNewBlock();
      }
    else /* SPECIAL */
      {
      if(GetFallBlock()==TRUE)
          {
          /* PLACE BLOCK AND UP SCORE */
          SetFallBlock(FALSE);
          PlaceAndNewBlock();
          }
      else
```

```
            /* GO TO NEXT MOVEMENT */
            SetFallBlock(TRUE);
        }
    }

int  CBlock::GetColor(void)          /* GET BLOCK COLOR                */
    {
    return iBlockColor;
    }

EDGE CBlock::GetEdge(void)           /* GET EDGE POINTER               */
    {
    EDGE edge;

    edge.iX1=iBlockX1;
    edge.iY1=iBlockY1;
    edge.iX2=iBlockX2;
    edge.iY2=iBlockY2;
    return edge;
    }

BOOL CBlock::GetFallBlock(void)     /* GET FALL BLOCK FLAG            */
    {
    return back->GetFallBlock();
    }

BOOL CBlock::GetPattern(int iY, int iX)
    {                               /* GET PATTERN DATA               */
    return bPattern[iY][iX];
    }

int  CBlock::GetPhase(void)          /* GET BLOCK PHASE                */
    {
    return iBlockPhase;
    }

int  CBlock::GetType(void)           /* GET BLCOK TYPE                 */
    {
    return iBlockType;
    }

void CBlock::InitPattern(void)
    {
    for(int j=0;j<4;j++)
      for(int i=0;i<4;i++)
          bPattern[j][i]=0;
    }

void CBlock::LeftBlock(void)
    {
    if(TRUE==CheckHitBlock(CHK_LEFT, 1))
      /* BLOCK IS ALREADY LEFT SIDE */
      return;
    /* MOVE BLOCK TO LEFT */
    MoveAndPrintBlock(KEY_LEFT);
    return;
    }

void CBlock::MakePattern(void)
    {
    InitPattern();
```

```
    for(int j=0;j<4;j++)
      for(int i=0;i<4;i++)
          bPattern[j][i]=shape->GetPattern(iBlockType,j,i);
    }

void CBlock::MoveAndPrintBlock(int iKeyType)
    {                              /* MOVE BLOCK AND PRINT DATA      */
    EraseBlock();
    MoveBlock(iKeyType);
    PrintBlock();
    score->PrintScore();
    highscore->PrintScore();
    }

void CBlock::MoveBlock(int iDir)/* MOVE BLOCK TO DIRECTION        */
    {
    EDGE edge;

    edge=back->GetEdge();
    switch(iDir)
      {
      case KEY_LEFT: /* LEFT  */
          if(iBlockX1>0)
              {
              iBlockX1--;
              iBlockX2--;
              }
          break;
      case KEY_RIGHT: /* RIGHT */
          if(iBlockX2<edge.iX2)
              {
              iBlockX1++;
              iBlockX2++;
              }
          break;
      case KEY_DOWN: /* DOWN */
          if(iBlockY2<edge.iY2)
              {
              iBlockY1++;
              iBlockY2++;
              }
          break;
      }
    }

void CBlock::PlaceAndNewBlock(void)
    {                                  /* PLACE AND MAKE NEW BLOCK       */
    PlaceBlock();
    CheckAndUpScore();
    UpScorePair((stage->GetStage())*10);
    back->SetKeyForce(MES_NEWBLOCK);
    }

void CBlock::PlaceBlock(void)
    {
    int  i,j;
    EDGE edge;

    edge=GetEdge();
    for(j=edge.iY1;j<=edge.iY2;j++)
      for(i=edge.iX1;i<=edge.iX2;i++)
```

```
            {
            if(TRUE==(GetPattern(j-edge.iY1, i-edge.iX1)))
                {
                back->SetMap(j, i, GetColor());
                back->SetRelXY(i, j);
                SetColor(GetColor());
                PrintText("□");
                SetColor(CLR_WHITE);
                }
            }
    }

void CBlock::PrintBlock(void) /* SET BLCOK                          */
    {
    int  i, j;
    EDGE edge;

    edge=back->GetEdge();
    SetColor(iBlockColor);
    for(j=iBlockY1;j<=iBlockY2;j++)
      {
      for(i=iBlockX1;i<=iBlockX2;i++)
            {
            back->SetRelXY(i, j);
            if(TRUE==GetPattern(j-iBlockY1, i-iBlockX1))
                PrintText("■");
            }
      }
    SetColor(CLR_WHITE);
    stage->PrintRemainBlock();
    }

void CBlock::RightBlock(void)
    {
    if(TRUE==CheckHitBlock(CHK_RIGHT, 1))
      /* BLOCK IS ALREADY RIGHT SIDE */
      return;
    /* MOVE BLOCK TO RIGHT */
    MoveAndPrintBlock(KEY_RIGHT);
    return;
    }

