2011年度 修士論文

ゲームを題材とした
プログラミング教育支援システムの
開発と評価

大阪産業大学大学院 工学研究科
博士前期課程 情報システム工学専攻

10MH04 東川諒央

目次

概要

プログラミング教育を受けたても自力でプロラグラムを書けない学生が少なくない事が近年問題視されている。この問題に対し、様々な先行研究が存在しているが未だ有効な成果をあげている教育方法は存在しない。現状のプログラミング教育において重要視されている要素は大きく分けて、アルゴリズムの学習方法と学生のモチーベーションの維持である。アルゴリズムと言語の構文の学習を切り分けて学習する方法が広く採用され、アルゴリズムの学習の後に言語を習得という順番で学習させる事で習得容易性を確保しようとしている。さらに、学習者のモチベーションを維持させる事は自主的な学習を促す事に繋がる。しかし、上記の方法では自身の思い描くプログラムの正当性を確認する為には正確なプログラムが要求される。これでは、この事実は、プログラミングの経験が無い学習者がプログラムを書く際に戸惑うのでは無いかと考えられる。

本研究ではこの問題を解決する為の手法として、プログラミング言語を抽象化することを考えた。言語の構文を抽象化し、プログラミングにおける複雑さを取り除く事で構文エラーでの躓きを減らす事ができる。通常、プログラミングに用いられている概念を、プログラミングの初学者に動作を見ずに想像させる事は困難である。そのため、学習者自身の手でプログラムを書き、その動作を確認する事が概念の理解の助けになる。しかし、プログラミングは言語によって決められた厳密な記述を必要とし、学習者が思い描くプログラムを記述し、直ぐに動作させる事は困難である。結果、学習者はプログラムの動作を確認するだけで、多大な時間を要する事になる。プログラミング言語を抽象化し、動作の確認までの時間を短かくする事ができれば通常のプログラミングより多くの回数のフィードバックを得る事ができる。そして、学習者がプログラミングに慣れるに従い抽象化された言語を少しずつ通常の言語に戻す。そうする事で、学習者が抽象的に捉えていた概念を具体的に理解させる事が出来ると考えた。

その為、従来の講義によるプログラミングの学習方法において学習者の理解度と、プログラミング時にどういった状況において躓くのかを調査した。習熟度調査はプログラミングを既に講義にて学習している三回生を対象に行なった。そして、習熟度調査の結果を基に学習者がプログラムの構築と実行を容易に行なえる環境を作成し、学習者への効果を一回生と三回生を対象に演習とアンケートによって把握した。

1 はじめに

情報科学系の教育としてのプログラミングを学習した学生のうち、プログラムが書けない学生は少なくない。この現象は日本だけの問題では無く国内外問わず非常に問題視されており、プログラミング教育の問題に関する記事が国内外を問わず Web 上で公開されている [1–3]。このような事実から、教育機関の指導が学習者のプログラミング能力の向上に影響を及ぼしていない事が理解できる。その為、現状での指導方法を改める必要があると考える。

学習者が教育を受けて尚プログラムを書けるに至らない原因として、プログラミングのアルゴリズムに関する先行研究 [4] ではいくつかの問題をあげている。問題の一つとして、アルゴリズムをプログラミング学習者が理解できていないことがある。ここで言うアルゴリズムとは、解決するべき問題が存在する場合に問題を解決するために踏むべき手順の事を指す。そのため、プログラミングにおいては問題解決のプロセスを論理的に把握することが肝要であると言える。もう一つの問題点は、プログラミングの初学習者は初めて学ぶ事が多いため平行して学習するものが増え、学習負荷が飛躍的にあがる。そのためプログラミングの学習で躓く学習者は、自身は何を理解していないのかを理解出来ない状況に陥いる。

また、大学等の教育機関の指導では基本的に一対多の体制で学習者を指導する為、全ての学習者に一対一で指導することは実質的に不可能であり、また全ての学習者の理解が及ぶまで指導することも難しい。学習者が授業を理解せず授業から遅れる場合に次への学習意欲を損ってしまう為、自発的学習もあまり望めない。この様な授業において発生するジレンマを解消するため、学習者が問題解決のプロセスを理解し易くなる環境を作ることに着目した。

学習者の意欲を向上させる事において、ゲームがプログラミングの学習においても有効であると云う事実はゲームにおけるプログラミング教育の先行研究 [5,6] からも伺える。現状の教育カリキュラムにより、大阪産業大学工学部情報システム工学科の学生がどの程度のプログラミングスキルを有しているかの検証を行なった。学習者の思考レベルに合せてプログラムを抽象的に記述し、動作する環境を用意することにより、常時学習者に合せた環境を用意する。そして、学習者の思考に合せた学習方法を用い、プログラムの抽象的記述が可能な環境時における学生の学習意欲とその効果を検証する。

2 本研究の目的

本研究は初学者を対象とした入門向けのプログラミング学習の効果を高めることを目指したものである。本研究の目的は次の 3 点に集約される。

- プログラミング初学者やプログラミングが出来ない人がプログラミングという行為を経験する
- 学習者が実際に動作するプログラムを容易に書ける環境を構築する
- 構築した環境を実際に学習者に適用し、その教育効果を調査する

上述した 3 点の目的を達成する為には、先ず研究対象であるプログラミング学習者の学習傾向を調査する必要がある。現段階の教育方法で学習者がどの様な理解を示しているのかを知る必要があり、学習者が理解し辛い部分もまた知る必要がある。学習者のプログラミングにおける不理解の傾向を知る事で、学習者が実際にプログラミングを行ない、作成したプログラムを容易に動作させられる環境を構築する段階に移行できる。プログラミングの学習状況の調査は演習によって行なう。学習者が講義で説明を受けていない言語でも演習を実施する。これにより、言語の違いによるプログラミングの行ない易さも同時に調査する。

プログラミング初学者が論理的記述、考えを習得する事は容易にはいかないであろう事は先行研究 [4,5] により明らかになっている。そのため、研究 [4] ではプログラミングを学習する前段階で学習者にアルゴリズムを考えさせる方針を取っている。しかし、そのステップをプログラミングの中に組み込む事が出来るのであれば学習者がプログラミングに関与する時間を増やす事が出来、論理的記述を学ぶ機会を増やす事が出来るのではないかとの仮説を立てた。その為には学習者が考えるレベルの記述でプログラムが動作する必要がある。そこで、抽象的記述によるプログラムの動作環境を構築する事を研究の視野に入れた。

本研究での抽象的記述とは、実際に使用するプログラミング言語の構文に大きく左右されずにプログラミングを可能にする記述を指す。この記述によって学習者の考えをダイレクトに実行しフィードバックを得ることが出来ると考えた。

そしてその環境を構築する為には、ある程度のプログラミングを行なう為の題材が必要である。言語自体を作成してしまうと結局その言語の仕様を覚える必要があり、学習専用の言語を覚える事になる。それではプログラミング初学者がプログラムの作成と動作をさせる事が出来たとしても、実用的な学習を行なう時には新しい言語を覚え直す必要が出てくる。しかし既存の言語では覚える事が多く、また構文の学習に躓く学習者が出てしまう。この問題には、プログラミングの範囲を狭める事で対処する。そのため、プログラミングの題材にはゲームを用いる。ゲームを用いる事でプログラミングの範囲を限定的にし、その範囲内で、プログラミング初学者が誘発させ易い言語の構文を抽象化する。さらに、限定的なプログラミングの環境を用意し、明確な目的を設定する事で学習者は目的を理解しやすくなる。学習方法としてはゲームの攻略用プログラムを学習者に書かせることにした。そこに抽象的に記述できる段階を設けることで学習者の思考レベルに合わせたプログラミングが行なえるようにする。

上記のように、抽象的記述をすることで実行できる環境を用意することでアルゴリズムと言語構文をなるべく分離し、学習者にプログラミングを行なわせる事ができる。また言語の記述レベルを学習者に合せることによって作成、評価、フィードバック、修正のルーチンを容易に行なうことが出来る。これをもって学習者はプログラミングという行為を体験し論理的表現を覚えられる。またプログラミング自体の経験を得ることが出来るため、エラーやバグへの対処、考え方も段階を追って学習できるものと考えられる。

3 先行研究の考察

プログラミング教育に関する先行研究はこれまでにも数多く存在しており、既に明らかになりつつある説も存在する。本章では先行研究によって明らかにされてきている部分について整理し、プログラミング教育に必要とされる要素について考察する。

3.1 アルゴリズム構築能力育成の導入教育: 実作業による概念理解に基づくアルゴリズム構築体験とその効果

研究 [4] では、プログラミング時におけるアルゴリズムの重要性について調査している。同論文において実践すべきと考えられている教育内容は、学習者自らがアルゴリズムを組み立てる事を実践し、それを実行可能なプログラミング言語で記述し、自身が作成したプログラムの正しさを検証することまでを含めた内容である。

この研究では、学習者に対し完成されたプログラムを与え、その動作を確認するような教育方法では学習者がプログラムのアルゴリズムを理解する機会を失してしまう事を危惧している。そのため、同論文において提示されている教育を実施するうえで問題となるのは、教育の方法である。プログラミングの初学者がアルゴリズムの学習と同時にプログラミング言語を学習することによって、アルゴリズムを十分に学習するまでの負担が大きくなる。したがって、学習者がプログラミング言語の学習にすべての注意を向けがちとなり、作成したアルゴリズムが正しいかどうかを十分注意させるのが困難であることが問題点として指摘されている。専門教育では、プログラミング言語を習得した後に時間をかけてアルゴリズムに関する学習をすることも可能であるが、一般情報教育の一環としてプログラミング教育を実施する場合では、専門教育と比較して教育全体にかけられる時間は短い。そのため「アルゴリズムを十分に学習できない」という問題はより顕著に現れることになる。この問題を解決するための教育方法を提案し、その効果を実験授業によって検証した結果について述べており、アルゴリズム構築能力の育成を円滑に実施するためには、講義による説明ではなく、アルゴリズムの概念を体感的に理解させることが重要であると指摘している。

実験検証では学習者が手作業で実行したアルゴリズムをプログラミング言語で記述させ、手作業で解決したものと同一の問題をコンピュータを使って解決する体験を持たせることを行なっている。この教育で想定しているのはプログラミングの初学者であるため、手作業で実行したアルゴリズムをプログラミングするための方法に工夫が必要であると述べている。同研究以前では手作業で理解したアルゴリズムとそのプログラミングを連携させる手法は提案されておらず、「手作業によるアルゴリズムに関する学習」と「プログラミングによるアルゴリズムの学習」は分割されている。前述した体験的な学習とプログラミングを組み合わせた試みとして、手作業によるアルゴリズム構築体験作業がある。アルゴリズムを理解する教具として学習者が自らの手で扱えるカードを用い、学習者がカードを使ってアルゴリズムを手作業で実行する。その後に探索や整列のプログラミングの学習を行う方法が挙げられており、これらの研究における体験的な学習は、プログラミング言語を使ったアルゴリズムの学習の前段階で実施することが想定されている。プログラミングの初学者は複雑なアルゴリズムをプログラムから読み解く能力がないため、学習初めにプログラムを読解し、アルゴリズムを理解することは困難である。そのため、教具を使って手作業でアルゴリズムを実行することで、そのアルゴリズムを体感的に理解することができる。また、複雑なプログラムを提示され、それを読み解くことに意欲を失うことがないと考えられる。

プログラミングの演習には「ことだま on Squeak」を用いている為、プログラミング時に初学者が陥り易い構文によるエラーが発生しにくくなっている。そのため学習者はプログラミング時に、純粋にアルゴリズムのみを考える時間が多くとれ、その事が研究の成功の要因であったと述べている。

同研究から、学習者がアルゴリズムを考える時間を多く取り、言語学習と分離させる事の重要性や、構文エラーによる躓きを排する事がプログラミング教育における課題となる事が分かる。

3.2 ボードゲームの戦略プログラミングを題材とした Java 演習の支援システムの開発

研究 [5] では、近年増加傾向にある、ゲーム戦略を題材にしたプログラミング教育手法を模索している。ゲーム戦略を用いた教育の中で、戦略プログラム同士での対戦をさせ、大会形式によって評価を行なう演習が注目されている。教育効果をより高める為には、多くの対戦過程を学生に提示し、対戦結果から戦略を修正させるような工夫が必要である。同研究では、五五ゲームというボードゲームを学生のプログラミング学習に用いており、ゲームルールは五目並べに石取りを加え、禁じ手やプログラム動作不具合による勝敗決定を取り入れた物である。その為プログラムが最後まで止まらずに動作する事も大会での重要な要素となる。ゲームの特徴として、ルールはよく知られている五目並べの変種である為理解し易い。しかし、石取りルール、禁じ手の追加、資料が少ないなどの為に必勝法が分かっていない。これらの仮定要素のおかげで学習者は自分で試行錯誤しながらプログラムを作成する必要があり、自主的な考察を促す事が出来、演習題材として魅力的と捉えられる。

学生が行なうプログラミング部分は予め用意されている着手用メソッドのオーバーライドである。その為学習者が実際に作成する範囲は狭く、ゲームの着手メソッドをプログラムする上で必要となる仕様を理解するだけでプログラムの動作を確認することが出来る。この演習の際、学生には評価値を用いた戦略プログラムのサンプルを配布し、少なくともこのサンプルプログラムに勝利する事を目標とさせている。5 週間という演習期間を設け、更に最後の中間大会と最終大会の 2 回の大会を設けている。中間大会では大半の学生がランダムに打つ戦略を取っていたのに対し、サンプルプログラムの提示を行なってからは、戦略の組み立て方を理解した学生が多く見られた事が述べられている。そのため学習者はサンプルが全く無い状態でプログラミングを行なうよりも、ある程度の指標を用意し、その上で考える幅を持たせる事が必要であると考えられる。また、サンプルの配布のみにとどまらず、デバッグを行なう為の任意の局面からのゲーム開始機能を持たせている。この機能により練習局面の配布が可能である。戦略が分からずプログラムを完成させる事が出来ない学生には詰め将棋に当たる練習問題を与え、その解答により最低限の課題点としての救済を与えている。

また大会運営は Web サーバ上により行なわれる。学生が Web ブラウザから提出する事でコードがコンパイルされ、バイナリとともにサーバへ保存される。後に提出された学生のソースの暫定順位を決める為に必要十分な対戦相手をリストアップし、自動対戦させる。その戦績を記録し、全体の勝敗表に反映させる事で新たな順位が決まる。そのためソースの提出を行なうだけで、学生は自分のプログラムの順位をブラウザ上から確認できる。サーバに順位が反映されるまでに時間が多少かかるものの、リアルタイムに学生の戦績が変動する事を確認できる。これにより、プログラム提出後の学生の更なる試行錯誤を促す事が出来る。

3.3 プログラミング入門支援ゲームの試作と評価

研究 [6] は、プログラミング未学習者を対象にしたプログラミング学習前ゲームである。この研究の目的はゲームを学習に用いる事による学習者のモチベーション維持と、ゲームを行なう事によってプログラミングに必要とされる考え方を身に付ける事である。通常のプログラミング学習である場合、学習者が普段行なっている考え方と異なる思考方法を求められる。プログラミングにおいては論理的思考を求められる事がこの論文では述べられているが、プログラミング初学者がその思考を習得しつつ、言語の構文やアルゴリズムを理解する事が困難である事も挙げられている。その為、プログラミング時の考え方をゲームから学べるシステムを考案している。

ゲーム中に身に付けられると仮定されている知識と体験にはプログラミングの基本 3 構造である、逐次実行、条件分岐、繰り返しがある。ゲームを通してこれらを体験する事で、後のプログラミングの学習を行なう際に、授業理解の促進が狙え、難解と捉えられがちなプログラミングへの敷居を下げられるとしている。また、同時にこのゲームを用いる事でプログラミングにおいて大切であるとされている思考力を鍛える事も兼ていると述べている。

この研究で使用されているゲームはオリジナルのボードゲームで、その目的は、決められた手数内でゲームのキャラクターをゴールへ導く事である。キャラクターにはボード上を上下左右に移動する行為が与えられており、最初は単順に移動するだけで攻略できる様になっている。しかし、ゲームを進めるに従い単順に移動するだけでは攻略する事が出来ず、繰り返しや、条件分岐といったルールを用いる事によってのみ攻略出来るステージが登場する。このゲームでは繰り返しをパッケージ化と言う表現をしている。パッケージ化を行なう事で複数回の行動を 1 ステップにまとめ、繰り返し利用することが出来る。ステージによって手数が決まっている為、手数の足りない場面ではパッケージ化を用いる事で手数が節約でき、通常の方法では攻略できないステージを攻略する事が出来る。条件分岐はゲームの登場キャラクターである a4w にアイテムを装備させる事で再現しており、アイテムの装備時は特殊な行動が必要なマスを自動で進む様になっている。

同研究では学習者への効果の評価手段としてアンケートを用いている。そのアンケートの回答では研究の目的として挙げていた、学習者のモチベーションの維持やプログラミングに必要である基本 3 構造の一部の学習の効果に肯定的な意見が見られている。そのため、学習者にプログラミングを意識させる事無く、プログラミングに近い行為を行なわせる事が学習者のモチベーションと前提知識の習得に良い効果を与えている事が分かる。一方、条件分岐や繰り返し文においての解答にはネガティブなものが見られる。繰り返し文については単順にそのステージまで辿りついていなかった学生が見られる為、ゲームの進め方の見直しが必要である事が述べられている。しかし、条件分岐についてはゲームによって理解が出来た等の解答は得られていない。そのため、ゲームのシステムが条件分岐を思考する為に適していない可能性がある。学習効果の向上には不十分な結果ではあるが、プログラミングの学習以前にプログラムの構築方法を学習出来る事がこの論文で指摘されている。