void CBlock::RollBlock(void)
    {
    int  i, j;
    BOOL bWork[4][4];
    EDGE edge;

    edge=CalcBottom(iBlockX1, iBlockY1, iBlockType, iBlockPhase);
    for(j=0;j<=iBlockY2-iBlockY1;j++)
      for(i=0;i<=iBlockX2-iBlockX1;i++)
            bWork[iBlockX2-iBlockX1-i][j]=bPattern[j][i];
    InitPattern();
    for(j=0;j<=iBlockX2-iBlockX1;j++)
      for(i=0;i<=iBlockY2-iBlockY1;i++)
            bPattern[j][i]=bWork[j][i];
    SetBlock(edge);
    }

void CBlock::RollPhase(void)
    {
```

```
    iBlockPhase=(iBlockPhase+1)%4;
    }

void CBlock::SetBlock(EDGE edge)    /* SET BLCOK                          */
    {
    iBlockX1=edge.iX1;
    iBlockY1=edge.iY1;
    iBlockX2=edge.iX2;
    iBlockY2=edge.iY2;
    }

void CBlock::SetEachPosition(void)
    {
    EDGE  edgeFrame;          /* FRAME POSITION                    */

    edgeFrame=back->GetEdge();
    back->SetTitleDisplayPos(edgeFrame.iX1-14, edgeFrame.iY1);
    stage->SetDisplayPosition(edgeFrame.iX2+5, edgeFrame.iY1+7);
    score->InitScore(edgeFrame.iX2+5, edgeFrame.iY1+3);
    highscore->SetScorePos(edgeFrame.iX2+5, edgeFrame.iY1+5);
    }

void CBlock::SetFallBlock(BOOL bFall)
    {
    back->SetFallBlock(bFall);
    }

void CBlock::ShiftBlock(int iData)  /* SHIFT BLOCK POSITION          */
    {
    iBlockX1-=iData;
    iBlockX2-=iData;
    }

void CBlock::TestAndRollBlock(void) /* TEST AND ROLL BLOCK           */
    {
    int iRollable=TRUE;
    int iShiftSize=AnswerRollBlock();
    if(iShiftSize>0)
      {
      /* HAVE TO MOVE BLOCK TO LEFT */
      if(TRUE!=(CheckHitBlock(CHK_LEFT, iShiftSize)))
      {
      /* CAN MOVE BLOCK TO LEFT */
      EraseBlock();
      ShiftBlock(iShiftSize);
      PrintBlock();
      }
      else
      /* CANNOT MOVE BLOCK TO LEFT */
      iRollable=FALSE;
      }
    if(iRollable==TRUE)
      {
      /* ROLL BLOCK */
      EraseBlock();
      RollPhase();
      RollBlock();
      PrintBlock();
      DisplayScores();
      }
    }
```

```
void CBlock::TestGameOver(void)     /* TEST GAME OVER                    */
    {
    if(TRUE==CheckHitBlock(CHK_NOW, 0))
      {
      /* CANNOT MOVE BLOCK */
      PlaceBlock();
      DisplayScores();
      back->PrintGameOver();
      /* WAIT USER KEY INPUT */
      while(FALSE==(back->CheckKey()))
          ;
      if('y'!=(back->GetKey()))
          /* GO TO FINISH */
          back->SetKeyForce(KEY_ESC);
      else
          /* RESTART NEW GAME */
          back->SetKeyForce(MES_RESTARTGAME);
      }
    else
      {
      PrintBlock();
      DisplayScores();
      }
    }

void CBlock::UpScorePair(LONG lData)
    {                          /* UP SCORE AND HIGHSCORE          */
    LONG l, lHigh;

    l=score->UpScore(lData);
    lHigh=highscore->GetScore();
    if(l>lHigh)
      highscore->SetScore(l);
    }
```

ハ　ブロック原形クラス

```
#include "CSHAPE.H"

CShape::CShape(int iShapeNo)
    {                                   /* CONSTRUCTOR                  */
    int i;

    iEntryNo=iShapeNo;
    for(i=0;i<iShapeNo;i++)
      {
      pShape[i]=(stShape *)malloc(sizeof(struct stShape));
      }
    }

CShape::~CShape(void)
    {                                   /* DESTRUCTOR                   */
    int i;

    for(i=0;i<iEntryNo;i++)
      {
      free(pShape[i]);
      }
    }

BOOL CShape::GetPattern(int iType,int iY,int iX)
    {                                   /* GET PATTERN DATA             */
    return pShape[iType]->bPattern[iY][iX];
    }

int CShape::GetSizeX(int iType)
    {                                   /* GET X SIZE                   */
    return pShape[iType]->iSizeX;
    }

int CShape::GetSizeY(int iType)
    {                                   /* GET Y SIZE                   */
    return pShape[iType]->iSizeY;
    }

int CShape::GetTypeNo(void)
    {                                   /* GET TYPE NO                  */
    return iEntryNo;
    }

void CShape::SetPattern(int iType,char *sBitPattern)
    {
    int i;                              /* GET PATTERN DATA             */
    for(i=0;i<16;i++)
      {
      pShape[iType]->bPattern[i/4][i%4]=0;
      }
    pShape[iType]->iSizeX=0;
    pShape[iType]->iSizeY=0;
    for(i=0;i<16;i++)
      {
      if(sBitPattern[i]=='¥0')
          break;
      if(sBitPattern[i]=='1')
          {
          pShape[iType]->bPattern[i/4][i%4]=1;
          if((i%4)>(pShape[iType]->iSizeX))
              pShape[iType]->iSizeX=i%4;
          if((i/4)>(pShape[iType]->iSizeY))
              pShape[iType]->iSizeY=i/4;
          }
      }
    }
```

ニ　得点クラス

```
#include "CSCORE.H"

void CScore::ClearScore(void)        /* CLEAR CURRENT SCORE            */
    {
    lScore=0;
    }

LONG CScore::GetScore(void)          /* GET CURRENT SCORE              */
    {
    return lScore;
    }

void CScore::InitScore(int iX, int iY)
    {
    ClearScore();
    SetScorePos(iX, iY);
    }

void CScore::PrintScore(void)        /* DISPLAY CURRENT SCORE          */
    {
    SetXY(iScoreX, iScoreY);
    PrintText("得点    :");
    PrintNumber(lScore);
    }

void CScore::SetScore(LONG lData)    /* SET CURRENT SCORE              */
    {
    lScore=lData;
    }

void CScore::SetScorePos(int iX, int iY)
    {             /* SET CURRENT SCORE POSTION      */
    if(iX>(ScreenWidth()/2-8))
      iX=ScreenWidth()/2-8;
    iScoreX=iX;
    iScoreY=iY;
    }

LONG CScore::UpScore(LONG lInc)      /* INCREASE SCORE                 */
    {
    SetScore(GetScore()+lInc);
    return GetScore();
    }
```

ホ　最高点クラス

```
#include "CHISCORE.H"

CHighScore::CHighScore(void)
    {
    keep=new CKeep;
    keep->SetName("HIGH.TXT");
    LoadScore();
    }

CHighScore::~CHighScore(void)
    {
    delete keep;
    }

void CHighScore::LoadScore(void)
    {                           /* LOAD HIGH SCORE              */
    SetScore(keep->PullScore());
    }

void CHighScore::PrintScore(void)/* DISPLAY HIGH SCORE           */
    {
    SetXY(iScoreX, iScoreY);
    PrintText("最高点  :");
    PrintNumber(lScore);
    }

void CHighScore::SaveScore(void)
    {                           /* SAVE HIGH SCORE              */
    keep->PushScore(GetScore());
    }

void CHighScore::SetScorePos(int iX, int iY)
    {
    CScore::SetScorePos(iX, iY);
    }
```

ヘ ステージクラス

```
#include "CSTAGE.H"

CStage::CStage(int iStartSpeed, int iBlockInStage)
    {                                /* CONSTRUCTOR                    */
    iSpeed=iStartSpeed;
    iBlockStageUnit=iBlockInStage;
    iDefaultSpeed=iSpeed;
    iBlockNo=0;
    }

int  CStage::GetBlockNo(void)       /* GET CURRENT BLOCK NO           */
    {
    return iBlockNo;
    }

int  CStage::GetDefaultSpeed(void)    /* GET DEFAULT SPEED            */
    {
    return iDefaultSpeed;
    }

int  CStage::GetFallPos(void)      /* GET BLOCK FALL POSITION         */
    {
    return (iBlockNo/iBlockStageUnit/9)*3;
    }

int CStage::GetRemainBlock(void)       /* GET REMAINING BLOCK NO      */
    {
    return iBlockStageUnit-iBlockNo%iBlockStageUnit-1;
    }

int CStage::GetSpeed(void)    /* GET SPEED                    */
    {
    return iSpeed;
    }

int CStage::GetStage(void)         /* GET STAGE NO                   */
    {
    return (iBlockNo/iBlockStageUnit)%9+1;
    }
```