3.4 JavaSript プログラミングと情報教養教育

研究 [7] では情報教育が始まった高校生に情報教養教育の中である程度のプログラミング教育が必要であることを指摘している。その理由として、情報機器の普及が挙げられる。情報機器が身の回りにあるため、近年の学生は情報機器の操作については困難を感じない。しかし、操作が出来るだけではリテラシ教育には不十分で、情報機器の原理を具体的な操作により体験し、理解する事が重要であるとしている。リテラシ教育では、原理を通して技術を理解することで急激に情報化されてきている社会を正しく捉えられる様になる事が望ましい。そしてその教育は、学習者が一方的に情報を与えられて覚える方法では成功しない事を述べている。リテラシの様な単順な正答が無い場合、学習者自身が独自の答えを考察する必要がある。そのためには学生自身に工夫させる実習を行なう事が有用である事を指摘している。そして、そのためにはプログラミングを学ぶ必要がある。情報処理学会では、2005 年 10 月に “日本の情報教育・情報処理教育に関する提言 2005” を公表し、手順的な自動処理を体験学習する重要性を訴えている。

その提言 [8] の一部を論文中で次の様に引用している。“十分多くの人に、「課題を分析し、系統的に解決策を考え、 コンピュータに実行可能な形で明示的に表現し、 実行結果を検討し必要なら反復改良する」 プロセス (以下「手順的な自動処理」の構築と呼ぶ) を体験的に理解してもらう必要がある。”

さらにプログラミングの教育には、自身が作成したものが動作する楽しさを学習者の意欲に繋げることも重要視されている。そのため容易に開発環境と実行環境を用意することができ、GUI によってプログラムの動作のイメージや確認が行ない易い JavaScript を題材に扱っている。JavaScript であれば、ブラウザがあれば動作し、記述にはテキストエディタがあれば開発できる為学習者をプログラミングに集中させる事ができ、尚且つ手軽に演習を行なう事ができる。しかし、JavaScript のみでは、ローカルサーバのファイルを直接的に操作する事は出来ない為他に C 言語等の学習を通してファイル操作プログラミングを体験する事が望ましいとしている。

プログラミングの学習には、“変数と代入・逐次実行・制御構造・関数と引数” などの機能に関するものは前提知識として学習させ、端末を利用した演習を行っている。演習ではプログラムの処理速度を学習者に意識させており、アルゴリズムの違いによる計算時間の変化を理解させる内容としている。

ただ漫然と学習させるだけでは無く、プログラムを作成し実行する工程も、学習者の意識を誘導する事で情報機器の動作原理を知る機会になると分かる。

3.5 先行研究のまとめ

先行研究の特色から、プログラミングの学習に貢献すると考えられる要素は下記の 4 点である。

言語学習とアルゴリズム学習の分離

プログラミング学習時の言語構文を学習する行為とアルゴリズムの学習を同時に行なうと学習負荷が非常に高くなる為、学習タイミングを切り離し、単体でも学習が容易であるアルゴリズムから学習することが望ましい。後に言語を学ぶ事で学習者が、プログラムの誤りが構文かアルゴリズムのどちらにあるのかを見分ける事が出来る様になる。

環境の準備と実行が容易である言語選択

容易に環境の準備、実行を行なえる言語による学習を行なう事で、教育機関によって用意されている端末以外でのプログラミングを行ない易くなる。また、実行を容易に行なう事が出来る為、自身が書いたプログラムの動作を簡単に確認する事ができる。

他者のアルゴリズムの学習機会

プログラミングの学習者が自分では考えつかないアルゴリズムを他者から取り入れる機会を設けることで自身のプログラムとの比較が行なえ、より短時間での学習者の成長を見込める。その際には、単順にコピーをするだけでは意味の無い学習環境を用意する必要がある。

娯楽により関連知識や経験を得る

学習者が勉学を意識することなく、娯楽により高いモチベーションを維持した状態で娯楽の一要素として、学習内容に関連した知識や経験を得る事が見込める。

先行研究には、上述の 4 点の内のいずれかの要素が含まれている。しかし前述の先行研究とその要素はプログラミングの一部分の学習を行なわせるものや、従来通りの学習方法で環境を変えたものが見られ、学習者がプログラミングを行なう事を簡素化、実行させる考察が見られない。一度に多くのものを学習する事は非常に理解を遅らせる原因となるが、プログラミングを学習し、習得する為にはプログラミングの工程の全てを後々経験する必要がある。特に大学の様な短かい期間においての学習となると、学習者のプログラミング経験は多く無く、プログラミングの学習は、知識に偏る事態が起こる。そのような事態を避け、自身でプログラミングを行なえる様にする為には、学習者が自分の考えを試し結果についての考察を積極的に行なわせるべきであり、プログラミングにおける工程を経験によって身に付ける事が重要であると考える。

以上の考察から、本研究ではプログラミング初学者が容易にプログラミングを行ない動作を確認出来る環境がある事で、学習効率が向上すると考え作成した。

4 研究内容

本章では、本研究において使用したゲームおよび、ゲームを用いた学習システムについて説明する。

4.1 利用したゲーム

本研究で利用したゲームは Battle Ship である。Battle Ship とはテーブルゲーム[*1]の一つで戦艦ゲーム、軍艦ゲームなどとも呼ばれる。戦艦ゲームとは、10 × 10 の盤上に見えないよう配置された互いの艦船を交互に攻撃し、相手の全艦船を沈没させる事を目的としたゲームである。

ルールは下記の通りである。

- マス目の設定は縦横各 10 マス。
- 軍艦に耐久力は設定されない。代わりに大きさが設定され、空母（5 マス）、戦艦（4 マス）、巡洋艦（3 マス）、潜水艦（3 マス）、駆逐艦（2 マス）となっている。軍艦のマス目全部に攻撃を受けると沈没となる。
- 軍艦は移動できない。
- 軍艦の隣接マスに攻撃を受けた場合、「水しぶき」など[*2]の申告をしなくてよい。
- 各軍艦にコストや回数制限付きで特殊な攻撃がある。（索敵・絨毯爆撃・連続攻撃等）

今回、この Battle Ship ゲームをプログラミングの初学者用としてコンピュータゲーム化した。

4.2 コンピュータゲーム化におけるルール

元々の Battle Ship を学習に利用するにはルールが複雑すぎる。このルールをそのまま適用し学習者がルールを踏まえたプログラムを書くとなると、プログラムを書く上で考慮すべき要素が多く複雑になるため初学用途としては好ましくない。その為コンピュータゲーム化するにあたり特殊な攻撃の要素を無くすことでルールを簡略化した。

コンピュータゲームは一人プレイ用で、ルールは下記の通りである。

- 海域としてのマス目の設定は縦横 10 マス。
- 敵の軍鑑のみ配置される。配置はランダムに行われる。
- 軍艦には大きさが設定され、戦艦 (5 マス) 、巡洋艦 (4 マス) 、潜水艦 (3 マス) 、駆逐艦 (3 マス) 、輸送艦 (2 マス) となっている。軍艦のマス目全部に攻撃を受けると沈没となる。
- 攻撃は 1 手につき 1 マスにのみ行う。海域内且つ未攻撃のマスであれば他に攻撃の制限は無い。
- 軍鑑は移動しない。
- 全ての軍鑑が沈没した時点でゲームは終了となる。

[*1] 紙と筆記用具があればプレイ可能。三人以上でプレイ可能だが、ゲームが非常に複雑になるのでプレイヤーは基本的に二人。

[*2] 敵艦を発見する為の要素。

4.3 学習用システム

学習者は前述したコンピュータゲーム用ルールに基づいてゲームを行い、より少ない攻撃回数でのクリアを目指すものとした。

そして、コンピュータゲーム用システムには 3 タイプのプログラムを用意した。各プログラムには別々の役割りがある。ゲームをアルゴリズムの学習に用いる上で必要な機能を持たせたプログラムが次の 3 タイプである。

Practice

ゲームをプレイすることでルールを把握し、思考中のアルゴリズムに具体性を持たせる。

Verification

記述されたアルゴリズムが理想通りに実装されているかをチェックする。

Score

アルゴリズムの善し悪しを判断する為、評価を行う。

これらの機能が学習者のガイドラインとなることで、学習システムを用いてプログラミングを行う際に、設計・実行・評価・再構築のルーチンを行いやすくなる。

Verification と Score は Battle Ship の攻撃を担う関数を shoot という名前で別ファイルに分けている。学習者はこの関数を書き換えることによって攻撃のアルゴリズムを実装する。動作はいずれも terminal 内で実行される。

4.3.1 Practice

このプログラムは他の二つとは目的が異なり、学習者がゲームをプレイするためのものである。Battle Ship を実際にプレイし、アルゴリズムを考えるためのもので攻撃したい座標をインタープリタに入力する。

Practice の実行画面を図 1 に示す。

図 1　Practice の実行画面

図 1 は 38 回の攻撃が終った状態である。さらに、38 回目の攻撃が敵艦に命中した事がわかり、残っている敵艦の一覧の表示もされており、インタプリタのプロンプトには 39 回目の攻撃座標の入力を促している。

4.3.2 Verification

このプログラムは実装した攻撃関数の挙動を調べるためのもので、攻撃したマスと攻撃結果、そして現在の手数を一手ずつ出力するプログラムである。軍艦を全て沈没させることによってゲームは終了する。

Verification の実行画面を図 2 に示す。

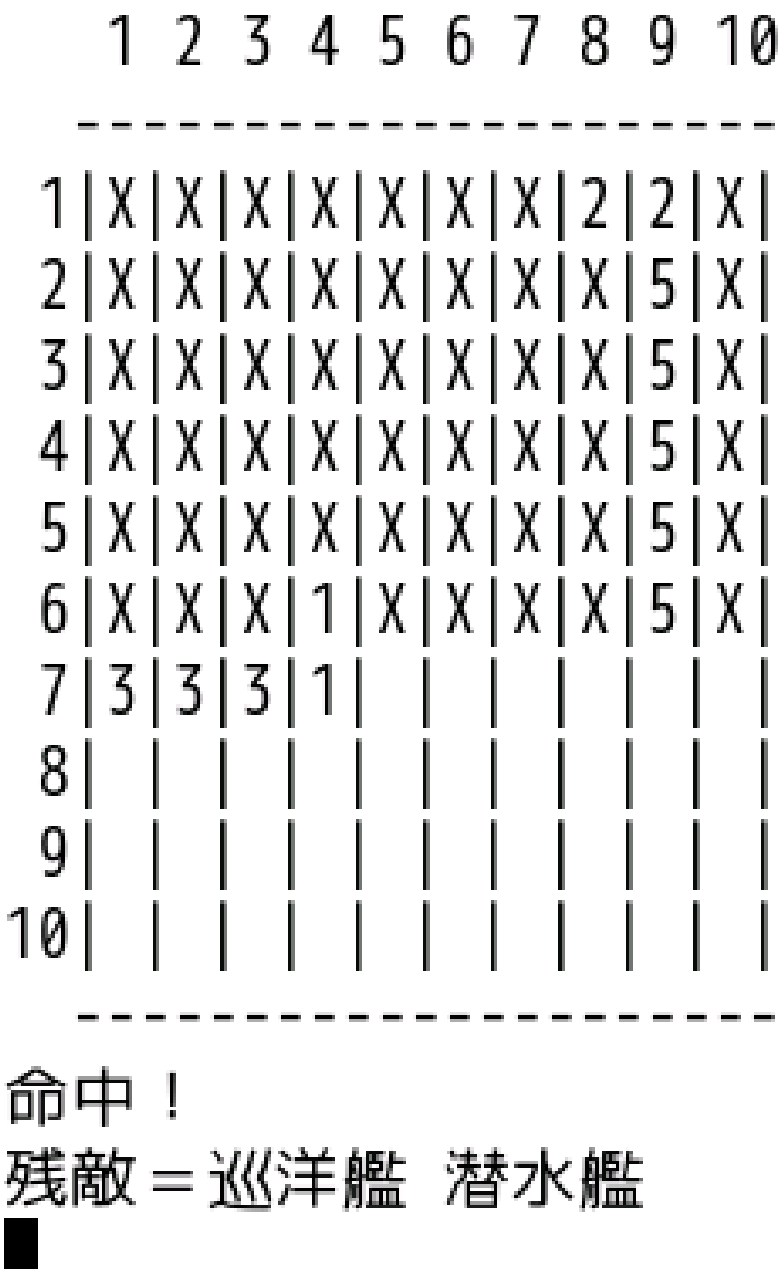

図 2　Verification の実行画面

図 2 は 64 回目の攻撃が終った状態である。64 回目の攻撃が敵艦に命中した事が分かる。残っている敵艦の一覧が表示されており、Enter キーを押す事で次の攻撃が行なわれる。

4.3.3 Score

このプログラムは shoot 関数によって実装した攻撃アルゴリズムのゲーム終了までの平均攻撃回数を調べるものである。100 回ゲームを繰り返し、その平均攻撃回数を出力する。

```
 86回目のプレイ：攻撃回数100回
 87回目のプレイ：攻撃回数 96回
 88回目のプレイ：攻撃回数 89回
 89回目のプレイ：攻撃回数 99回
 90回目のプレイ：攻撃回数 95回
 91回目のプレイ：攻撃回数 62回
 92回目のプレイ：攻撃回数100回
 93回目のプレイ：攻撃回数 83回
 94回目のプレイ：攻撃回数 94回
 95回目のプレイ：攻撃回数 98回
 96回目のプレイ：攻撃回数100回
 97回目のプレイ：攻撃回数 94回
 98回目のプレイ：攻撃回数 99回
 99回目のプレイ：攻撃回数 66回
100回目のプレイ：攻撃回数 85回
平均攻撃回数＝ 87.48
```

図 3　Score の実行画面

ゲームを 100 回繰り返し、各回の攻撃回数と 100 回の平均攻撃回数を表示する。このプログラムではゲームの盤面は表示されない。

4.4 検証用システム

プログラミング学習者の学習傾向を調査する為に Battle Ship とマインスイーパをプログラム化した。二つのゲームをプログラム化した理由は、言語環境の違いによるプログラミングの難易度を調査する為である。言語環境の違いを演習によって調査するには二度の演習を行なう必要があり、同じゲームによる演習を行なった場合、演習参加者が前回の攻略方法を理解した状態でのプログラミングになる可能性がある。これにより、言語を変えると共にゲームも違うものを利用する必要がある。

この調査では、既にプログラミングを講義により学習している学生を対象に行う必要があった為、当時 (2010 年度) の 三回生を対象にシステムを用意した。その際に、プログラミング自体は学習した状態で言語は未学習である事が望ましかった。これには未学習の言語を用いる事でプログラミング初学者の状況と似せた環境にする意図がある。そこで、言語には Python を用いた。Python は初学者向けの言語と言われており、C 言語と比べた場合にデータの扱いが容易である。そのため Python を用いることで抽象化された言語環境の調査も同時に行なう事が可能であると考えた。

Python で作成されたシステムには、4.3 において述べた Battle Ship プログラムの 3 つの機能を用意した。それらを用いて攻撃用関数を学生が作成する仕様である。

さらに Python を用いたプログラミングのみの場合、言語の仕様が理解できない事を考慮し、C 言語によるプログラミング調査も行なう事にした。しかし、同じ演習内容を異なる言語で行なうと前回の演習の考えを持った状態で演習に臨んでしまうため、適正な結果が得られない。そのため異なるゲームプログラムを 2 回目の演習に用いた。

4.5 抽象的記述によるプログラミングについて

4.4 で説明した検証用のシステムでは、どのプログラムを用いてもプログラミング言語をそのまま学習することになるので学習難度が今迄と変わらない。その為本研究で用いるゲームの攻略にあたってはアルゴリズムを抽象的に記述することによってプログラミングを行なえるようにした。ゲーム自体は C 言語で記述されているが、C 言語の記述を殆ど用いることなく攻略プログラムを記述することが出来る。

4.5.1 初学者向けシステム

最大限に抽象化した状態のプログラミングでは、あらかじめ用意した攻略用のマクロを準備し、自身の考えるアルゴリズム通りに動作するようマクロを用いて記述するだけで実行できるようになっている。最大限の抽象化においての構文省略は下記の通りである。

- 条件分岐の省略
- 繰り返し文の省略
- 関数のテンプレート化による周辺知識の排除
- 変数の省略
- セミコロンの省略

下記が攻撃用に用意した攻撃マクロである。

- F000: ランダムに盤面を攻撃
- F001: 左上から 1 マスおきに順に撃つ
- F002: 左上から 2 マスおきに順に撃つ
- F003: 左上から 2 マスおきに順に撃ち下段に下がると攻撃ポイントが 1 マス右にずれる
- F004: 左上から 3 マスおきに順に撃つ
- F005: 左上から 3 マスおきに順に撃ち下段に下がると攻撃ポイントが 1 マス右にずれる
- F006: 左上から 4 マスおきに順に撃つ
- F007: 左上から 4 マスおきに順に撃ち下段に下がると攻撃ポイントが 1 つ右にずれる
- F008: 縦横の座標が奇数のマスに攻撃
- F009: 縦横の座標が偶数のマスに攻撃

更に特殊な条件においてのみ利用できるマクロも用意した。

- F010: 攻撃が当たった場所から上に 4 つ攻撃する
- F011: 攻撃が当たった場所から右に 4 つ攻撃する
- F012: 攻撃が当たった場所から下に 4 つ攻撃する
- F013: 攻撃が当たった場所から左に 4 つ攻撃する

14 個のマクロの組合せを用いて最初のプログラミングを行なってもらい、その後段階を経て論理的記述を増やしていく。先に説明した通り、上記のマクロを用いるだけで攻略出来るようになっているためプログラミングに必要な構文は殆ど必要はない仕様となっている。

マクロによる攻略プログラム作成例

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "senkan.h"

void
shoot ()
{
    F009
    F005
    F003
    F007
    F010
    F012
    F011
    F013
    return;
}
```

4.5.2 次段階向けシステム

次のステップでは抽象度を下げ C 言語の構文を用いる事がプログラミング上で必要になる。しかし、初学者向けシステムの様に、ある程度の動作を行なう関数を用意してある。但しマクロとして用意していた時とは変わり、プログラミング言語の構文を殆ど使用せずに攻略することは不可能となっている。そのため、最低でも条件分岐と関数、変数の扱いを理解する必要があるように仮定してある。

sb
: 盤面情報を保持

i
: 現攻撃回数

kekka
: 前回攻撃結果格納

gx
: 縦の攻撃座標を持つ

gy
: 横の攻撃座標を持つ

at_flg
: 特殊攻撃関数実行の為の情報を格納, 真の時に特殊攻撃関数実行可能

艦船 HP を保持 (沈没しているかの確認に利用, 0 であれば沈没)

戦艦 skp 5
巡洋艦 jkp 4
駆逐艦 kkp 3
潜水艦 smp 2
輸送艦 ysp 2

盤面情報

N 0 // 未攻撃
BM 1 // 命中
BH 2 // ハズレ
SK 11 // length:5 戦艦沈没
JK 12 // length:4 巡洋艦沈没
KK 13 // length:3 駆逐艦沈没
SM 14 // length:3 潜水艦沈没
YS 15 // length:2 輸送艦沈没

引数は使用していないためそのための知識は必要が無い。下記は用意した攻撃関数である。

- random100() : ランダムに盤面を攻撃
- oneSkip() : 左上から 1 マスおきに順に撃つ
- twoSkip() : 左上から 2 マスおきに順に撃つ
- twoRight() : 左上から 2 マスおきに順に撃ち下段に下がると攻撃ポイントが 1 マス右にずれる
- threeSkip() : 左上から 3 マスおきに順に撃つ

- threeRight()：左上から 3 マスおきに順に撃ち下段に下がると攻撃ポイントが 1 マス右にずれる
- fourSkip()：左上から 4 マスおきに順に撃つ
- fourRight()：左上から 4 マスおきに順に撃ち下段に下がると攻撃ポイントが 1 つ右にずれる
- oddSkip()：縦横の座標が奇数のマスに攻撃
- evenSkip()：縦横の座標が偶数のマスに攻撃

下記は用意した特殊攻撃関数である。

- upAt()：攻撃が当たった場所から上に 4 つ攻撃する
- rightAt()：攻撃が当たった場所から右に 4 つ攻撃する
- downAt()：攻撃が当たった場所から下に 4 つ攻撃する
- leftAt()：攻撃が当たった場所から左に 4 つ攻撃する

初学者向け同様 14 個の関数を用意しているが、関数同士の動作を繋げるものは用意されておらず、学習者には関数の接続部分を記述しなければならない。

次段階向けシステム攻略プログラム作成例

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "senkan.h"

void
shoot ()
{
     if (!at_flg && kekka == BM)
        {
            bx = gx;
            by = gy;
            at_flg = 1;
        }
    else if (at_flg && kekka == BH)
        {
            at_flg = 0;
        }

    if (at_flg)
        {
            upAt();
        }
    else
        {
            random100();
        }

  return;
}
```

4.5.3 マインスイーパ

マインスイーパは Battle Ship と同じくボードゲームで、ゲームとして自己完結している。他人を介さずにゲームをプレイする事が出来る為これもまた Battle Ship と同じく攻略プログラミングとしては難易度が低めである。

マインスイーパは基本的に、9 × 9 のマスにランダム配置された 10 個の地雷のマスを避けて、全ての地雷以外のマスを開ける事が目的のゲームである。但し今回の調査に当ってルールを改変しており、盤面は 10 × 10 に変更し、地雷の数も 5 個に減らした。この変更は、盤面を広げ地雷を減らす事でゲームの難易度を下げ複雑なアルゴリズムを用いずにクリアできる様にする為である。

開けたマスには地雷に関する情報が入っており、開けられたマスの隣接する周辺 8 マスにセットされている地雷の数を返す。なので周辺に 1 つの地雷がある場合は開けたマスの情報は 1 である。この情報を利用し、ゲームを攻略するプログラムを書いてもらう事を 2 回目の演習とした。マインスイーパはコンピュータゲームの中でもポピュラーなゲームで、演習に利用する部屋の端末にもインストールされていた。そのため、このゲームの為に用意したのは攻略プログラムを動作させるプログラムだけである。このゲームでは地雷以外の 95 マスを無事に開ける事が出来ればクリアで、100 回のプレイのうち何回攻略できたのかをアルゴリズムの優位性の基準とした。

5 調査

本研究においては研究の前段階においての調査と、抽象的記述によるプログラミング環境の影響の調査の 2 種類を行なった。先ず、研究の前段階の調査では本学情報システム工学科の 2010 年度の三回生を対象に、本学のプログラミング教育カリキュラムを受けてのプログラミングの理解度を調査した。

三回生を対象にした前段階調査は合計 2 回行なっており、異なるシステムを用いてのプログラミングの演習を行なった。

その後調査結果を基に学習者が理解し辛い概念や構文を可能な限り抽象化したシステムを作成した。作成したシステムがプログラミング初学者のプログラミングにおいて、どの程度の助けになるかを 2010 年度一回生と 2011 年度一回生を対象に、演習とアンケートによる調査を行なった。

また、プログラミングに慣れた学生向けに言語構文の抽象度を下げたシステムを作成し、三回生を対象に演習を行なった。

5.1 1 回目: 三回生向け前段階調査

1 回目の調査には Python にて作成した Battle Ship を用いた。演習には 2 週間の期間を使った。内容は 1 週目にいくつかのサンプルプログラムを提示し、それを参考に指定されたアルゴリズムの実装を行なわせた。その後、レポート課題として日本語による Battle Ship の攻略アルゴリズムの提案を行なわせ、その提出を求めた。2 週目には各々が考えた攻略プログラムを実際に実装させた。2 週目の課題は各々が作成したプログラムを期日迄に提出する事とした。

5.1.1　1 回目: 三回生向け前段階アンケート結果と考察

三回生向けの演習に参加した学生は 16 名である。1 週目に行なった Battle Ship の攻略アルゴリズムの提案に関しては 5 人が攻略の為のアルゴリズムを提出しており、その中でプログラムにそのまま利用出来る程の詳細な説明が出来ていたのは 4 人である。演習終了時にプログラムを提出させたが、ゲームの攻略とまではいかないまでもある程度の意図を持った動作をするプログラムを書けていたのは 7 人である。その後、今回の演習についてのアンケートを取った。

表 1　Python の使用経験について

Python を使ったことが無い	15 人
Python を使ったことがある	1 人

表 1 を見て分かる様に、今回の演習参加者の内 Python でのプログラミング経験者は 1 人である。このことから、言語の構文規則を知らない状態からプログラミングを行なう事が出来たといえる。実際に、指定されたアルゴリズムを実装する 1 週目の講義では、学生は Python 独特の構文規則に悩まされているケースが多々見受けられた。しかし 2 週目の演習が開始した時には順応し始めている学生も見られたため、スクリプト言語によるプログラミングは理解し易いようである。

表 2　授業外でのプログラミング経験について

何もやっていない	10 人
たまに趣味で書く	5 人
仕事をした事がある	0 人
毎日書いている	1 人
日常的にシステムのロジックを考える	0 人

表 2 では学外での自発的なプログラミングを行なっているかを調べた。学生がプログラミングに対して普段どの様に捉えているかが理解できると考えたからである。この質問では 6 人の学生が学業とは関係の無い機会にてプログラミングをしていると言う結果がでている。16 人中の 6 人なので数としては多くないが、プログラミングに対しての意識は全くない訳では無いと分った。

表 3　指定アルゴリズムの実装について

アルゴリズムの実装方法が分からない	11 人
プログラムが思った通りに動かない	4 人
何とか実装できた	0 人
無駄なルーチンを実行しているが直せない	1 人
不満はない	0 人

表 3 では 11 人が指定されたアルゴリズムの実装方法が思いつかなかった事が分かる。半数以上が自身が考えていないアルゴリズムを理解することが出来ていなかった。

表 4　課題で悩んだ箇所について (複数回答可)

アルゴリズムが思いつかない	8 人
基本的な構文が分からない	6 人
エラーに悩まされる	3 人
プログラムが思い通りに動かない	2 人
他に最適なアルゴリズムがあると思うが分からない	2 人

そして、表 4 では今回の演習で悩んだ箇所について答えさせた。アルゴリズムが分からない学生は 9 人、Python の構文が理解できない学生は 6 人という結果から、半数近くの学生はアルゴリズムか構文、もしくはその両方がネックとなっている事が分かる。しかし、今回の演習だけでは単に経験の無い言語でのプログラミングに戸惑った事がプログラミングに影響を全く与えていないとは良い切れず、C 言語プログラミングの理解度を調べる必要がある。

5.2　2 回目: 三回生向け前段階調査

2 回目: 三回生向けの演習に参加した学生は 13 名である。2 回目の演習は 1 回目とゲームを変え、マインスイーパの攻略を課題とした。講義の進め方は 1 週目からプログラミングを行なわせた。かなりポピュラーなゲームの為殆どの学生がルールを知っており、攻略方法もある程度理解していると考えたからである。その日の課題として攻略アルゴリズムの提案を前回同様に行なわせ、2 週目の演習で作成したプログラムの提出を行なわせた。

5.2.1　2 回目: 三回生向け前段階調査アンケート結果と考察

今回の演習では初回の演習とは違い 1 年前に講義で習った C 言語での演習という事で演習に参加している学生は始めからプログラミングを行なう事が出来ていた。しかし、1 週目は順調に見えたものの 2 週目の課題提出の時には前回と同じ学生が課題を提出しているような状況である。

表 5　C 言語と Python のどちらが良かったか

C 言語	7 人
どちらも難しい	5 人
Python	2 人

表 5 では今回 2 つの言語によってプログラミングを行なった感想として、どちらの言語が好ましかったかを調査した。結果として学習した事のある言語を好ましいと思う傾向にある事が分かる。この事を考慮すると、授業で習う言語をベースに学習を進めて行く事が学習者の理解の助けになると考えられる。

表 6　プログラミングで分からなかったこと (複数回答可)

条件分岐を使うことに気づかなかった	1 人
考えている条件がプログラミング出来ない	6 人
繰り返しを使う事に気付かなかった	2 人
思うように繰り返しが出来無い	5 人
ポインタの意味が分からない	1 人
引数の使い方が分からない	4 人

表 7　引数が使えない原因

引数が何かが分からない	3 人
引数が何をするものか分からない	2 人
引数についての解説が不十分	0 人
良い使い方が分からない	2 人
特に問題はない	6 人

表 6、7 から今回の演習参加者に引数に関して理解出来ていない学生が居ることが分かった。引数を理解できていない事が今回の演習に影響を及ぼす訳ではないが、プログラミングを行なう上で非常に重要な要素ある事には間違いがない。その為引数の使い方と、その意味を学習者が理解できる要素が必要であると考えられる。

表 8　課題で悩んだ箇所について (複数回答可)

エラーの意味が分からない	2 人
攻略の手順が分からない	3 人
攻略の手順が説明出来ない	3 人
構文の使い方が分からない	4 人
確率の問題が出来ない	1 人
計算は分かるがプログラムが書けない	4 人

さらに、アンケートで今回の演習について理解出来なかった部分を調査したところ、 Python を用いての演習時程の人数ではないが構文によって詰まる学生が居た。これは表 8 からその傾向が見られる。

5.3 三回生向け演習まとめ

前段階調査では、言語を変えて演習を行なう事で学習者のプログラミングへの取り組みを調査する事が出来た。プログラミングの教育を受けた事によるメリットは、言語の扱い方を学んだ為に順応が早かったことである。そして、基礎的な構文を明確にではないが把握している様子が見られた。しかし、引数の扱いや基本的な論理的記述があまり出来ておらず、自分の考えをプログラムに反映させる事に苦労していた。そのため、先行研究 [4] でも言われいる通り初学者には先ずアルゴリズムを意識させる事が重要であると考えられる。その上で論理的な記述方法を学習することが出来れば良いと考えられる。

その為には先ず、多くの学生が詰っていた原因である構文の問題と、プログラミングを難しくする要因である関数や、条件分岐、繰り返しといった構文も学習者が気にしなくてもプログラミングを行なえる環境が必要だと考えられる。

5.4　2010 年度 一回生向け演習

2 回の 三回生向け演習による調査によって得られた結果から、初学者に取って重要な要素であろう抽象的にプログラミングを行なう環境を用意し、一回生向けに演習を行なった。一回生は C 言語を講義で学習しないが、用意したシステムでは C 言語を用いて動作するようにした。抽象的に記述することで動作するようにする以上は使用する言語は関係無く、そうした場合、より低レイヤーでの記述が可能な言語を選択する事が将来的な学習には役立つと考えた為である。

演習内容は、基本的に 三回生向けに行なった演習と同じで、学習システムを使ったプログラミングを行なわせ、課題として日本語による攻略プログラムの提案を行なわせた。そして、2 週目に学習者が納得の行くプログラムを提出させた。授業の終わりにはアンケートを行ない、演習に用いたシステムについての感想やプログラミングへの意識を調査した。

5.4.1　2010 年度 一回生向け演習に対するアンケートと考察

2010 年度 一回生向け演習では 35 名が参加した。集計したアンケートの分析を行なった。その際に設問毎の関係性に着目した。

表 9　大学入学までのプログラミング経験

入学するまで知らなかった	9 人
知っていた	14 人
独学で勉強した	4 人
指導を受けたことがある	7 人
プログラムを書く仕事をした	1 人

表 10　プログラムの動作を説明できるか?

どのように動いているか分からない	11 人
他人に説明はできないが分かる	21 人
他人に説明できる	3 人

表 11　どのようにプログラミングしたか?

適当に並べた	10 人
手当たり次第に試した	14 人
少しずつ修正して理想に近づけた	6 人
攻略方法を考え設計に沿うように実装した	5 人

表 9 でプログラミング未経験者は 23 人となっている。表 10 より、その内 9 人は自分の書いたプログラムの動作が理解できず、残り 14 人は他人に説明することが出来ないとしている。

更に表 9 と表 11 の関係性に着目して集計したところ、プログラミング未経験者の内 8 人が並べただけで 10 人が総当たり、3 人がフィードバックをプログラムの修正に使用、残る 2 人がコーディング前に設計を行ったという結果が得られた。こえらからプログラミング未経験者に見られる傾向として論理的に考察、理解が出来ていないことが伺える。

表 12　ゲームのルールを理解しているか?

最後まで良く分からなかった	11 人
ゲームをプレイして理解した	19 人
説明だけで理解した	2 人
ルールは既に知っていた	1 人
実際にゲームをプレイしたことがある	2 人

演習に用いたゲームのルールに関し、表 12 から 11 名もの学生が完全には理解できていない事が分かる。学習システムとしては、この様なプログラミングの本質と関係が無い問題に躓くと学習の妨げになってしまう為に望ましくない。

表 13　演習は難しかったか?

自分のやっている事が分からなかった	8 人
思ったとおりに動いてくれなかった	8 人
完璧では無いが思った通りに動いた	12 人
並べるだけなので非常に簡単だった	7 人

そして、演習内容について何をしているのか分からない学生が 8 名いた。残りの学生は問題なくルールを把握していたが、ルールの把握の様な問題はプログラミング以前の問題で、最も解消するべき問題であると考えられる。

5.4.2　2010 年度 一回生向け演習に対するまとめ

一回生向け演習の課題は学習者が、構文エラー等の問題で躓きアルゴリズムを考えるに至らない状況を回避できるアプローチを考え、実装することである。三回生向けの演習によって構文が学習者の障害となっていることが確認でき、一回生の演習では構文の緩和によって学習者がフィードバックを得易くなることを確認できた。このことから抽象的記述による、容易なプログラムの実行環境は学習のルーチンの点で有効であることが判明した。

しかし別の問題も判明している。それは、ゲームを用いる事によりルールの理解が出来ず躓く問題や、現状ではアルゴリズムを考えなくても目的を達成することが出来ることである。ただ、プログラミングを学習するための最初の段階と捉えれば、アルゴリズムの自由度が低く攻略するだけであればマクロひとつで終わる事も許容できる。問題はゲームのルールを理解できないことである。これには、解説を行なっているページの改善や授業説明の工夫によって改善出来ると考えられる。

5.5　2011 年度 一回生向け演習

次に 一回生を対象にした演習について述べる。これは去年度に同様の演習を行っており、マクロによるプログラミングの効果を去年度と比較する為に行った。その為演習についての詳細は変更していない。しかし、演習後のアンケート以外に学生の学習への意欲を図る為、演習最終日に自由退室を許可し演習に取り組んだ時間を図ることにした。また、追加のアンケートを後日課題として自分がプログラミングで理解出来ない部分と、プログラミングの講義に何を求めるかという自由記述アンケートを行った。

5.5.1 2011 年度 一回生向け演習に対するアンケートと考察

一回生向けの演習に参加した学生は 46 名である。使用したプログラムには変更点は無く、2010 年度の演習と同じ内容である。

表 14　大学入学までのプログラミング経験

入学するまで知らなかった	15 人
知っていた	21 人
独学で勉強した	3 人
指導を受けたことがある	7 人
プログラムを書く仕事をした	0 人

表 15　プログラムの動作を説明できるか?

どのように動いているか分からない	17 人
他人に説明はできないが分かる	23 人
他人に説明できる	6 人

表 16　どのようにプログラミングしたか?

適当に並べた	11 人
手当たり次第に試した	18 人
少しずつ修正して理想に近づけた	12 人
攻略方法を考え設計に沿うように実装した	5 人

表 17　ゲームのルールを理解しているか?

最後まで良く分からなかった	14 人
ゲームをプレイして理解した	20 人
説明だけで理解した	2 人
ルールは既に知っていた	2 人
実際にゲームをプレイしたことがある	8 人

表 18　演習は難しかったか?