```
void CStage::NewStage(void)
    {
    if(1!=GetStage())
      /* MAKE SPEED FASTER */
      SetSpeed(GetSpeed()*2/3);
    else
      /* RESET SPEED TO FIRST TIME */
      SetSpeed(GetDefaultSpeed());
    PrintStage();
    PrintRemainBlock();
    }

void CStage::PrintRemainBlock(void)  /* PRINT REMAINING BLOCK NO     */
    {
    SetXY(iDisplayX, iDisplayY+2);
    PrintText("残り     :");
    PrintNumber(GetRemainBlock(), 2);
    }

void CStage::PrintStage(void)  /* PRINT STAGE NO                */
    {
    SetXY(iDisplayX, iDisplayY);
    PrintText("ステージ:");
    PrintNumber(GetStage());
    }

void CStage::SetBlockNo(int iData) /* SET BLOCK NO            */
    {
    iBlockNo=iData;
    }

void CStage::SetBlockStageUnit(int iData) /* SET BLOCK STAGE UNIT        */
    {
    iBlockStageUnit=iData;
    }

void CStage::SetDisplayPosition(int iX, int iY)
    {                              /* SET DISPLAY POSITION          */
    if(iX>(ScreenWidth()/2-8))
      iX=ScreenWidth()/2-8;
```

```
    iDisplayX=iX;
    iDisplayY=iY;
    }

void CStage::SetSpeed(int iData)    /* SET SPEED                    */
    {
    if(iData>0)
      iSpeed=iData;
    else
      iSpeed=1;
    }

BOOL CStage::UpBlockNo(void)        /* UP BLOCK NO                  */
    {
    iBlockNo++;
    if(iBlockNo%iBlockStageUnit==0)
      return(TRUE);
    else
      return(FALSE);
    }
```

ト　基本関数ラップクラス

```
#include "CWRAP.H"

void CWrap::Clrscr(void)
    {
    clrscr();
    }

void CWrap::Delay(unsigned i)
    {
    delay(i);
    }

void CWrap::Gotoxy(int x,int y)
    {
    gotoxy(x,y);
    }

int CWrap::ScreenHeight(void)
    {
    struct text_info ti;
    gettextinfo(&ti);
    return ti.screenheight;
    }

int CWrap::ScreenWidth(void)
    {
    struct text_info ti;
    gettextinfo(&ti);
    return ti.screenwidth;
    }

void CWrap::Textcolor(int i)
    {
    textcolor(i);
    }
```

チ　画面表示クラス

```
#include "CSCREEN.H"

CScreen::CScreen(void)              /* CONSTRUCTOR                    */
    {
    iSizeY=ScreenHeight();
    iSizeX=ScreenWidth()/2;
    iSizeMax=iSizeX*iSizeY;
    }

int  CScreen::CheckKey(void)        /* CHECK KEY HIT                  */
    {
    if(Kbhit())
      {
      iKey=Getch();
      while(Kbhit())
          iKey=Getch();
      return TRUE;
      }
    else
      {
      if(bKeyForce==TRUE)
          return TRUE;
      else
          return FALSE;
      }
    }

void CScreen::ClearScreen(void)     /* CLEAR ALL SCREEN               */
    {
    Clrscr();
    iCurPos=0;
    }

int  CScreen::GetX(void)            /* GET CURSOR X POSITION          */
    {
    return iCurPos%iSizeX;
    }

int  CScreen::GetY(void)            /* GET CURSOR Y POSITION          */
```

```
    {
    return iCurPos/iSizeX;
    }

int  CScreen::GetKey(void)           /* GET KEY DATA                    */
    {
    if(bKeyForce==TRUE)
      {
      bKeyForce=FALSE;
      return iKeyForce;
      }
    else
      return iKey;
    }

void CScreen::IncCursor(int iData)    /* INCREASE CURSOR POSITION        */
    {
    if(iSizeMax<=iCurPos+iData)
      iCurPos=(iCurPos+iData)%iSizeX+(iSizeY-1)*iSizeX;
    else
      iCurPos+=iData;
    }

void CScreen::Pause(void)             /* PAUSE ONLY                      */
    {
    Getch();
    }

void CScreen::PrintNumber(LONG l)     /* PRINT NUMBER                    */
    {
    Sprintf(sNumberWork,"%ld",l);
    IncCursor(strlen(sNumberWork));
    Cprintf(sNumberWork);
    }

void CScreen::PrintNumber(int i)      /* PRINT NUMBER                    */
    {
    Sprintf(sNumberWork,"%d",i);
    IncCursor((strlen(sNumberWork)+1)/2);
    Cprintf(sNumberWork);
    }
```

```
void CScreen::PrintNumber(int i, int iCol)
    {                              /* PRINT NUMBER (INT, FIX)          */
    switch(iCol)
      {
      case 1:
          Sprintf(sNumberWork,"%01d ", i);
          break;
      case 2:
          Sprintf(sNumberWork,"%02d", i);
          break;
      case 3:
          Sprintf(sNumberWork,"%03d ", i);
          break;
      case 4:
          Sprintf(sNumberWork,"%04d", i);
          break;
      default:
          Sprintf(sNumberWork,"%05d ", i);
          break;
      }
    IncCursor((strlen(sNumberWork)+1)/2);
    Cprintf(sNumberWork);
    }

void CScreen::PrintText(char *sText) /* PRINT STRING                    */
    {
    IncCursor((strlen(sText)+1)/2);
    Cprintf("%s", sText);
    }

void CScreen::Rectangle(int iX1, int iY1, int iX2, int iY2)
    {                              /* DRAW RECTANGLE                   */
    int i;