自分のやっている事が分からなかった	12 人
思ったとおりに動いてくれなかった	13 人
完璧では無いが思った通りに動いた	16 人
並べるだけなので非常に簡単だった	5 人

学習者の演習に対する評価と演習中の行動、思考の傾向はこのアンケートにより、同じ傾向にある事が分かる。その為 2010 年度 一回生向け演習時に判明した問題点の改善が必要であると考えられる。

5.5.2　2011 年度 一回生向け演習に対するまとめ

昨年度同様殆どの学生が自身の攻撃プログラムの動作を確認していた。この演習にて学生に必要とするのは、ゲームを攻略するためのアルゴリズムと C 言語で作成されたプログラムをコンパイル、実行し、動作の修正を行なう事である。その為にはコードを書かなければいけないが、マクロによってコードは簡略化されている。マクロには各攻撃同士を繋ぐコードが記述されている為、記述の順番に関係無く各マクロの動作は保証される。その為に記述順によるエラーは発生しない仕組みとなっている。これによって学生の考えを容易にプログラムとして反映させることが出来ている。プログラミング未経験者が 7 割程度を占める演習で 2010 年度、2011 年度共に学習者のプログラミングは実行、プログラム修正まで辿りつく事が出来た。しかし、ゲーム内容の理解が出来なかった学習者がいた為、ゲームのプレイ時にルールの解説を行うなどのサポートが必要であった。

5.6　2011 年度 三回生向け演習

三回生にはあらかじめ用意された関数を用いての演習を行なわせた。大垣講師が担当する講義で情報ネットワーク 2 の授業時間を用いて演習を行なった。対象者は講義を受講している学生が対象である。この演習ではアンケートの収集は行なっておらず、レポート課題の一環としてプログラムの提出を学生に求めた。期間は 2 週間である。今回の目的はプログラミング学習者はマクロを用いずにゲームの攻略プログラミングが可能かを調べる事にあった。学習者がプログラミングを既に学習している事を前提としている為に今回は三回生を対象に行なった。

5.6.1　2011 年度 三回生向け演習に対する考察

授業を受講している内、有効な学生[*3]は 28 名である。学生の演習結果は芳しくなかった。演習の課題提出者数は 2 名である。その二つともが事前に用意された関数を用いずにプログラミングされたもので、これは本システムを用いての学習の最終目的である。

他の演習実施者は 一回生向けに行なった演習と大きく結果が異なり、学習者のプログラム実行までシステムによって誘導出来なかった。原因として、学生が今回の補助システムを理解出来なかった事が考えられる。学習者が容易に使用する事が出来無かった事から、関数の使い易さや、ゲームの攻略プログラミングの難度を改善する必要がある。

5.6.2　2011 年度 三回生向け演習に対するまとめ

先ず、プログラミング演習を既に履修している 三回生の演習についての考察だが、プログラミング能力のバラツキに関しては、去年度の調査結果からある程度の予測は出来ていた。しかし今年度の調査は、必修科目のプログラミング演習で習う C 言語を用いたのにも関わらず去年の調査と比べ、予想と反した結果になっている。Python での演習を行った時は習った事のない言語による構文問題が原因の大きな割り合いを占めていると考えていたが、違った。

攻撃関数を全て学習者が実装する手法では、学生が自分の考えるアルゴリズムを実装出来ていなかった。その為、今回の演習ではこちらである程度の攻撃関数を作っており、尚且つ複雑さの排除の為に攻撃関数を使うのに引数を必要としない。その為、学生が実際に行なうプログラミングは自分が使用する関数同士がうまく機能するように条件付けや繰り返しを行なうだけである。今回の条件で学生がプログラミング出来なかったという事は、まだ学生が一度に考えるべき範囲が広すぎたと考えられる。もしくは、未だ学生が扱える構文を減らす必要があるようにも考えられる。この演習でとり扱った構文は、演習を受けた学習者にとって範囲が広かったようである。

今回の調査からは、プログラミング教育において度々問題視されている構文問題は要因の一つではあるが、プログラミングの学習の壁となっているのは他に大きな要因があることが分かる。

[*3] 一度以上授業のレポートを提出している学生

6 考察とまとめ

この章では研究の目的の達成度評価を行ない、研究のまとめを行なう。本研究の目的は次の 3 点であった。

- プログラミング初学者やプログラミングが出来ない人がプログラミングという行為を経験する
- 学習者が実際に動作するプログラムを容易に書ける環境を構築する
- 構築した環境を実際に学習者に適用し、その教育効果を調査する

先ず、プログラミングの初学者がプログラミングという行為を経験する事について、一回生の演習では 2010 年度、2011 年度の両方において殆どの学生が自身の攻撃プログラムの動作を確認している。この事からプログラミング未経験の学習者がプログラムの作成、実行し、動作確認を行なえる所までが本システムによって可能になっている事が分かる。

次に、学習者が実際に動作するプログラムを容易に書ける環境の構築について考察を行なう。プログラムを容易に書ける状態というのは、構文のエラーによる躓きを無くし、プログラミング学習者が理解出来ていない概念をブラックボックス化して扱える状態である。一回生向けに行なった演習内容では、抽象化のレベルが学習者の思考レベルにあっていた為、問題無くプログラムを書く事が出来ていた。しかし、三回生向けに行なった演習では学習者の思考レベルにあっていなかった為、学習者が演習課題をこなす事が出来なかった。この演習にて学生が理解すべき事柄はゲームを攻略するためのアルゴリズムと、C 言語で作成されたプログラムをコンパイル、実行する所までである。そのためにはコードを書かなければいけないが、マクロによってコードは簡略化されている。マクロには各攻撃同士を繋ぐコードが内包されているため、記述の順番に関係無く各マクロの動作は保証される。そのために記述順によるエラーは発生しない仕組みである。これによって学生の考えを容易にプログラムとして反映させることが出来ている。

これらにより、言語の構文とプログラミングにおける概念の抽象化自体は学習者がプログラミングを行なう助けとなっていた事が分かる。その為、学生の思考レベルに合わせ、抽象化のレベルを詳細に調整出来る様にする事が今後の課題と言える。

最後に教育効果の調査について、今回の演習で用いた機能について、学生からの評価は良く更にこのシステムで、自分で C 言語のコードを記述する学生が出てきた。こういった学生が最初に手を出したのが条件分岐で、デフォルトで用意されているマクロの実行に条件を付けてマクロを実行する時としない場合を作り出そうとしていた。確かに繰り返しよりも条件分岐の方が概念として分かりやすく感じられ、学生からもイメージし易い。なので、マクロを利用した状態で条件分岐を考えさせる課題を次のステップとして用意すると良いと考えている。また、関数や変数をいつ学生に見せるかも考える必要があり、これからの方針とその可能性をもう少し具体的に詰める事が必要である。

また 三回生向けに行なった演習では、演習としての難易度が高く、ゲームルールの把握から始めなければいけないのは時間的な制約もあり学習者の負担になりかねないことが分かった。その為あくまでも段階を経て学習をさせることを念頭に置き、マクロによるプログラミングでゲームの把握とアルゴリズムの醸成を狙う事が必要である。学生に開示する情報の調整とともに改善が必要である事が分かった。

ゲームプログラムの仕様の詳細化については、集計したアンケート結果から Battle Ship ゲームそのものについてのルールを理解していない学生が多く見られた。そういった学生が理解できるようにゲームやプログラミングを行なう上での仕様を詳細に公開する必要があると考えた。さらに、三回生向けの学生の演習結果を見た限り関数に関する詳細なリファレンスが無かったことが原因とも考えられる。そのため、画像やデモを用いての解説を行なう等の措置を行なう事を課題として考えている。

そして、今回は行なう事が出来なかったがプログラミングの講義を受講した学生を対象にマクロによるプログラミングと機能制限を行なったプログラミングの両方を順に実施し、その結果を調査する事が必要である。

謝辞

本論文を執筆するにあたり、能勢 和夫教授には主査として本論文の細部に渡り様々なご指導を頂いた事に感謝致します。近江 和生教授、並びに前川 佳徳教授には副査として発表の場を通してご助言を頂いた事で研究の方向性を確認し、修正を重ねてこられた事に感謝致します。大垣 斉先生には本研究の実施の機会を与えて頂き、その遂行にあたってご指導を頂いたことに深く感謝致します。筆者が学部生の頃から気に掛けて頂き今尚プログラミングの指導をしてくださっている水野 貴弘氏、配属されてすぐの筆者に進むべき道を示し一からプログラミングの指導をして下さった小山 翔平氏の両名には感謝しております。教育研究について学生への指導方法を考察するに当り、研究の一環に協力して頂いた津川 翔丞氏、坂田 龍之介氏、北田 宏樹氏、小川 淳矢氏、高橋 徹氏、大橋 侑弥氏には感謝しております。また、研究の調査を行う為にアンケート調査に協力して下さった学生の皆様にも同じく感謝しております。そして、team.andrew ML に参加してくださっている皆様方には御助言を賜りました。最後になりましたが博士前期課程に進む事を承諾し、見守り続けてくれた両親に深く深く感謝致します。本研究によりこれからのプログラミング教育へどの程度の影響を与えられるかは甚だ疑問ではありますが、皆様のお力添えにより論文という形を得ることが出来ました。筆者に機会と御助言を下さった皆様にこの場を借りて深く感謝の意を表し、謝辞と致します。

参考文献

[1] Jeff Atwood. Why can't programmers.. program? http://www.codinghorror.com/blog/2007/02/why-cant-programmers-program.html, Feb 2007.

[2] 西尾泰和. プログラミング教育に関する考察. http://www.nishiohirokazu.org/blog/2006/07/post_95.html, July 2006.

[3] 楠正憲. プログラミング教育が機能しない背景. http://d.hatena.ne.jp/mkusunok/20080827/progedu, 08 2008.

[4] 杉浦学, 松沢芳昭, 岡田健, 大岩元. アルゴリズム構築能力育成の導入教育: 実作業による概念理解に基づくアルゴリズム構築体験とその効果. 情報処理学会論文誌, 第 49 巻, pp. 3409 – 3427, 2008.

[5] 尾崎浩和, 富永浩之, 山崎敏範. ボードゲームの戦略プログラミングを題材とした java 演習の支援システムの開発. 情報処理学会研究報告 コンピュータと教育研究会報告, 第 108 巻, pp. 1–8, 2006.

[6] 辻大地. プログラミング入門支援ゲームの試作と評価. 大阪産業大学修士論文, 2010.

[7] 江澤義典. Javascript プログラミングと情報教養教育. 関西大学情報学部紀要「情報研究」, 第 26 巻, pp. 1–10, 2007.

[8] 情報処理学会情報処理教育委員会. 日本の情報教育・情報処理教育に関する提言 2005. http://www.ipsj.or.jp/12kyoiku/proposal-20051029.html, 10 2005.

[9] 河村一樹. JavaScript による情報教育入門. 大学教育出版, 2011.

付録 A　三回生向け前段階調査 Battle Ship ゲーム

A.1　Practice

A.1.1　practice.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

################################################################################
# Battleship game module
#
# ----------盤面情報の初期化----------------------------------------------------
# initialization              盤面の作成
#         - dispositionOfShip 艦船の配置
#
# ----------盤面情報の表示----------------------------------------------------
# displaySubmarine            盤面の座標を表示
#         - attackStatement   各艦船の状態を表示
################################################################################

import sys
import random
#from shoot import shoot

# 盤面情報
TATE        = 10
YOKO        = 10

# 奇数 & 偶数
ODD_NUMBER   = 7
EVEN_NUMBER  = 8

# 盤面情報
N           = 0
HIT         = 1
MISS        = 2

# 艦船番号
WARSHIP     = 11
CRUISER     = 12
DESTROYER   = 13
SUBMARINE   = 14
```

```
TRANSPORTS  = 15

BATTLESHIP  = [TRANSPORTS, SUBMARINE, DESTROYER, CRUISER, WARSHIP]

# 艦船 HP
warshipP    = 5
cruiserP    = 4
destroyerP  = 3
submarineP  = 3
transportsP = 2

submarineFlg = 0

def initialization ():
        global submarineFlg
        submarine = [[ N for i in range (0, TATE)] for j in range (0, YOKO)]
        if submarineFlg == 0:
                submarineFlg = 1
                return submarine [:]
        for ship in [warshipP, cruiserP, destroyerP, submarineP, transportsP]:
                battleShip = BATTLESHIP.pop()
                dispositionOfShip (ship, battleShip, submarine)
        return submarine [:]

def dispositionOfShip (ship, battleShip, submarine):
        shipLength = range (0, ship)
        if random.randint (ODD_NUMBER, EVEN_NUMBER) == EVEN_NUMBER:
                while 1:
                        judge = 0
                        y = random.randint (0, TATE-1)
                        x = random.randint (0, YOKO-1 - ship)
                        # 未処理マスの確認
                        for i in shipLength:
                                judge += submarine [y][x+i]
                        # 艦船の配置
                        if judge == N:
                                for i in shipLength:
                                        submarine [y][x+i] = battleShip
                                return
        else :
                while 1:
                        judge = 0
                        y = random.randint (0, TATE-1 - ship)
```

```
                        x = random.randint (0, YOKO-1)
                        # 未処理マスの確認
                        for i in shipLength:
                                judge += submarine [y+i][x]
                        # 艦船の配置
                        if judge == N:
                                for i in shipLength:
                                        submarine [y+i][x] = battleShip
                                return

def displaySubmarine (table, submarine):
        sys.stdout.write ("   1 2 3 4 5 6 7 8 9 10\n")
        sys.stdout.write ("  ---------------------\n")

        # 攻撃済情報の公開
        for y in range (TATE):
                sys.stdout.write ("%2d" % (y+1))
                for x in range (YOKO):
                        sys.stdout.write ("|")
                        attackStatement (y, x, table, submarine)
                sys.stdout.write ("|\n")
        sys.stdout.write ("  ---------------------\n")

def attackStatement (y, x, table, submarine):
        if   submarine [y][x] == HIT:
                if   table [y][x] == WARSHIP:
                        if warshipP    == 0:
                                sys.stdout.write ("5")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == CRUISER:
                        if cruiserP    == 0:
                                sys.stdout.write ("4")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == DESTROYER:
                        if destroyerP  == 0:
                                sys.stdout.write ("3")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == SUBMARINE:
                        if submarineP  == 0:
                                sys.stdout.write ("3")
```

```
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == TRANSPORTS:
                        if transportsP == 0:
                                sys.stdout.write ("2")
                        else :
                                sys.stdout.write ("1")
        elif submarine [y][x] == MISS:
                sys.stdout.write ("x")
        else :
                sys.stdout.write (" ")

def leftoverEnemy ():
        if (warshipP   <= 0 and cruiserP   <= 0 and
            destroyerP <= 0 and submarineP <= 0 and transportsP <= 0):
                sys.stdout.write ("全艦撃沈!")
                return True
        sys.stdout.write ("残敵 = ")
        if warshipP    > 0: sys.stdout.write ("戦艦, ")
        if cruiserP    > 0: sys.stdout.write ("巡洋艦, ")
        if destroyerP  > 0: sys.stdout.write ("駆逐艦, ")
        if submarineP  > 0: sys.stdout.write ("潜水艦, ")
        if transportsP > 0: sys.stdout.write ("輸送艦")
        sys.stdout.write ("\n")

def hitJudgement (table, submarine):
        noneAttack = submarine [y][x]
        damage     = table     [y][x]

        global warshipP, cruiserP, destroyerP, submarineP, transportsP

        if noneAttack != N:
                return MISS
        else:
                if   damage == WARSHIP:
                        submarine [y][x] = HIT
                        warshipP        -= 1
                        return WARSHIP
                elif damage == CRUISER:
                        submarine [y][x] = HIT
                        cruiserP        -= 1
                        return CRUISER
                elif damage == DESTROYER:
```

```
                        submarine [y][x] = HIT
                        destroyerP      -= 1
                        return DESTROYER
                elif damage == SUBMARINE:
                        submarine [y][x] = HIT
                        submarineP      -= 1
                        return SUBMARINE
                elif damage == TRANSPORTS:
                        submarine [y][x] = HIT
                        transportsP      -= 1
                        return TRANSPORTS
        submarine [y][x] = MISS
        return MISS

def hitDisplay (result):
        if   result == MISS:
                sys.stdout.write ("ハズレ!\n")
                return MISS
        elif WARSHIP <= result <= TRANSPORTS:
                sys.stdout.write ("命中!")
                if   result == WARSHIP    and warshipP    <= 0:
                        sys.stdout.write ("戦艦沈没!\n")
                        return WARSHIP
                elif result == CRUISER    and cruiserP    <= 0:
                        sys.stdout.write ("巡洋艦沈没!\n")
                        return CRUISER
                elif result == DESTROYER  and destroyerP  <= 0:
                        sys.stdout.write ("駆逐艦沈没!\n")
                        return DESTROYER
                elif result == SUBMARINE  and submarineP  <= 0:
                        sys.stdout.write ("潜水艦沈没!\n")
                        return SUBMARINE
                elif result == TRANSPORTS and transportsP <= 0:
                        sys.stdout.write ("輸送艦沈没!\n")
                        return TRANSPORTS
                sys.stdout.write ("\n")
                return HIT


INITIAL_SHOOT     = 0
SECOND_SHOOT      = 1
FINISH_SHOOT      = 100
```

```
# 座標軸
y = 0
x = 0

submarine = initialization ()
table     = initialization ()

leftoverEnemy ()

#shoot (INITIAL_SHOOT, submarine, MISS)
displaySubmarine (table, submarine)

while True:
        try:
                x = (int (raw_input("攻撃座標入力 X: ")) -1)
                y = (int (raw_input("攻撃座標入力 Y: ")) -1)
                if (0 <= x and x < 10) and (0 <= x and x < 10):
                        break
        except ValueError:
                print "入力が不正です"

result_tmp = hitJudgement (table, submarine)
displaySubmarine (table, submarine)
result = hitDisplay(result_tmp)
print "攻撃回数: %d" % (INITIAL_SHOOT+1)

for i in range(SECOND_SHOOT, FINISH_SHOOT):
        while True:
                try:
                        x = (int (raw_input("攻撃座標入力 X: ")) -1)
                        y = (int (raw_input("攻撃座標入力 Y: ")) -1)
                        if ((0 <= x and x < 10) and (0 <= x and x < 10)) and \
                                        (submarine[y][x] == 0):
                                break
                except ValueError:
                        print "Error: 入力が不正です"
        result_tmp = hitJudgement (table, submarine)
        displaySubmarine (table, submarine)
        result = hitDisplay(result_tmp)
        print "攻撃回数: %d" % (i+1)
        # ゲーム終了
        if leftoverEnemy ():
                break
```