    SetXY(iX1, iY1);
    PrintText("┌");
    for(i=iX1+1;i<=iX2-1;i++)
      PrintText("─");
    PrintText("┐");
    SetXY(iX1, iY2);
```

```
    PrintText("└");
    for(i=iX1+1;i<=iX2-1;i++)
      PrintText("─");
    PrintText("┘");
    for(i=1;i<iY2-iY1;i++)
      {
      SetXY(iX1, iY1+i);
      PrintText("│");
      SetXY(iX2, iY1+i);
      PrintText("│");
      }
    }

void CScreen::SetColor(int iData)    /* SET COLOR                    */
    {
    Textcolor(iData);
    }

void CScreen::SetKeyForce(int iKey)  /* SET KEY FORCE FOR GETKEY     */
    {
    bKeyForce=TRUE;
    iKeyForce=iKey;
    }

void CScreen::SetOffset(int iX, int iY)
    {                              /* SET CURSOR POSITION OFFSET   */
    iCurXoffset=iX+1;
    iCurYoffset=iY+1;
    }

void CScreen::SetRelXY(int iX, int iY)
    {                              /* SET RELATIVE  X, Y (0, 0)-   */
    iCurPos=(iX+iCurXoffset)+(iY+iCurYoffset)*iSizeX;
    Gotoxy((iX+iCurXoffset+1)*2, iY+iCurYoffset+1);
    }

void CScreen::SetXY(int iX, int iY)   /* SET CURSOR ON X, Y (0, 0)-  */
    {
    iCurPos=iX+iY*iSizeX;
    Gotoxy((iX+1)*2, iY+1);
    }
```

リ　ファイル入出力クラス

```
#include "CDATUM.H"

CDatum::CDatum(char *sName, BOOL bType)
    {                       /* DATA OPEN FILE                  */
    long lCurpos;

    strcpy(sFileName, sName);
    if(bType==OPN_WRITE)
      fileStatic=fopen(sFileName,"w+");
    else
      fileStatic=fopen(sFileName,"r");
    bOpenType=bType;
    if(fileStatic!=NULL)
      {
      lCurpos=ftell(fileStatic);
      fseek(fileStatic, 0L, SEEK_END);
      iLength=ftell(fileStatic)+1;
      fseek(fileStatic, lCurpos, SEEK_SET);
      }
    else
      iLength=0;
    }

CDatum:: CDatum(void)                /* DATA CLOSE FILE                */
    {
    if(fileStatic!=NULL)
      fclose(fileStatic);
    }

void CDatum::GetData(char *sData)    /* GET CHAR FROM FILE             */
    {
    if(fileStatic!=NULL)
      {
      fseek(fileStatic, 0L, SEEK_SET);
      fgets(sData, iLength, fileStatic);
      }
    else
      strcpy(sData,"");
    return;
    }

int  CDatum::GetLength(void)         /* GET DATA LENGTH                */
    {
    return iLength;
    }

int  CDatum::PutData(char *sData)
    {
                  /* PUT CHAR TO FILE                */
    if(fileStatic!=NULL)
      {
      fputs(sData, fileStatic);
      iLength=strlen(sData)+1;
      }
    else
      iLength=0;
    return iLength;
    }
```

ｽ 最高点保存クラス

```
#include "CKEEP.H"

LONG CKeep::PullScore(void)          /* PULL HIGH SCORE FROM FILE     */
    {
    CDatum *datum;
    char sWork[25];
    LONG lRet;

    datum=new CDatum(sHighScore, OPN_READ);
    datum->GetData(sWork);
    lRet=atol(sWork);
    delete datum;
    return lRet;
    }

void CKeep::PushScore(LONG lScore)
    {                                    /* PUSH HIGH SCORE TO FILE       */
    CDatum *datum;
    char sWork[25];

    datum=new CDatum(sHighScore, OPN_WRITE);
    ltoa(lScore, sWork, 10);
    datum->PutData(sWork);
    delete datum;
    }

void CKeep::SetName(char *sName)     /* SET FILE NAME                 */
    {
    strcpy(sHighScore, sName);
    }
```

(11) メイン関数作成

各クラスの実装が終わったら、それらを関連させてブロックゲームを動作させるためのメイン関数を作成する。

```
/*************************************************************************/
/*  ブロックゲーム  for Borland C++ Ver 3.1                              */
/*************************************************************************/
/*    Copyright(c) 1996,1997 by 労働省                                   */
/*    All Rights Reserved.                                               */
/*************************************************************************/