```
sys.stdout.write ("攻撃回数 = %d\n" % i+1)
```

A.2 Verification

A.2.1 verification.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

################################################################################
# Battleship game module
#
# ----------盤面情報の初期化----------------------------------------------------
# initialization              盤面の作成
#         - dispositionOfShip 艦船の配置
#
# ----------盤面情報の表示----------------------------------------------------
# displaySubmarine            盤面の座標を表示
#         - attackStatement   各艦船の状態を表示
################################################################################

import sys
import random
from shoot import shoot

# 盤面情報
TATE        = 10
YOKO        = 10

# 奇数 & 偶数
ODD_NUMBER   = 7
EVEN_NUMBER  = 8

# 盤面情報
N           = 0
HIT         = 1
MISS        = 2

# 艦船番号
WARSHIP     = 11
CRUISER     = 12
DESTROYER   = 13
SUBMARINE   = 14
TRANSPORTS  = 15
```

```
BATTLESHIP  = [TRANSPORTS, SUBMARINE, DESTROYER, CRUISER, WARSHIP]

# 艦船 HP
warshipP    = 5
cruiserP    = 4
destroyerP  = 3
submarineP  = 3
transportsP = 2

submarineFlg = 0

def initialization ():
        global submarineFlg
        submarine = [[ N for i in range (0, TATE)] for j in range (0, YOKO)]
        if submarineFlg == 0:
                submarineFlg = 1
                return submarine [:]
        for ship in [warshipP, cruiserP, destroyerP, submarineP, transportsP]:
                battleShip = BATTLESHIP.pop()
                dispositionOfShip (ship, battleShip, submarine)
        return submarine [:]

def dispositionOfShip (ship, battleShip, submarine):
        shipLength = range (0, ship)
        if random.randint (ODD_NUMBER, EVEN_NUMBER) == EVEN_NUMBER:
                while 1:
                        judge = 0
                        y = random.randint (0, TATE-1)
                        x = random.randint (0, YOKO-1 - ship)
                        # 未処理マスの確認
                        for i in shipLength:
                                judge += submarine [y][x+i]
                        # 艦船の配置
                        if judge == N:
                                for i in shipLength:
                                        submarine [y][x+i] = battleShip
                                return
        else :
                while 1:
                        judge = 0
                        y = random.randint (0, TATE-1 - ship)
                        x = random.randint (0, YOKO-1)
                        # 未処理マスの確認
```

```
                        for i in shipLength:
                                judge += submarine [y+i][x]
                        # 艦船の配置
                        if judge == N:
                                for i in shipLength:
                                        submarine [y+i][x] = battleShip
                                return

def displaySubmarine (table, submarine):
        sys.stdout.write ("   1 2 3 4 5 6 7 8 9 10\n")
        sys.stdout.write ("  ---------------------\n")

        # 攻撃済情報の公開
        for y in range (TATE):
                sys.stdout.write ("%2d" % (y+1))
                for x in range (YOKO):
                        sys.stdout.write ("|")
                        attackStatement (y, x, table, submarine)
                sys.stdout.write ("|\n")
        sys.stdout.write ("  ---------------------\n")

def attackStatement (y, x, table, submarine):
        if   submarine [y][x] == HIT:
                if   table [y][x] == WARSHIP:
                        if warshipP    == 0:
                                sys.stdout.write ("5")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == CRUISER:
                        if cruiserP    == 0:
                                sys.stdout.write ("4")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == DESTROYER:
                        if destroyerP  == 0:
                                sys.stdout.write ("3")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == SUBMARINE:
                        if submarineP  == 0:
                                sys.stdout.write ("3")
                        else :
                                sys.stdout.write ("1")
```

```
                elif table [y][x] == TRANSPORTS:
                        if transportsP == 0:
                                sys.stdout.write ("2")
                        else :
                                sys.stdout.write ("1")
        elif submarine [y][x] == MISS:
                sys.stdout.write ("x")
        else :
                sys.stdout.write (" ")

def leftoverEnemy ():
        if (warshipP   <= 0 and cruiserP   <= 0 and
            destroyerP <= 0 and submarineP <= 0 and transportsP <= 0):
                sys.stdout.write ("全艦撃沈!\n")
                return True
        sys.stdout.write ("残敵 = ")
        if warshipP    > 0: sys.stdout.write ("戦艦, ")
        if cruiserP    > 0: sys.stdout.write ("巡洋艦, ")
        if destroyerP  > 0: sys.stdout.write ("駆逐艦, ")
        if submarineP  > 0: sys.stdout.write ("潜水艦, ")
        if transportsP > 0: sys.stdout.write ("輸送艦")
        sys.stdout.write ("\n")

def hitJudgement (table, submarine):
        noneAttack = submarine [y][x]
        damage     = table     [y][x]

        global warshipP, cruiserP, destroyerP, submarineP, transportsP

        if noneAttack != N:
                return MISS
        else:
                if   damage == WARSHIP:
                        submarine [y][x] = HIT
                        warshipP        -= 1
                        return WARSHIP
                elif damage == CRUISER:
                        submarine [y][x] = HIT
                        cruiserP        -= 1
                        return CRUISER
                elif damage == DESTROYER:
                        submarine [y][x] = HIT
                        destroyerP      -= 1
```

```
                        return DESTROYER
                elif damage == SUBMARINE:
                        submarine [y][x] = HIT
                        submarineP      -= 1
                        return SUBMARINE
                elif damage == TRANSPORTS:
                        submarine [y][x] = HIT
                        transportsP      -= 1
                        return TRANSPORTS
        submarine [y][x] = MISS
        return MISS

def hitDisplay (result):
        if   result == MISS:
                sys.stdout.write ("ハズレ!\n")
                return MISS
        elif WARSHIP <= result <= TRANSPORTS:
                sys.stdout.write ("命中!\n")
                if   result == WARSHIP    and warshipP    <= 0:
                        sys.stdout.write ("戦艦沈没!\n")
                        return WARSHIP
                elif result == CRUISER    and cruiserP    <= 0:
                        sys.stdout.write ("巡洋艦沈没!\n")
                        return CRUISER
                elif result == DESTROYER  and destroyerP  <= 0:
                        sys.stdout.write ("駆逐艦沈没!\n")
                        return DESTROYER
                elif result == SUBMARINE  and submarineP  <= 0:
                        sys.stdout.write ("潜水艦沈没!\n")
                        return SUBMARINE
                elif result == TRANSPORTS and transportsP <= 0:
                        sys.stdout.write ("輸送艦沈没!\n")
                        return TRANSPORTS
                sys.stdout.write ("\n")
                return HIT


INITIAL_SHOOT     = 0
SECOND_SHOOT      = 1
FINISH_SHOOT      = 100

submarine = initialization ()
table     = initialization ()
```

```
# 座標軸
y = 0
x = 0

leftoverEnemy ()

y, x = shoot (INITIAL_SHOOT, submarine, MISS, y, x)
displaySubmarine (table, submarine)
sys.stdout.write (raw_input("Press any key to continue...\n"))

result_tmp = hitJudgement (table, submarine)
displaySubmarine (table, submarine)
result = hitDisplay(result_tmp)
print "攻撃 %3d 回目, X:%d Y:%d" % (1, x, y)


for i in range(SECOND_SHOOT, FINISH_SHOOT):
        sys.stdout.write (raw_input("Press any key to continue...\n"))
        y, x = shoot (i, submarine, result, y, x) # test:
        result_tmp = hitJudgement (table, submarine)
        displaySubmarine (table, submarine)
        result = hitDisplay(result_tmp)
        print "攻撃 %3d 回目, X:%d Y:%d" % (i+1, x, y)
        # ゲーム終了
        if leftoverEnemy ():
                break
sys.stdout.write ("攻撃回数 = %d\n" % (i+1))
```

A.3 Score

A.3.1 score.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

################################################################################
# Battleship game module
#
# ----------盤面情報の初期化----------------------------------------------------
# initialization              盤面の作成
#         - dispositionOfShip 艦船の配置
#
# ----------盤面情報の表示----------------------------------------------------
# displaySubmarine            盤面の座標を表示
#         - attackStatement   各艦船の状態を表示
################################################################################

import sys
import random
from shoot import shoot

# 盤面情報
TATE        = 10
YOKO        = 10

# 奇数 & 偶数
ODD_NUMBER   = 7
EVEN_NUMBER  = 8

# 盤面情報
N           = 0
HIT         = 1
MISS        = 2

# 艦船番号
WARSHIP     = 11
CRUISER     = 12
DESTROYER   = 13
SUBMARINE   = 14
TRANSPORTS  = 15
```

```
submarineFlg = 0
sum          = 0

def initialization ():
        global submarineFlg
        submarine = [[ N for i in range (0, TATE)] for j in range (0, YOKO)]
        if submarineFlg == 0:
                submarineFlg = 1
                return submarine [:]
        for ship in [warshipP, cruiserP, destroyerP, submarineP, transportsP]:
                battleShip = BATTLESHIP.pop()
                dispositionOfShip (ship, battleShip, submarine)
        return submarine [:]

def dispositionOfShip (ship, battleShip, submarine):
        shipLength = range (0, ship)
        if random.randint (ODD_NUMBER, EVEN_NUMBER) == EVEN_NUMBER:
                while 1:
                        judge = 0
                        y = random.randint (0, TATE-1)
                        x = random.randint (0, YOKO-1 - ship)
                        # 未処理マスの確認
                        for i in shipLength:
                                judge += submarine [y][x+i]
                        # 艦船の配置
                        if judge == N:
                                for i in shipLength:
                                        submarine [y][x+i] = battleShip
                                return
        else :
                while 1:
                        judge = 0
                        y = random.randint (0, TATE-1 - ship)
                        x = random.randint (0, YOKO-1)
                        # 未処理マスの確認
                        for i in shipLength:
                                judge += submarine [y+i][x]
                        # 艦船の配置
                        if judge == N:
                                for i in shipLength:
                                        submarine [y+i][x] = battleShip
                                return
"""
```

```
def displaySubmarine (table, submarine):
        sys.stdout.write ("   1 2 3 4 5 6 7 8 9 10\n")
        sys.stdout.write ("  --------------------\n")

        # 攻撃済情報の公開
        for y in range (TATE):
                sys.stdout.write ("%2d" % (y+1))
                for x in range (YOKO):
                        sys.stdout.write ("|")
                        attackStatement (y, x, table, submarine)
                sys.stdout.write ("|\n")
        sys.stdout.write ("  --------------------\n")

def attackStatement (y, x, table, submarine):
        if   submarine [y][x] == HIT:
                if   table [y][x] == WARSHIP:
                        if warshipP    == 0:
                                sys.stdout.write ("5")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == CRUISER:
                        if cruiserP    == 0:
                                sys.stdout.write ("4")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == DESTROYER:
                        if destroyerP  == 0:
                                sys.stdout.write ("3")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == SUBMARINE:
                        if submarineP  == 0:
                                sys.stdout.write ("3")
                        else :
                                sys.stdout.write ("1")
                elif table [y][x] == TRANSPORTS:
                        if transportsP == 0:
                                sys.stdout.write ("2")
                        else :
                                sys.stdout.write ("1")
        elif submarine [y][x] == MISS:
                sys.stdout.write ("x")
        else :
```

```
                sys.stdout.write (" ")
"""
def leftoverEnemy ():
        if (warshipP   <= 0 and cruiserP   <= 0 and
            destroyerP <= 0 and submarineP <= 0 and transportsP <= 0):
                return True

def hitJudgement (table, submarine):
        noneAttack = submarine [y][x]
        damage     = table     [y][x]

        global warshipP, cruiserP, destroyerP, submarineP, transportsP

        if noneAttack != N:
                return MISS
        else:
                if   damage == WARSHIP:
                        submarine [y][x] = HIT
                        warshipP        -= 1
                        return WARSHIP
                elif damage == CRUISER:
                        submarine [y][x] = HIT
                        cruiserP        -= 1
                        return CRUISER
                elif damage == DESTROYER:
                        submarine [y][x] = HIT
                        destroyerP      -= 1
                        return DESTROYER
                elif damage == SUBMARINE:
                        submarine [y][x] = HIT
                        submarineP      -= 1
                        return SUBMARINE
                elif damage == TRANSPORTS:
                        submarine [y][x] = HIT
                        transportsP      -= 1
                        return TRANSPORTS
        submarine [y][x] = MISS
        return MISS

def hitDisplay (result):
        if   result == MISS:
                return MISS
        elif WARSHIP <= result <= TRANSPORTS:
```

```
                if   result == WARSHIP    and warshipP    <= 0:
                        return WARSHIP
                elif result == CRUISER    and cruiserP    <= 0:
                        return CRUISER
                elif result == DESTROYER  and destroyerP  <= 0:
                        return DESTROYER
                elif result == SUBMARINE  and submarineP  <= 0:
                        return SUBMARINE
                elif result == TRANSPORTS and transportsP <= 0:
                        return TRANSPORTS
                return HIT


for nplay in range(0, 100):
        # 艦船 HP
        warshipP    = 5
        cruiserP    = 4
        destroyerP  = 3
        submarineP  = 3
        transportsP = 2
        BATTLESHIP  = [TRANSPORTS, SUBMARINE, DESTROYER, CRUISER, WARSHIP]
        INITIAL_SHOOT     = 0
        SECOND_SHOOT      = 1
        FINISH_SHOOT      = 100
        # 座標軸
        y = 0
        x = 0
        submarine = initialization ()
        table     = initialization ()
        submarineFlg = 0
        y, x = shoot (INITIAL_SHOOT, submarine, MISS, y, x)
        result_tmp = hitJudgement (table, submarine)
        result = hitDisplay(result_tmp)

        for i in range(SECOND_SHOOT, FINISH_SHOOT):
                y, x = shoot (i, submarine, result, y, x) # test:
                result_tmp = hitJudgement (table, submarine)
                result = hitDisplay(result_tmp)
                # ゲーム終了
                if leftoverEnemy ():
                        break
        sum = sum + i+1
        print "%3d 回目のプレイ: 攻撃回数 %3d 回" % (nplay+1, i)
```

```
sys.stdout.write ("平均攻撃回数 = %3.2d\n" % (sum / 100))
```

A.4 攻略関数テンプレート

A.4.1 senkan.py

```
def shoot (attackTimes, submarine, attackResult, y, x):
    return y, x
```

A.5 学生配布用サンプル攻撃関数

A.5.1 sample1.py

```
def shoot (attackTimes, submarine, attackResult):
    import random
    global y,x
    while 1:
        y = random.randint(0, 9)
        x = random.randint(0, 9)
        if submarine[y][x] == 0:
            return y, x
```

A.5.2 sample2.py

```
def shoot (attackTimes, submarine, attackResult):
    global y, x, point_flg
    if   attackTimes > 0:
        x += 4
    else :
        point_flg = 0
        y = 0
        x = 0

    if x >= 10:
        y += 1
        x  = point_flg

    if y >= 10:
        if   (point_flg == 0):
            point_flg += 1
            y = 0
            x = point_flg
        elif (point_flg == 1):
            point_flg += 1
            y = 0
            x = point_flg
        elif (point_flg == 2):
            point_flg += 1
            y = 0
            x = point_flg
        elif (point_flg == 3):
            point_flg += 1
            y = 0
```

```
        x = point_flg
return y, x
```

A.5.3 sample3.py

```
def shoot (attackTimes, submarine, attackResult):
        global y, x
        y = attackTimes / 10
        x = attackTimes % 10
        return y, x
```

A.6 学生課題用攻撃関数

A.6.1 issue.py

```
def shoot (attackTimes, submarine, attackResult):
        global y, x
        if   attackTimes == 50:
                y = 0
                x = 1
        elif 0 < attackTimes and  attackTimes < 50:
                if   y % 2 == 0:
                        if   x+2 >= 10:
                                y += 1
                                x  = 1
                        else :
                                x += 2
                else:
                        if   x+2 >= 10:
                                y += 1
                                x  = 0
                        else :
                                x += 2
        elif 50 <= attackTimes :
                if   y % 2 == 0:
                        if   x+2 >= 10:
                                y += 1
                                x  = 0
                        else :
                                x += 2
                else:
                        if x+2 >= 10:
                                y += 1
                                x  = 1
                        else :
                                x += 2
        else :
                y = 0
                x = 0
        return y, x
```

付録 B　三回生向け前段階調査マインスイーパ

B.1　ヘッダーファイル

B.1.1　mine.h

```
#define TATE 10  /* 盤面の大きさ TATE × YOKO */
#define YOKO 10
#define NUMB 5  /* 爆弾（地雷）数  NUMB */

#define NOP 11    /* 開いていないパネルの値 */
#define NOBOMB 0
#define BOMB 99

#define TRUE 1
#define FALSE 0
#define OUT 99
```

B.2 Verification

B.2.1 minemain1.c

```
#include <stdio.h>
#include <time.h>
#include "mine.h"

void panelopen(int panel[TATE][YOKO], int *x, int *y, int init);
void createpanel(int orgpanel[TATE][YOKO]);
void initpanel(int panel[TATE][YOKO]);
int checkbomb(int orgpanel[TATE][YOKO], int x, int y);
int checkpanel(int panel[TATE][YOKO], int x, int y);
void setpanel(int orgpanel[TATE][YOKO], int panel[TATE][YOKO], int x, int y);
void printpanel(int panel[TATE][YOKO], int orgpanel[TATE][YOKO]);

int main(void)
{
  int orgpanel[TATE][YOKO], panel[TATE][YOKO];
  int i, x=0, y=0, max, check;

  max = TATE * YOKO - NUMB; /* 爆弾のないパネルの数 */

  srand((int)time(NULL));

  createpanel(orgpanel); /* 縦 (TATE)，横 (YOKO)，爆弾数 (NUMB) の盤面を作成 */
  initpanel(panel);
  for (i = 1; i <= max; i++) {
      panelopen(panel, &x, &y, i); /* ユーザ定義関数 panelopen の呼び出し */
      check = checkbomb(orgpanel,x,y);
      if(check == 1) {        /* 爆弾だったらプレイ終了 */
                printf("%3d 回目:(%2d,%2d)---BOMB!\n", i, x, y);
        break;
      } else if(check == 99 || checkpanel(panel,x,y) == 99) {
        i--;                /* 盤面外か既に開いているパネルならやり直し */
        continue;
      } else {                /* それ以外なら盤面情報を更新 */
                  printf("%3d 回目:(%2d,%2d)\n", i, x, y);
        setpanel(orgpanel,panel,x,y);
      }
    }
  putchar('\n');
  printpanel(panel, orgpanel);
```