/*************************************************************************/
/*  CLASS INHERITANCE TABLE                                              */
/*************************************************************************/
/*    CWrap--+---CScreen--+---CStage                                     */
/*           |            +---CBack                                      */
/*           |            +---CScore------CHighScore                     */
/*           |            +---CBlock                                     */
/*           +---CDatum                                                  */
/*           +---CKeep                                                   */
/*           +---CShape                                                  */
/*************************************************************************/

/*************************************************************************/
/*  CLASSRELATION TABLE                                                  */
/*************************************************************************/
/* +------------+    +--------+    +--------+                           */
/* | CHighScore |--->| CKeep  |--->| CDatum |                           */
/* +------------+    +--------+    +--------+                           */
/*                                                                       */
/*       |           +--------+    +--------+                           */
/*       +-----------| CBlock |--->| CScore |                           */
/*                   +--------+    +--------+                           */
/*                     | | |                                             */
/*     +---------------+ | +-----------+                                 */
/*     |                 |             |                                 */
/*     v                 v             v                                 */
/* +--------+        +--------+    +--------+                           */
/* | CStage |        | CBack  |    | CShape |                           */
/* +--------+        +--------+    +--------+                           */
/*************************************************************************/
```

```
/* INCLUDE FILES */
#include "BLOCK.H"

void main(void)
    {
    BOOL  bCheckResult;       /* RETURN OF HIT BLOCK              */
    CBack  *back;             /* CBack OBJECT                     */
    CBlock *block;            /* CBlock OBJECT                    */
    CHighScore *highscore;    /* CHighScore OBJECT                */
    CScore *score;            /* CScore OBJECT                    */
    CShape *shape;            /* CShape OBJECT                    */
    CStage *stage;            /* CStage OBJECT                    */

    /* SET PARAMETER FOR SETTING BACKGROUND POSITION ON SCREEN    */
    int   iFrameX1=14;        /* LEFT-TOP X POSITION              */
    int   iFrameY1=1;         /* LEFT-TOP Y POSITION              */
    int   iFrameX2=26;        /* RIGHT-BOTTOM X POSITION          */
    int   iFrameY2=23;        /* RIGHT-BOTTOM Y POSITION          */
    int   iSpeed=200;         /* INITIAL SPEED OF FALING BLOCKS   */
    int   iBlockInStage=60;   /* BLOCK NUMBER DURING 1 STAGE      */

    /* CREATE BLOCK INITIAL SHAPE     */
    shape=new CShape(7); // MAX 20
    shape->SetPattern(0,"1111000000000000");
    shape->SetPattern(1,"1110100000000000");
    shape->SetPattern(2,"1110001000000000");
    shape->SetPattern(3,"1100011000000000");
    shape->SetPattern(4,"0110110000000000");
    shape->SetPattern(5,"1100110000000000");
    shape->SetPattern(6,"1110010000000000");
    /* CREATE STAGE FOR GAME */
    back=new CBack(iFrameX1, iFrameY1, iFrameX2, iFrameY2);
    stage=new CStage(iSpeed, iBlockInStage);
    highscore=new CHighScore;
    back->SetKeyForce(MES_NEWGAME);
    /* LOOP FOR MESSAGE */
    while(1==1)
      {
      back->WaitKey(stage->GetSpeed());
      switch(back->GetKey())
```

```
{
case KEY_ESC: /* QUIT */
    block->UpScorePair(0);
    highscore->SaveScore();
    delete block;
    delete score;
    delete stage;
    delete highscore;
    delete back;
    delete shape;
    exit(0);
case KEY_DOWN: /* DOWN MOVE */
    block->DownBlock();
    break;
case MES_FALLBLOCK: /* FALL BLOCK */
    block->FallBlock();
    break;
case KEY_LEFT: /* LEFT MOVE */
    block->LeftBlock();
    break;
case KEY_RIGHT: /* RIGHT MOVE */
    block->RightBlock();
    break;
case KEY_SPACE: /* RIGHT ROLL */
    block->TestAndRollBlock();
    break;
case MES_NEWBLOCK: /* CREATE NEW BLOCK */
    if(TRUE==(stage->UpBlockNo()))
        {
        /* STAGE IS FINISHED */
        back->PrintFallDown(stage->GetSpeed());
        delete block;
        block=new CBlock(back, shape, stage, score, highscore);
        back->SetKeyForce(MES_NEWSTAGE);
        break;
        }
    /* CREATE NEW BLOCK */
    delete block;
    block=new CBlock(back, shape, stage, score, highscore);
    block->TestGameOver();
    break;
```

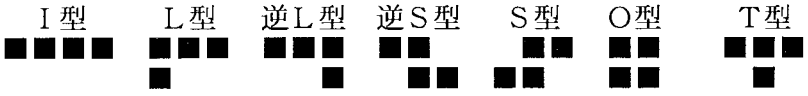
```
        case MES_PAUSE: /* PAUSE */
            back->Pause();
            break;
        case MES_SPEEDUP: /* SPEED UP */
            stage->SetSpeed((stage->GetSpeed())/2);
            break;
        case MES_SPEEDDOWN: /* SPEED DOWN */
            stage->SetSpeed((stage->GetSpeed())*2);
            break;
        case MES_RESTARTGAME: /* NEW GAME */
            delete back;
            delete score;
            delete block;
            delete stage;
            back=new CBack(iFrameX1, iFrameY1, iFrameX2, iFrameY2);
            stage=new CStage(iSpeed, iBlockInStage);
        case MES_NEWGAME: /* NEW GAME */
            score=new CScore;
            block=new CBlock(back, shape, stage, score, highscore);
            block->SetEachPosition();
        case MES_REDRAW: /* REDRAW SCREEN */
            back->ClearScreen();
            back->PrintTitle();
            back->PrintFrame();
            back->PrintBack();
            block->DisplayScores();
            block->PrintBlock();
            stage->PrintStage();
            break;
        case MES_NEWSTAGE: /* NEW STAGE */
            back->ClearFrame();
            stage->NewStage();
            break;
        }
    }
}
```

7 プログラムの改造例

これまで説明してきた「ブロックゲーム」は、オブジェクト指向分析・設計・プログラミング手法によって設計してある。そこで、クラスに与えるパラメタを変更し、再コンパイルすることによって、簡単にゲームの内容を改造することができる。それについて説明する。

(1) ブロック原形クラスのパラメタ変更

現在のメイン関数では、ブロック原形クラス（ＣＳｈａｐｅクラス）のＳｅｔＰａｔｔｅｒｎメンバ関数は、既に説明した７種類のブロックを原形として定義するものである。




```
shape=new CShape(7); // MAX 20
shape->SetPattern(0,"1111000000000000");
shape->SetPattern(1,"1110100000000000");
shape->SetPattern(2,"1110001000000000");
shape->SetPattern(3,"1100011000000000");
shape->SetPattern(4,"0110110000000000");
shape->SetPattern(5,"1100110000000000");
shape->SetPattern(6,"1110010000000000");
       :
       :
delete shape;
```

Ｓｈａｐｅクラスを作成するときに指定する「７」という数は、上記７個のブロックを定義することを指定するためのものである。

その後、７個のブロックを実際に定義する。

まず、1111000000000000　という文字列は、４文字ずつ区切ることによって、Ｉ型を定義している。

```
1111000000000000  →  1111
                     0000
                     0000
                     0000
```

また、1110100000000000　という文字列は、４文字ずつ区切ることによって、Ｌ型を定義している。

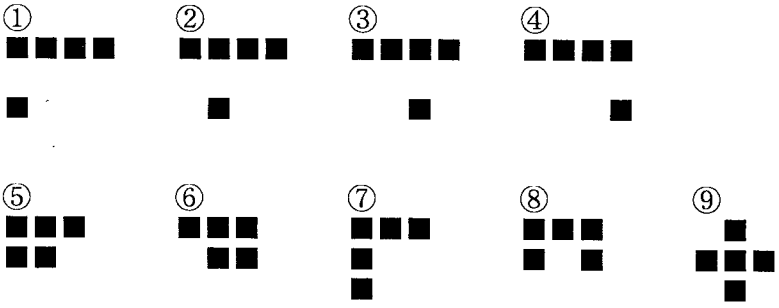
```
1110100000000000  →  1110
                     1000
                     0000
                     0000
```

従って、Ｓｈａｐｅクラスのパラメタを変更することによって、ゲームの内容を改造することができる。

例えば、次のように定義する。