```
  putchar('\n');
}

void createpanel(int panel[][10])
{
  int i, j;
  int t, y;

  for(i = 0; i < 10; i++) {
    for(j = 0; j < 10; j++) {
      panel[i][j] = NOBOMB;
    }
  }

  i = 1;
  while(i <= 5) {
    t = rand() % 10;
    y = rand() % 10;
    if(panel[t][y] == 99) {
      continue;
    } else {
      panel[t][y] = BOMB;
      i++;
    }
  }
}

void initpanel(int panel[TATE][YOKO])
{
  int i, j;

  for(i = 0; i < TATE; i++) {
    for(j = 0; j < YOKO; j++) {
      panel[i][j] = NOP;
    }
  }
}

int checkbomb(int panel[][10], int x, int y)
{
  if (x < 0 || x >= 10) return OUT;
  if (y < 0 || y >= 10) return OUT;
```

```
  if (panel[x][y] == 99) {
    return TRUE;
  }
  return FALSE;
}

int checkpanel(int panel[TATE][YOKO], int x, int y)
{
        int n;

        n = panel[x][y];
        if (n < 9) {
                return OUT;
        }
        else {
                return TRUE;
        }
}

void setpanel(int orgpanel[TATE][YOKO], int panel[TATE][YOKO], int x, int y)
{
  int num = 0;

  if(x > 0) {
    if(orgpanel[x-1][y] == 99) num++;
    if(y > 0) {if(orgpanel[x-1][y-1] == 99) num++;}
    if(y < 9) {if(orgpanel[x-1][y+1] == 99) num++;}
  }
  if(y > 0) {if(orgpanel[x][y-1] == 99) num++;}
  if(y < 9) {if(orgpanel[x][y+1] == 99) num++;}
  if(x < 9) {
    if(orgpanel[x+1][y] == 99) num++;
    if(y > 0) {if(orgpanel[x+1][y-1] == 99) num++;}
    if(y < 9) {if(orgpanel[x+1][y+1] == 99) num++;}
  }

  panel[x][y] = num;
}

void printpanel(int panel[TATE][YOKO], int orgpanel[TATE][YOKO])
{
  int i,j;
  for(i = 0; i < TATE; i++) {
```

```
    for(j = 0; j < YOKO; j++) {
      if(orgpanel[i][j] == BOMB) {
        printf("B ");
      } else if(panel[i][j] == NOP) {
        printf("- ");
      } else {
        printf("%d ", panel[i][j]);
      }
    }
    putchar('\n');
  }
}
```

B.3 Score

B.3.1 minemain100.c

```
#include <stdio.h>
#include <time.h>
#include "mine.h"

#define KURIKAESI 10000

void panelopen(int panel[TATE][YOKO], int *x, int *y, int init);
void createpanel(int orgpanel[TATE][YOKO]);
void initpanel(int panel[TATE][YOKO]);
int checkbomb(int orgpanel[TATE][YOKO], int x, int y);
int checkpanel(int panel[TATE][YOKO], int x, int y);
void setpanel(int orgpanel[TATE][YOKO], int panel[TATE][YOKO], int x, int y);
void printpanel(int panel[TATE][YOKO], int orgpanel[TATE][YOKO]);

int main(void)
{
  int orgpanel[TATE][YOKO], panel[TATE][YOKO];
  int i, x=0, y=0, max, checkb, checkp;
  int kaisu, seikou, avek;

  max = TATE * YOKO - NUMB; /* 爆弾のないパネルの数 */
  seikou = 0;
  avek = 0;
  srand((int)time(NULL));

  for (kaisu = 1; kaisu <= KURIKAESI; kaisu++) {
    createpanel(orgpanel); /* 縦(TATE)，横(YOKO)，爆弾数(NUMB)の盤面を作成 */
    initpanel(panel);
    for (i = 1; i <= max; i++) {
      panelopen(panel, &x, &y, i); /* ユーザ定義関数 panelopen の呼び出し */
      checkb = checkbomb(orgpanel,x,y);
      checkp = checkpanel(panel,x,y);
      if(checkb == TRUE) {        /* 爆弾だったらプレイ終了 */
                /*
                printf("%3d 回目:(%2d,%2d)---BOMB!\n", i, x, y);
                */
        break;
      } else if(checkb == OUT || checkp == OUT) {
        i--;                /* 盤面外か既に開いているパネルならやり直し */
```

```
        continue;
      } else {                  /* それ以外なら盤面情報を更新 */
                /*
                printf("%3d回目:(%2d,%2d)\n", i, x, y);
                */
        setpanel(orgpanel,panel,x,y);
      }
    }

    printf("%3d回目：%2d\n", kaisu, i-1);
        /*
        putchar('\n');
    printpanel(panel, orgpanel);
    putchar('\n');
    */
    if ( i > max) {
          seikou++;
        }
  avek = avek + i - 1;
  }
  printf("成功回数：%d\n", seikou);
  printf("平均開示数：%6.2f\n", (float)avek / KURIKAESI);
}

void createpanel(int panel[TATE][YOKO])
{
  int i, j;
  int t, y;

  for(i = 0; i < TATE; i++) {
    for(j = 0; j < YOKO; j++) {
      panel[i][j] = NOBOMB;
    }
  }

  i = 1;
  while(i <= NUMB) {
    t = rand() % TATE;
    y = rand() % YOKO;
    if(panel[t][y] == BOMB) {
      continue;
    } else {
      panel[t][y] = BOMB;
```

```
      i++;
    }
  }
}

void initpanel(int panel[TATE][YOKO])
{
  int i, j;

  for(i = 0; i < TATE; i++) {
    for(j = 0; j < YOKO; j++) {
      panel[i][j] = NOP;
    }
  }
}

int checkbomb(int panel[TATE][YOKO], int x, int y)
{
  if (x < 0 || x >= TATE) return OUT;
  if (y < 0 || y >= YOKO) return OUT;

  if (panel[x][y] == BOMB) {
    return TRUE;
  }
  return FALSE;
}

int checkpanel(int panel[TATE][YOKO], int x, int y)
{
        int n;

        n = panel[x][y];
        if (n < 9) {
                return OUT;
        }
        else {
                return TRUE;
        }
}

void setpanel(int orgpanel[TATE][YOKO], int panel[TATE][YOKO], int x, int y)
{
  int num = 0;
```

```
  if(x > 0) {
    if(orgpanel[x-1][y] == BOMB) num++;
    if(y > 0) {if(orgpanel[x-1][y-1] == BOMB) num++;}
    if(y < YOKO-1) {if(orgpanel[x-1][y+1] == BOMB) num++;}
  }
  if(y > 0) {if(orgpanel[x][y-1] == BOMB) num++;}
  if(y < YOKO-1) {if(orgpanel[x][y+1] == BOMB) num++;}
  if(x < TATE-1) {
    if(orgpanel[x+1][y] == BOMB) num++;
    if(y > 0) {if(orgpanel[x+1][y-1] == BOMB) num++;}
    if(y < YOKO-1) {if(orgpanel[x+1][y+1] == BOMB) num++;}
  }

  panel[x][y] = num;
}

void printpanel(int panel[TATE][YOKO], int orgpanel[TATE][YOKO])
{
  int i,j;
  for(i = 0; i < TATE; i++) {
    for(j = 0; j < YOKO; j++) {
      if(orgpanel[i][j] == BOMB) {
        printf("B ");
      } else if(panel[i][j] == NOP) {
        printf("- ");
      } else {
        printf("%d ", panel[i][j]);
      }
    }
    putchar('\n');
  }
}
```

B.4 攻略用関数テンプレート

B.4.1 panelopen.c

付録 C　一回生向け演習 Battle Ship ゲーム

C.1　ヘッダーファイル

C.1.1　senkan.h

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 0   // 未攻撃
#define BM 1  // 命中
#define BH 2  // ハズレ
#define SK 11 // length:5 戦艦沈没
#define JK 12 // length:4 巡洋艦沈没
#define KK 13 // length:3 駆逐艦沈没
#define SM 14 // length:3 潜水艦沈没
#define YS 15 // length:2 輸送艦沈没

#define F010 if(i==1){fun_c++; up = fun_c;} if(up==at_flg){upAt();}
#define F011 if(i==1){fun_c++; rig = fun_c;} if(rig==at_flg){rightAt();}
#define F012 if(i==1){fun_c++; dow = fun_c;} if(dow==at_flg){downAt();}
#define F013 if(i==1){fun_c++; lef = fun_c;} if(lef==at_flg){leftAt();}

#define F000 if(SK<=kekka&&kekka<=YS) {         \
        at_flg=0;                               \
    }                                           \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1) {        \
        if(up==at_flg&&gx-1<0) {                \
            at_flg++;                           \
        } else if(rig==at_flg&&10<=gy+1) {      \
            at_flg++;                           \
        } else if(dow==at_flg&&10<gx+1) {       \
            at_flg++;                           \
        } else if(lef==at_flg&&gy-1<0) {        \
            at_flg++;                           \
        }                                       \
    }                                           \
    if(kekka==BM && 0<fun_c) {                  \
        if(0<fun_c&&at_flg==0) {                \
            at_flg++;bx=gx;by=gy;               \
        } else if(at_flg==fun_c+1) {            \
            at_flg=0;                           \
```

```
        }                                       \
    } else {                                    \
        random100();                            \
    }

#define F001 if(SK<=kekka&&kekka<=YS) {         \
        at_flg=0;                               \
    }                                           \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1) {        \
        if(up==at_flg&&gx-1<0) {                \
            at_flg++;                           \
        } else if(rig==at_flg&&10<=gy+1) {      \
            at_flg++;                           \
        } else if(dow==at_flg&&10<gx+1) {       \
            at_flg++;                           \
        } else if(lef==at_flg&&gy-1<0) {        \
            at_flg++;                           \
        }                                       \
    }                                           \
    if(kekka==BM && 0<fun_c) {                  \
        if(0<fun_c&&at_flg==0) {                \
            at_flg++;bx=gx;by=gy;               \
        } else if(at_flg==fun_c+1) {            \
            at_flg=0;                           \
        }                                       \
    } else {                                    \
        oneSkip();                              \
    }                                           \

#define F002 if(SK<=kekka&&kekka<=YS) {         \
        at_flg=0;                               \
    }                                           \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1) {        \
        if(up==at_flg&&gx-1<0) {                \
            at_flg++;                           \
        } else if(rig==at_flg&&10<=gy+1) {      \
            at_flg++;                           \
        } else if(dow==at_flg&&10<gx+1) {       \
            at_flg++;                           \
        } else if(lef==at_flg&&gy-1<0) {        \
            at_flg++;                           \
        }                                       \
    }                                           \
```

```
    if(kekka==BM && 0<fun_c) {                  \
    if(0<fun_c&&at_flg==0) {                    \
        at_flg++;bx=gx;by=gy;                   \
    } else if(at_flg==fun_c+1) {                \
        at_flg=0;                               \
    } else {                                    \
        twoSkip();                              \
    }                                           \

#define F003 if(SK<=kekka&&kekka<=YS) {         \
        at_flg=0;                               \
    }                                           \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1) {        \
        if(up==at_flg&&gx-1<0){                 \
            at_flg++;                           \
        } else if(rig==at_flg&&10<=gy+1) {      \
            at_flg++;                           \
        } else if(dow==at_flg&&10<gx+1) {       \
            at_flg++;                           \
        } else if(lef==at_flg&&gy-1<0) {        \
            at_flg++;                           \
        }                                       \
    }                                           \
    if(kekka==BM && 0<fun_c) {                  \
        if(0<fun_c&&at_flg==0) {                \
            at_flg++;bx=gx;by=gy;               \
        } else if(at_flg==fun_c+1) {            \
            at_flg=0;                           \
        }                                       \
    } else {                                    \
        twoRight();                             \
    }                                           \

#define F004 if(SK<=kekka&&kekka<=YS){          \
        at_flg=0;                               \
    }                                           \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1){         \
        if(up==at_flg&&gx-1<0){                 \
            at_flg++;                           \
        }                                       \
        else if(rig==at_flg&&10<=gy+1){         \
            at_flg++;                           \
        }                                       \
```

```
        else if(dow==at_flg&&10<gx+1){          \
            at_flg++;                           \
        }                                       \
        else if(lef==at_flg&&gy-1<0){           \
            at_flg++;                           \
        }                                       \
    }                                           \
    if(kekka==BM && 0<fun_c){                   \
        if(0<fun_c&&at_flg==0){                 \
            at_flg++;bx=gx;by=gy;               \
        }                                       \
        else if(at_flg==fun_c+1){               \
            at_flg=0;                           \
        }                                       \
    }                                           \
    else{                                       \
        threeSkip();                            \
    }                                           \

#define F005 if(SK<=kekka&&kekka<=YS){          \
        at_flg=0;                               \
    }                                           \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1){         \
        if(up==at_flg&&gx-1<0){                 \
            at_flg++;                           \
        }                                       \
        else if(rig==at_flg&&10<=gy+1){         \
            at_flg++;                           \
        }                                       \
        else if(dow==at_flg&&10<gx+1){          \
            at_flg++;                           \
        }                                       \
        else if(lef==at_flg&&gy-1<0){           \
            at_flg++;                           \
        }                                       \
    }                                           \
    if(kekka==BM && 0<fun_c){                   \
        if(0<fun_c&&at_flg==0){                 \
            at_flg++;bx=gx;by=gy;               \
        }                                       \
        else if(at_flg==fun_c+1){               \
            at_flg=0;                           \
        }                                       \
```

```
    }                                       \
    else{                                   \
        threeRight();                       \
    }                                       \

#define F006 if(SK<=kekka&&kekka<=YS){      \
        at_flg=0;                           \
    }                                       \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1){     \
        if(up==at_flg&&gx-1<0){             \
            at_flg++;                       \
        }                                   \
        else if(rig==at_flg&&10<=gy+1){     \
            at_flg++;                       \
        }                                   \
        else if(dow==at_flg&&10<gx+1){      \
            at_flg++;                       \
        }                                   \
        else if(lef==at_flg&&gy-1<0){       \
            at_flg++;                       \
        }                                   \
    }                                       \
    if(kekka==BM && 0<fun_c){               \
        if(0<fun_c&&at_flg==0){             \
            at_flg++;bx=gx;by=gy;           \
        }                                   \
        else if(at_flg==fun_c+1){           \
            at_flg=0;                       \
        }                                   \
    }                                       \
    else{                                   \
        fourSkip();                         \
    }                                       \

#define F007 if(SK<=kekka&&kekka<=YS){      \
        at_flg=0;                           \
    }                                       \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1){     \
        if(up==at_flg&&gx-1<0){             \
            at_flg++;                       \
        }                                   \
        else if(rig==at_flg&&10<=gy+1){     \
            at_flg++;                       \
```

```
        }                                       \
        else if(dow==at_flg&&10<gx+1){          \
            at_flg++;                           \
        }                                       \
        else if(lef==at_flg&&gy-1<0){           \
            at_flg++;                           \
        }                                       \
    }                                           \
    if(kekka==BM && 0<fun_c){                   \
        if(0<fun_c&&at_flg==0){                 \
            at_flg++;bx=gx;by=gy;               \
        }                                       \
        else if(at_flg==fun_c+1){               \
            at_flg=0;                           \
        }                                       \
    }                                           \
    else{                                       \
        fourRight();                            \
    }                                           \

#define F008 if(SK<=kekka&&kekka<=YS){          \
        at_flg=0;                               \
    }                                           \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1){         \
        if(up==at_flg&&gx-1<0){                 \
            at_flg++;                           \
        }                                       \
        else if(rig==at_flg&&10<=gy+1){         \
            at_flg++;                           \
        }                                       \
        else if(dow==at_flg&&10<gx+1){          \
            at_flg++;                           \
        }                                       \
        else if(lef==at_flg&&gy-1<0){           \
            at_flg++;                           \
        }                                       \
    }                                           \
    if(kekka==BM && 0<fun_c){                   \
        if(0<fun_c&&at_flg==0){                 \
            at_flg++;bx=gx;by=gy;               \
        }                                       \
        else if(at_flg==fun_c+1){               \
            at_flg=0;                           \
```

```
        }                                       \
    }                                           \
    else{                                       \
        oddSkip();                              \
    }                                           \

#define F009 if(SK<=kekka&&kekka<=YS){          \
        at_flg=0;                               \
    }                                           \
    if(i!=1&&0<at_flg&&at_flg<fun_c+1){         \
        if(up==at_flg&&gx-1<0){                 \
            at_flg++;                           \
        }                                       \
        else if(rig==at_flg&&10<=gy+1){         \
            at_flg++;                           \
        }                                       \
        else if(dow==at_flg&&10<gx+1){          \
            at_flg++;                           \
        }                                       \
        else if(lef==at_flg&&gy-1<0){           \
            at_flg++;                           \
        }                                       \
    }                                           \
    if(kekka==BM && 0<fun_c){                   \
        if(0<fun_c&&at_flg==0){                 \
            at_flg++;bx=gx;by=gy;               \
        }                                       \
        else if(at_flg==fun_c+1){               \
            at_flg=0;                           \
        }                                       \
    }                                           \
    else{                                       \
        evenSkip();                             \
    }                                           \

extern int tsb[10][10], sb[10][10];
extern int skp, jkp, kkp, smp, ysp;
extern int gx, gy;
extern int i, kekka, k;
extern int bx, by;
extern int fun_c, at_flg;
extern int up, rig, dow, lef;
```