```
shape=new CShape(9); // MAX 20
shape->SetPattern(0,"1111100000000000");
shape->SetPattern(1,"1111010000000000");
shape->SetPattern(2,"1111001000000000");
shape->SetPattern(3,"1111000100000000");
shape->SetPattern(4,"1110110000000000");
shape->SetPattern(5,"1110011000000000");
shape->SetPattern(6,"1110100010000000");
shape->SetPattern(7,"1110101000000000");
shape->SetPattern(8,"0100111001000000");
      :
      :
delete shape;
```

この場合、Ｓｈａｐｅクラスは、９種類のブロックを原形として定義することになる。



(2) 背景クラスのパラメタ変更

現在のメイン関数では、背景クラス（ＣＢａｃｋクラス）のコンストラクタによって、背景となる枠の大きさを設定している。

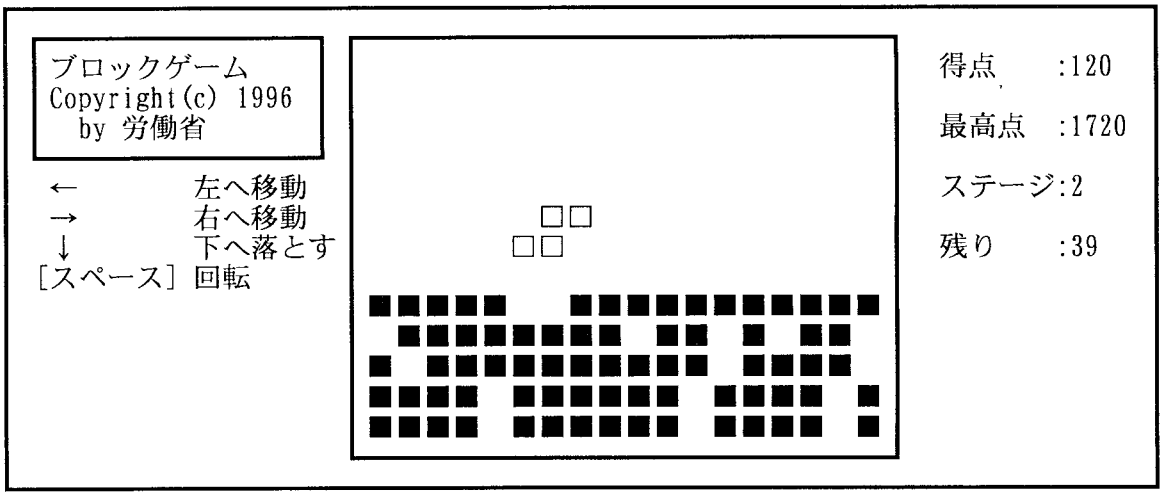
```
int   iFrameX1=14;        /* LEFT-TOP X POSITION          */
int   iFrameY1=1;         /* LEFT-TOP Y POSITION          */
int   iFrameX2=26;        /* RIGHT-BOTTOM X POSITION      */
int   iFrameY2=23;        /* RIGHT-BOTTOM Y POSITION      */
      :
      :
back=new CBack(iFrameX1,iFrameY1,iFrameX2,iFrameY2);
      :
      :
delete back;
```

この場合、Ｘ座標は１４～２６、Ｙ座標は１～２３という範囲になっている。この値を変更することによって、ブロックゲームの画面の大きさを変えることができる。

例えば、次のように定義する。

```
int    iFrameX1=10;        /* LEFT-TOP X POSITION              */
int    iFrameY1=3;         /* LEFT-TOP Y POSITION              */
int    iFrameX2=36;        /* RIGHT-BOTTOM X POSITION          */
int    iFrameY2=20;        /* RIGHT-BOTTOM Y POSITION          */
       :
       :
back=new CBack(iFrameX1, iFrameY1, iFrameX2, iFrameY2);
       :
       :
delete back;
```

この場合、画面は横長になる。



(3) ステージクラスのパラメタ変更

現在のメイン関数では、ステージクラス（ＣＳｔａｇｅクラス）のコンストラクタによって、ステージごとのブロック落下の速度と総ブロック数を設定している。

```
int    iSpeed=200;         /* INITIAL SPEED OF FALLING BLOCKS  */
int    iBlockInStage=60;   /* BLOCK NUMBER DURING 1 STAGE      */
              :
              :
stage=new CStage(iSpeed, iBlockInStage);
              :
              :
delete stage;
```

この場合、最初のステージのブロックの落下速度は２００ms、ステージごとの総ブロック数は６０個となる。この値を変更することによって、ブロックゲームのステージごとのブロック落下速度とステージごとの総ブロック数を変えることができる。例えば、次のように定義する。

```
int    iSpeed=500;          /* INITIAL SPEED OF FALLING BLOCKS  */
int    iBlockInStage=30;    /* BLOCK NUMBER DURING 1 STAGE      */
                :
                :
stage=new CStage(iSpeed,iBlockInStage);
                :
                :
delete stage;
```

この変更によって、最初のステージのブロックの落下速度は５００ms、ステージごとの総ブロック数は３０個となり、より容易にステージをクリアできるようになる。

8 プログラムの移植

現在、「ブロックゲーム」はＭＳ－ＤＯＳ、Windows3. 1、そしてWindows95という３種類のＯＳ上で動作している。

しかし、基本関数ラップクラス（ＣＷｒａｐクラス）を変更することによって、現在動作している以外の環境に容易に移植することができる。

具体的には、以下の九つの関数を記述することができる環境であれば、移植は可能である。

- printf関数（表示関数）
- delay関数（遅延関数）
- getch関数（文字読み込み関数）
- gotoxy関数（カーソル移動関数）
- kbhit関数（キーボード入力検知関数）
- screenheight関数（画面のＹ方向の長さを知る関数）
- screenwidth関数（画面のＸ方向の長さを知る関数）
- sprintf関数（フォーマット付き変換関数）
- textcolor関数（文字色設定関数）