C.2 攻撃抽象化用関数ファイル

C.2.1 function.c

```
#include <stdio.h>
#include <stdlib.h>
#include "senkan.h"

extern int tsb[10][10], sb[10][10];
extern int skp, jkp, kkp, smp, ysp;
extern int gx, gy;
extern int i, kekka;
extern int bx, by, at_flg;

void random100();
void oneSkip();
void twoSkip();
void twoRight();
void threeSkip();
void threeRight();
void fourSkip();
void fourRight();
void oddSkip();
void evenSkip();
void upAt();
void rightAt();
void downAt();
void leftAt();

void random100()
{
  do
    {
      gx = rand() % 10;
      gy = rand() % 10;
    } while(sb[gx][gy] != N);

  return;
}

void oneSkip()
{
  int tate, yoko;
```

```
  static int finish[10][10] =
    { {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0}
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] == 1 && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void twoSkip()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
    };
```

```
  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void twoRight()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {0, 1, 0, 0, 1, 0, 0, 1, 0, 0},
      {0, 0, 1, 0, 0, 1, 0, 0, 1, 0},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {0, 1, 0, 0, 1, 0, 0, 1, 0, 0},
      {0, 0, 1, 0, 0, 1, 0, 0, 1, 0},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {0, 1, 0, 0, 1, 0, 0, 1, 0, 0},
      {0, 0, 1, 0, 0, 1, 0, 0, 1, 0},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
    };
  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
```

```
  return;
}

void threeSkip()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void threeRight()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {0, 1, 0, 0, 0, 1, 0, 0, 0, 1},
      {0, 0, 1, 0, 0, 0, 1, 0, 0, 0},
      {0, 0, 0, 1, 0, 0, 0, 1, 0, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
```

```
      {0, 1, 0, 0, 0, 1, 0, 0, 0, 1},
      {0, 0, 1, 0, 0, 0, 1, 0, 0, 0},
      {0, 0, 0, 1, 0, 0, 0, 1, 0, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {0, 1, 0, 0, 0, 1, 0, 0, 0, 1},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void fourSkip()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
```

```
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void fourRight()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {0, 1, 0, 0, 0, 0, 1, 0, 0, 0},
      {0, 0, 1, 0, 0, 0, 0, 1, 0, 0},
      {0, 0, 0, 1, 0, 0, 0, 0, 1, 0},
      {0, 0, 0, 0, 1, 0, 0, 0, 0, 1},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {0, 1, 0, 0, 0, 0, 1, 0, 0, 0},
      {0, 0, 1, 0, 0, 0, 0, 1, 0, 0},
      {0, 0, 0, 1, 0, 0, 0, 0, 1, 0},
      {0, 0, 0, 0, 1, 0, 0, 0, 0, 1},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void oddSkip()
{
```

```
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void evenSkip()
{
  int tate, yoko;
  static int finish[10][10] =
    { {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
```

```
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void upAt()
{
  int at_count;

  for(at_count = 1; at_count < 5; at_count++)
    {
      if (0 <= bx - at_count && sb[bx - at_count][by] == N)
        {
          gx = bx - at_count;
          gy = by;
          if (bx - at_count+1 < 0)
            {
              at_flg++;
            }
          return;
        }
    }
  at_flg++;

  return;
}

void rightAt()
{
  int at_count;
  for(at_count = 1; at_count < 5; at_count++)
```

```
    {
      if (by + at_count < 10 && sb[bx][by + at_count] == N)
        {
          gx = bx;
          gy = by + at_count;
          if (10 <= by + at_count+1)
            {
              at_flg++;
            }
          return;
        }
    }
  at_flg++;

  return;
}

void downAt()
{
  int at_count;

  for(at_count = 1; at_count < 5; at_count++)
    {
      if (bx + at_count < 10 && sb[bx + at_count][by] == N)
        {
          gx = bx + at_count;
          gy = by;
          if (10 <= bx + at_count+1)
            {
              at_flg++;
            }
          return;
        }
    }
  at_flg++;

  return;
}

void leftAt()
{
  int at_count;
```

```
  for(at_count = 1; at_count < 5; at_count++)
    {
      if (0 <= by - at_count && sb[bx][by - at_count] == N)
        {
          gx = bx;
          gy = by - at_count;
          if (by - at_count+1 < 0)
            {
              at_flg++;
            }
          return;
        }
    }
  at_flg++;

  return;
}
```

C.3 Practice

C.3.1 senkanplay.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char line[100];

#define N 0
#define BM 1
#define BH 2
#define SK 11
#define JK 12
#define KK 13
#define SM 14
#define YS 15

void haichi (int sb[][10]);
void dispsb (int sb[][10]);
int zanteki (int skp, int jkp, int kkp, int smp, int ysp);
void shoot (int sb[][10], int i, int *x, int *y);
int meichup (int sb[][10], int x, int y);
void meichud (int k);

int skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;

int
main (void)
{

  int sb[10][10];
  int i, x, y, k;

  haichi (sb);

  dispsb (sb);

  zanteki (skp, jkp, kkp, smp, ysp);

  shoot (sb, 1, &x, &y);
```

```
  printf ("(x,y) = (%d,%d)\n", x, y);

  for (i = 2; i <= 100; i++)
    {
      k = meichup (sb, x, y);
      dispsb (sb);
      meichud (k);
      if (zanteki (skp, jkp, kkp, smp, ysp))
        {
          break;
        }
      shoot (sb, i, &x, &y);
    }
  printf ("攻撃回数＝%d\n", --i);

  printf ("\nPress any key to continue...\n");
  getchar ();
}

void
haichi (int sb[][10])
{
  int i, j, x, y;

  srand ((unsigned int) time (NULL));

  for (i = 0; i < 10; i++)
    {
      for (j = 0; j < 10; j++)
        {
          sb[i][j] = N;
        }
    }

  if (rand () % 2 == 0)
    {                                   /* 戦艦縦方向 */
      y = rand () % 10;
      x = rand () % 6;
      sb[x][y] = SK;
      sb[x + 1][y] = SK;
      sb[x + 2][y] = SK;
      sb[x + 3][y] = SK;
      sb[x + 4][y] = SK;
```

```
  }
else
  {                                /* 戦艦横方向 */
    x = rand () % 10;
    y = rand () % 6;
    sb[x][y] = SK;
    sb[x][y + 1] = SK;
    sb[x][y + 2] = SK;
    sb[x][y + 3] = SK;
    sb[x][y + 4] = SK;
  }

if (rand () % 2 == 0)
  {                                /* 巡洋艦縦方向 */
    y = rand () % 10;
    x = rand () % 7;
    while (sb[x][y] != N || sb[x + 1][y] != N || sb[x + 2][y] != N
           || sb[x + 3][y] != N)
      {
        y = rand () % 10;
        x = rand () % 7;
      }
    sb[x][y] = JK;
    sb[x + 1][y] = JK;
    sb[x + 2][y] = JK;
    sb[x + 3][y] = JK;
  }
else
  {                                /* 巡洋艦横方向 */
    x = rand () % 10;
    y = rand () % 7;
    while (sb[x][y] != N || sb[x][y + 1] != N || sb[x][y + 2] != N
           || sb[x][y + 3] != N)
      {
        x = rand () % 10;
        y = rand () % 7;
      }
    sb[x][y] = JK;
    sb[x][y + 1] = JK;
    sb[x][y + 2] = JK;
    sb[x][y + 3] = JK;
  }
```

```
if (rand () % 2 == 0)
  {                                  /* 駆逐艦縦方向 */
    y = rand () % 10;
    x = rand () % 8;
    while (sb[x][y] != N || sb[x + 1][y] != N || sb[x + 2][y] != N)
      {
        y = rand () % 10;
        x = rand () % 8;
      }
    sb[x][y] = KK;
    sb[x + 1][y] = KK;
    sb[x + 2][y] = KK;
  }
else
  {                                  /* 駆逐艦横方向 */
    x = rand () % 10;
    y = rand () % 8;
    while (sb[x][y] != N || sb[x][y + 1] != N || sb[x][y + 2] != N)
      {
        x = rand () % 10;
        y = rand () % 8;
      }
    sb[x][y] = KK;
    sb[x][y + 1] = KK;
    sb[x][y + 2] = KK;
  }

if (rand () % 2 == 0)
  {                                  /* 潜水艦縦方向 */
    y = rand () % 10;
    x = rand () % 8;
    while (sb[x][y] != N || sb[x + 1][y] != N || sb[x + 2][y] != N)
      {
        y = rand () % 10;
        x = rand () % 8;
      }
    sb[x][y] = SM;
    sb[x + 1][y] = SM;
    sb[x + 2][y] = SM;
  }
else
  {                                  /* 潜水艦横方向 */
    x = rand () % 10;
```

```
      y = rand () % 8;
      while (sb[x][y] != N || sb[x][y + 1] != N || sb[x][y + 2] != N)
        {
          x = rand () % 10;
          y = rand () % 8;
        }
      sb[x][y] = SM;
      sb[x][y + 1] = SM;
      sb[x][y + 2] = SM;
    }

  if (rand () % 2 == 0)
    {                                   /* 輸送船縦方向 */
      y = rand () % 10;
      x = rand () % 9;
      while (sb[x][y] != N || sb[x + 1][y] != N)
        {
          y = rand () % 10;
          x = rand () % 9;
        }
      sb[x][y] = YS;
      sb[x + 1][y] = YS;
    }
  else
    {                                   /* 輸送船横方向 */
      x = rand () % 10;
      y = rand () % 9;
      while (sb[x][y] != N || sb[x][y + 1] != N)
        {
          x = rand () % 10;
          y = rand () % 9;
        }
      sb[x][y] = YS;
      sb[x][y + 1] = YS;
    }
}

void
dispsb (int sb[][10])
{
  int i, j;

  printf ("   1 2 3 4 5 6 7 8 9 10\n");
```

```
  printf ("  ----------------------\n");

  for (i = 0; i < 10; i++)
    {
      printf ("%2d", i + 1);
      for (j = 0; j < 10; j++)
        {
          putchar ('|');
          switch (sb[i][j])
            {
            case BM:
              putchar ('O');
              break;
            case BH:
              putchar ('X');
              break;
            default:
              putchar (' ');
              break;
            }
        }
      printf ("|\n");
    }

  printf ("  ----------------------\n");
}

int
zanteki (int skp, int jkp, int kkp, int smp, int ysp)
{
  if (skp == 0 && jkp == 0 && kkp == 0 && smp == 0 && ysp == 0)
    {
      printf ("全艦撃沈！\n");
      return 1;
    }

  printf ("残敵＝");
  if (skp > 0)
    {
      printf ("戦艦 ");
    }
  if (jkp > 0)
    {
```

```
      printf ("巡洋艦 ");
    }
  if (kkp > 0)
    {
      printf ("駆逐艦 ");
    }
  if (smp > 0)
    {
      printf ("潜水艦 ");
    }
  if (ysp > 0)
    {
      printf ("輸送船");
    }
  putchar ('\n');
  return 0;
}

void
shoot (int sb[][10], int i, int *x, int *y)
{
  int tate, yoko;

  while (1)
    {
      printf ("次弾(%d)発射位置？ 縦,横 ＝ ", i);
      fgets (line, sizeof (line), stdin);
      sscanf (line, "%d,%d", &tate, &yoko);
      if (tate >= 1 && tate <= 10 && yoko >= 1 && yoko <= 10)
        {
          *x = tate - 1;
          *y = yoko - 1;
          return;
        }
    }
}

int
meichup (int sb[][10], int x, int y)
{
  int a;

  a = sb[x][y];
```

```
  if (a == SK)
    {
      sb[x][y] = BM;
      skp--;
      return SK;
    }
  if (a == JK)
    {
      sb[x][y] = BM;
      jkp--;
      return JK;
    }
  if (a == KK)
    {
      sb[x][y] = BM;
      kkp--;
      return KK;
    }
  if (a == SM)
    {
      sb[x][y] = BM;
      smp--;
      return SM;
    }
  if (a == YS)
    {
      sb[x][y] = BM;
      ysp--;
      return YS;
    }
  if (a == BM)
    {
      return BH;
    }
  sb[x][y] = BH;
  return BH;
}

void
meichud (int k)
{
  if (k == BH)
```

```
  {
    printf ("ハズレ！\n");
  }
else if (k == SK)
  {
    printf ("命中！");
    if (skp <= 0)
      {
        printf ("戦艦沈没！");
      }
    putchar ('\n');
  }
else if (k == JK)
  {
    printf ("命中！");
    if (jkp <= 0)
      {
        printf ("巡洋艦沈没！");
      }
    putchar ('\n');
  }
else if (k == KK)
  {
    printf ("命中！");
    if (kkp <= 0)
      {
        printf ("駆逐艦沈没！");
      }
    putchar ('\n');
  }
else if (k == SM)
  {
    printf ("命中！");
    if (smp <= 0)
      {
        printf ("潜水艦沈没！");
      }
    putchar ('\n');
  }
else if (k == YS)
  {
    printf ("命中！");
    if (ysp <= 0)
```

```
        {
          printf ("輸送船沈没！");
        }
      putchar ('\n');
    }
}
```

C.4 Verification

C.4.1 senkan001.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "senkan.h"

char line[100];

void haichi ();
void dispsb ();
int zanteki ();
void shoot ();
int meichup ();
int meichud ();

int tsb[10][10], sb[10][10];
int skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;
int gx=0, gy=0;
int i, kekka, k;
int bx, by;
int fun_c=0, at_flg=0;
int up, rig, dow, lef;

int
main (void)
{
  int tate, yoko;

  haichi ();
  for (tate = 0; tate < 10; tate++)
    {
      for (yoko = 0; yoko < 10; yoko++)
        {
          sb[tate][yoko] = N;
        }
    }

  dispsb ();
  zanteki ();
```

```
  i = 1;
  shoot ();
  getchar ();

  for (i = 2; i <= 101; i++)
    {
      k = meichup ();
      printf("%d\n", k);
      dispsb ();
      kekka = meichud ();
      if (zanteki () || (i == 101))
        {
          break;
        }
      shoot ();
      // printf("%d %d", x+1, y+1);
      getchar ();
    }
  printf ("攻撃回数＝%d\n", --i);
  printf ("Press any key to continue...\n");
  getchar ();
}

void
haichi ()
{
  int tate, yoko, cx, cy;

  srand ((unsigned int) time (NULL));

  for (tate = 0; tate < 10; tate++)
    {
      for (yoko = 0; yoko < 10; yoko++)
        {
          tsb[tate][yoko] = N;
        }
    }

  if (rand () % 2 == 0)
    {                                   /* 戦艦縦方向 */
      cy = rand () % 10;
      cx = rand () % 6;
      tsb[cx][cy] = SK;
```

```
    tsb[cx + 1][cy] = SK;
    tsb[cx + 2][cy] = SK;
    tsb[cx + 3][cy] = SK;
    tsb[cx + 4][cy] = SK;
  }
else
  {                                /* 戦艦横方向 */
    cx = rand () % 10;
    cy = rand () % 6;
    tsb[cx][cy] = SK;
    tsb[cx][cy + 1] = SK;
    tsb[cx][cy + 2] = SK;
    tsb[cx][cy + 3] = SK;
    tsb[cx][cy + 4] = SK;
  }

if (rand () % 2 == 0)
  {                                /* 巡洋艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 7;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N
           || tsb[cx + 3][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 7;
      }
    tsb[cx][cy] = JK;
    tsb[cx + 1][cy] = JK;
    tsb[cx + 2][cy] = JK;
    tsb[cx + 3][cy] = JK;
  }
else
  {                                /* 巡洋艦横方向 */
    cx = rand () % 10;
    cy = rand () % 7;
    while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N
           || tsb[cx][cy + 3] != N)
      {
        cx = rand () % 10;
        cy = rand () % 7;
      }
    tsb[cx][cy] = JK;
    tsb[cx][cy + 1] = JK;
```

```
    tsb[cx][cy + 2] = JK;
    tsb[cx][cy + 3] = JK;
  }

if (rand () % 2 == 0)
  {                                 /* 駆逐艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 8;
      }
    tsb[cx][cy] = KK;
    tsb[cx + 1][cy] = KK;
    tsb[cx + 2][cy] = KK;
  }
else
  {                                 /* 駆逐艦横方向 */
    cx = rand () % 10;
    cy = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
      {
        cx = rand () % 10;
        cy = rand () % 8;
      }
    tsb[cx][cy] = KK;
    tsb[cx][cy + 1] = KK;
    tsb[cx][cy + 2] = KK;
  }

if (rand () % 2 == 0)
  {                                 /* 潜水艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 8;
      }
    tsb[cx][cy] = SM;
    tsb[cx + 1][cy] = SM;
    tsb[cx + 2][cy] = SM;
```

```
    }
  else
    {                                   /* 潜水艦横方向 */
      cx = rand () % 10;
      cy = rand () % 8;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
        {
          cx = rand () % 10;
          cy = rand () % 8;
        }
      tsb[cx][cy] = SM;
      tsb[cx][cy + 1] = SM;
      tsb[cx][cy + 2] = SM;
    }

  if (rand () % 2 == 0)
    {                                   /* 輸送船縦方向 */
      cy = rand () % 10;
      cx = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N)
        {
          cy = rand () % 10;
          cx = rand () % 9;
        }
      tsb[cx][cy] = YS;
      tsb[cx + 1][cy] = YS;
    }
  else
    {                                   /* 輸送船横方向 */
      cx = rand () % 10;
      cy = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N)
        {
          cx = rand () % 10;
          cy = rand () % 9;
        }
      tsb[cx][cy] = YS;
      tsb[cx][cy + 1] = YS;
    }
}

void
dispsb ()
```

```
{
  int tate, yoko;

  printf ("   1 2 3 4 5 6 7 8 9 10\n");
  printf ("  ---------------------\n");

  for (tate = 0; tate < 10; tate++)
    {
      printf ("%2d", tate + 1);
      for (yoko = 0; yoko < 10; yoko++)
        {
          putchar ('|');
          switch (sb[tate][yoko])
            {
            case BM:
              putchar ('O');
              break;
            case BH:
              putchar ('X');
              break;
            default:
              putchar (' ');
              break;
            }
        }
      printf ("|\n");
    }

  printf ("  ---------------------\n");
}

int
zanteki ()
{
  if (skp <= 0 && jkp <= 0 && kkp <= 0 && smp <= 0 && ysp <= 0)
    {
      printf ("全艦撃沈！\n");
      return 1;
    }

  printf ("残敵＝");
  if (skp > 0)
    {
```

```
      printf ("戦艦 ");
    }
  if (jkp > 0)
    {
      printf ("巡洋艦 ");
    }
  if (kkp > 0)
    {
      printf ("駆逐艦 ");
    }
  if (smp > 0)
    {
      printf ("潜水艦 ");
    }
  if (ysp > 0)
    {
      printf ("輸送船");
    }
  putchar ('\n');
  return 0;
}


int
meichup ()
{
  int a, b;

  a = tsb[gx][gy];
  b = sb[gx][gy];

  if (b != N)
    {
      return BH;
    }
  if (a == SK)
    {
      sb[gx][gy] = BM;
      skp--;
      return SK;
    }
  if (a == JK)
    {
```

```
      sb[gx][gy] = BM;
      jkp--;
      return JK;
    }
  if (a == KK)
    {
      sb[gx][gy] = BM;
      kkp--;
      return KK;
    }
  if (a == SM)
    {
      sb[gx][gy] = BM;
      smp--;
      return SM;
    }
  if (a == YS)
    {
      sb[gx][gy] = BM;
      ysp--;
      return YS;
    }
  sb[gx][gy] = BH;
  return BH;
}

int
meichud ()
{
  if (k == BH)
    {
      printf ("ハズレ！\n");
      return BH;
    }
  else if (k == SK)
    {
      printf ("命中！");
      if (skp <= 0)
        {
          printf ("戦艦沈没！\n");
          return SK;
        }
      putchar ('\n');
```

```
    }
  else if (k == JK)
    {
      printf ("命中！");
      if (jkp <= 0)
        {
          printf ("巡洋艦沈没！\n");
          return JK;
        }
      putchar ('\n');
    }
  else if (k == KK)
    {
      printf ("命中！");
      if (kkp <= 0)
        {
          printf ("駆逐艦沈没！\n");
          return KK;
        }
      putchar ('\n');
    }
  else if (k == SM)
    {
      printf ("命中！");
      if (smp <= 0)
        {
          printf ("潜水艦沈没！\n");
          return SM;
        }
      putchar ('\n');
    }
  else if (k == YS)
    {
      printf ("命中！");
      if (ysp <= 0)
        {
          printf ("輸送船沈没！\n");
          return YS;
        }
      putchar ('\n');
    }
  return BM;
}
```

C.5 Score

C.5.1 senkan100.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "senkan.h"

char line[100];

void haichi ();
void dispsb ();
int zanteki ();
void shoot ();
int meichup ();
int meichud ();

int tsb[10][10], sb[10][10];
int skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;
int gx=0, gy=0;
int i, kekka, k;
int bx, by;
int fun_c=0, at_flg=0;
int up, rig, dow, lef;

int
main (void)
{
  int j, a, b, total = 0;

  for (j = 1; j <= 100; j++)
    {
      skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;
      fun_c = 0, at_flg = 0;

      haichi ();
      for (a = 0; a < 10; a++)
        {
          for (b = 0; b < 10; b++)
            {
              sb[a][b] = N;
            }
```

```
        }

      zanteki ();
      i = 1;
      shoot ();

      for (i = 2; i <= 101; i++)
        {
          k = meichup ();
          kekka = meichud ();
          if (zanteki () || i == 101)
            {
              break;
            }
          shoot ();
        }
      i--;
      total = total + i;
      printf ("%3d 回目のプレイ：攻撃回数 %3d 回\n", j, i);
    }
  printf ("平均攻撃回数＝ %6.2f\n", total / 100.0);
}

void
haichi ()
{
  int tate, yoko, cx, cy;
  static unsigned int seed = 1;

  srand ((unsigned int) time (NULL) * seed);

  for (tate = 0; tate < 10; tate++)
    {
      for (yoko = 0; yoko < 10; yoko++)
        {
          tsb[tate][yoko] = N;
        }
    }

  if (rand () % 2 == 0)
    {                                   /* 戦艦縦方向 */
      cy = rand () % 10;
      cx = rand () % 6;
```

```
    tsb[cx][cy] = SK;
    tsb[cx + 1][cy] = SK;
    tsb[cx + 2][cy] = SK;
    tsb[cx + 3][cy] = SK;
    tsb[cx + 4][cy] = SK;
  }
else
  {                                 /* 戦艦横方向 */
    cx = rand () % 10;
    cy = rand () % 6;
    tsb[cx][cy] = SK;
    tsb[cx][cy + 1] = SK;
    tsb[cx][cy + 2] = SK;
    tsb[cx][cy + 3] = SK;
    tsb[cx][cy + 4] = SK;
  }

if (rand () % 2 == 0)
  {                                 /* 巡洋艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 7;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N
           || tsb[cx + 3][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 7;
      }
    tsb[cx][cy] = JK;
    tsb[cx + 1][cy] = JK;
    tsb[cx + 2][cy] = JK;
    tsb[cx + 3][cy] = JK;
  }
else
  {                                 /* 巡洋艦横方向 */
    cx = rand () % 10;
    cy = rand () % 7;
    while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N
           || tsb[cx][cy + 3] != N)
      {
        cx = rand () % 10;
        cy = rand () % 7;
      }
    tsb[cx][cy] = JK;
```

```
    tsb[cx][cy + 1] = JK;
    tsb[cx][cy + 2] = JK;
    tsb[cx][cy + 3] = JK;
  }

if (rand () % 2 == 0)
  {                                 /* 駆逐艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 8;
      }
    tsb[cx][cy] = KK;
    tsb[cx + 1][cy] = KK;
    tsb[cx + 2][cy] = KK;
  }
else
  {                                 /* 駆逐艦横方向 */
    cx = rand () % 10;
    cy = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
      {
        cx = rand () % 10;
        cy = rand () % 8;
      }
    tsb[cx][cy] = KK;
    tsb[cx][cy + 1] = KK;
    tsb[cx][cy + 2] = KK;
  }

if (rand () % 2 == 0)
  {                                 /* 潜水艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 8;
      }
    tsb[cx][cy] = SM;
    tsb[cx + 1][cy] = SM;
```

```
      tsb[cx + 2][cy] = SM;
    }
  else
    {                                   /* 潜水艦横方向 */
      cx = rand () % 10;
      cy = rand () % 8;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
        {
          cx = rand () % 10;
          cy = rand () % 8;
        }
      tsb[cx][cy] = SM;
      tsb[cx][cy + 1] = SM;
      tsb[cx][cy + 2] = SM;
    }

  if (rand () % 2 == 0)
    {                                   /* 輸送船縦方向 */
      cy = rand () % 10;
      cx = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N)
        {
          cy = rand () % 10;
          cx = rand () % 9;
        }
      tsb[cx][cy] = YS;
      tsb[cx + 1][cy] = YS;
    }
  else
    {                                   /* 輸送船横方向 */
      cx = rand () % 10;
      cy = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N)
        {
          cx = rand () % 10;
          cy = rand () % 9;
        }
      tsb[cx][cy] = YS;
      tsb[cx][cy + 1] = YS;
    }
  seed = rand ();
}
```

```
void
dispsb ()
{
  int i, j;

  printf ("   1 2 3 4 5 6 7 8 9 10\n");
  printf ("  ---------------------\n");

  for (i = 0; i < 10; i++)
    {
      printf ("%2d", i + 1);
      for (j = 0; j < 10; j++)
        {
          putchar ('|');
          switch (sb[i][j])
            {
            case BM:
              putchar ('O');
              break;
            case BH:
              putchar ('X');
              break;
            default:
              putchar (' ');
              break;
            }
        }
      printf ("|\n");
    }

  printf ("  ---------------------\n");
}

int
zanteki ()
{
  if (skp <= 0 && jkp <= 0 && kkp <= 0 && smp <= 0 && ysp <= 0)
    {
      /* printf("全艦撃沈！\n"); */
      return 1;
    }
/*
        printf("残敵＝");
```

```
        if(skp > 0) {
                printf("戦艦 ");
        }
        if(jkp > 0) {
                printf("巡洋艦 ");
        }
        if(kkp > 0) {
                printf("駆逐艦 ");
        }
        if(smp > 0) {
                printf("潜水艦 ");
        }
        if(ysp > 0) {
                printf("輸送船");
        }
        putchar('\n');
        */
  return 0;
}

int
meichup () /* x->gx, y->gy change */
{
  int a, b;

  a = tsb[gx][gy];
  b = sb[gx][gy];

  if (b != N)
    {
      return BH;
    }
  if (a == SK)
    {
      sb[gx][gy] = BM;
      skp--;
      return SK;
    }
  if (a == JK)
    {
      sb[gx][gy] = BM;
      jkp--;
      return JK;
```

```
    }
  if (a == KK)
    {
      sb[gx][gy] = BM;
      kkp--;
      return KK;
    }
  if (a == SM)
    {
      sb[gx][gy] = BM;
      smp--;
      return SM;
    }
  if (a == YS)
    {
      sb[gx][gy] = BM;
      ysp--;
      return YS;
    }
  sb[gx][gy] = BH;
  return BH;
}

int
meichud ()
{
  if (k == BH)
    {
      return BH;
    }
  else if (k == SK && skp <= 0)
    {
      return SK;
    }
  else if (k == JK && jkp <= 0)
    {
      return JK;
    }
  else if (k == KK && kkp <= 0)
    {
      return KK;
    }
  else if (k == SM && smp <= 0)
```

```
    {
      return SM;
    }
  else if (k == YS && ysp <= 0)
    {
      return YS;
    }
  return BM;
}
```

C.6 攻略関数テンプレート

C.6.1 shoot.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "senkan.h"

void
shoot ()
{

    return;
}
```

付録 D　三回生向け演習 Battle Ship ゲーム

D.1　ヘッダーファイル

D.1.1　senkan.h

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 0   // 未攻撃
#define BM 1  // 命中
#define BH 2  // ハズレ
#define SK 11 // length:5 戦艦沈没
#define JK 12 // length:4 巡洋艦沈没
#define KK 13 // length:3 駆逐艦沈没
#define SM 14 // length:3 潜水艦沈没
#define YS 15 // length:2 輸送艦沈没

extern int tsb[10][10], sb[10][10];
extern int skp, jkp, kkp, smp, ysp;
extern int gx, gy;
extern int i, kekka, k;
extern int bx, by;
extern int at_flg;
```

D.2 攻撃抽象化用関数ファイル

D.2.1 function.c

```
#include <stdio.h>
#include <stdlib.h>
#include "senkan.h"

extern int tsb[10][10], sb[10][10];
extern int skp, jkp, kkp, smp, ysp;
extern int gx, gy;
extern int i, kekka;
extern int bx, by, at_flg;

void random100();
void oneSkip();
void twoSkip();
void twoRight();
void threeSkip();
void threeRight();
void fourSkip();
void fourRight();
void oddSkip();
void evenSkip();
void upAt();
void rightAt();
void downAt();
void leftAt();

void random100()
{
  do
    {
      gx = rand() % 10;
      gy = rand() % 10;
    } while(sb[gx][gy] != N);

  return;
}

void oneSkip()
{
  int tate, yoko;
```

```
  static int finish[10][10] =
    { {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0}
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] == 1 && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void twoSkip()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
    };
```

```
  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void twoRight()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {0, 1, 0, 0, 1, 0, 0, 1, 0, 0},
      {0, 0, 1, 0, 0, 1, 0, 0, 1, 0},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {0, 1, 0, 0, 1, 0, 0, 1, 0, 0},
      {0, 0, 1, 0, 0, 1, 0, 0, 1, 0},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
      {0, 1, 0, 0, 1, 0, 0, 1, 0, 0},
      {0, 0, 1, 0, 0, 1, 0, 0, 1, 0},
      {1, 0, 0, 1, 0, 0, 1, 0, 0, 1},
    };
  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
```

```
  return;
}

void threeSkip()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void threeRight()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {0, 1, 0, 0, 0, 1, 0, 0, 0, 1},
      {0, 0, 1, 0, 0, 0, 1, 0, 0, 0},
      {0, 0, 0, 1, 0, 0, 0, 1, 0, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
```

```
      {0, 1, 0, 0, 0, 1, 0, 0, 0, 1},
      {0, 0, 1, 0, 0, 0, 1, 0, 0, 0},
      {0, 0, 0, 1, 0, 0, 0, 1, 0, 0},
      {1, 0, 0, 0, 1, 0, 0, 0, 1, 0},
      {0, 1, 0, 0, 0, 1, 0, 0, 0, 1},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void fourSkip()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
```

```
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void fourRight()
{
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {0, 1, 0, 0, 0, 0, 1, 0, 0, 0},
      {0, 0, 1, 0, 0, 0, 0, 1, 0, 0},
      {0, 0, 0, 1, 0, 0, 0, 0, 1, 0},
      {0, 0, 0, 0, 1, 0, 0, 0, 0, 1},
      {1, 0, 0, 0, 0, 1, 0, 0, 0, 0},
      {0, 1, 0, 0, 0, 0, 1, 0, 0, 0},
      {0, 0, 1, 0, 0, 0, 0, 1, 0, 0},
      {0, 0, 0, 1, 0, 0, 0, 0, 1, 0},
      {0, 0, 0, 0, 1, 0, 0, 0, 0, 1},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void oddSkip()
{
```

```
  int tate, yoko;
  static int finish[10][10] =
    { {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void evenSkip()
{
  int tate, yoko;
  static int finish[10][10] =
    { {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
      {0, 1, 0, 1, 0, 1, 0, 1, 0, 1},
      {1, 0, 1, 0, 1, 0, 1, 0, 1, 0},
```

```
    };

  for(tate = 0; tate < 10; tate++)
    {
      for(yoko = 0; yoko < 10; yoko++)
        {
          if(finish[tate][yoko] && sb[tate][yoko] == N)
            {
              gx = tate;
              gy = yoko;
              return;
            }
        }
    }
  return;
}

void upAt()
{
    int at_count;

    for(at_count = 1; at_count < 5; at_count++)
        {
            if (0 <= bx - at_count && sb[bx - at_count][by] == N)
                {
                    gx = bx - at_count;
                    gy = by;

                    return;
                }
        }

    return;
}

void rightAt()
{
    int at_count;
    for(at_count = 1; at_count < 5; at_count++)
        {
            if (by + at_count < 10 && sb[bx][by + at_count] == N)
                {
                    gx = bx;
```

```
                    gy = by + at_count;

                    return;
                }
        }

    return;
}

void downAt()
{
    int at_count;

    for(at_count = 1; at_count < 5; at_count++)
        {
            if (bx + at_count < 10 && sb[bx + at_count][by] == N)
                {
                    gx = bx + at_count;
                    gy = by;

                    return;
                }
        }

    return;
}

void leftAt()
{
  int at_count;

  for(at_count = 1; at_count < 5; at_count++)
      {
          if (0 <= by - at_count && sb[bx][by - at_count] == N)
              {
                  gx = bx;
                  gy = by - at_count;

                  return;
              }
      }

  return;
```

```
}
```

D.3 Practice

D.3.1 senkanplay.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char line[100];

#define N 0
#define BM 1
#define BH 2
#define SK 11
#define JK 12
#define KK 13
#define SM 14
#define YS 15

void haichi (int sb[][10]);
void dispsb (int sb[][10]);
int zanteki (int skp, int jkp, int kkp, int smp, int ysp);
void shoot (int sb[][10], int i, int *x, int *y);
int meichup (int sb[][10], int x, int y);
void meichud (int k);

int skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;

int
main (void)
{

  int sb[10][10];
  int i, x, y, k;

  haichi (sb);

  dispsb (sb);

  zanteki (skp, jkp, kkp, smp, ysp);

  shoot (sb, 1, &x, &y);
```

```
  printf ("(x,y) = (%d,%d)\n", x, y);

  for (i = 2; i <= 100; i++)
    {
      k = meichup (sb, x, y);
      dispsb (sb);
      meichud (k);
      if (zanteki (skp, jkp, kkp, smp, ysp))
        {
          break;
        }
      shoot (sb, i, &x, &y);
    }
  printf ("攻撃回数＝%d\n", --i);

  printf ("\nPress any key to continue...\n");
  getchar ();
}

void
haichi (int sb[][10])
{
  int i, j, x, y;

  srand ((unsigned int) time (NULL));

  for (i = 0; i < 10; i++)
    {
      for (j = 0; j < 10; j++)
        {
          sb[i][j] = N;
        }
    }

  if (rand () % 2 == 0)
    {                                   /* 戦艦縦方向 */
      y = rand () % 10;
      x = rand () % 6;
      sb[x][y] = SK;
      sb[x + 1][y] = SK;
      sb[x + 2][y] = SK;
      sb[x + 3][y] = SK;
      sb[x + 4][y] = SK;
```

```
  }
else
  {                                /* 戦艦横方向 */
    x = rand () % 10;
    y = rand () % 6;
    sb[x][y] = SK;
    sb[x][y + 1] = SK;
    sb[x][y + 2] = SK;
    sb[x][y + 3] = SK;
    sb[x][y + 4] = SK;
  }

if (rand () % 2 == 0)
  {                                /* 巡洋艦縦方向 */
    y = rand () % 10;
    x = rand () % 7;
    while (sb[x][y] != N || sb[x + 1][y] != N || sb[x + 2][y] != N
           || sb[x + 3][y] != N)
      {
        y = rand () % 10;
        x = rand () % 7;
      }
    sb[x][y] = JK;
    sb[x + 1][y] = JK;
    sb[x + 2][y] = JK;
    sb[x + 3][y] = JK;
  }
else
  {                                /* 巡洋艦横方向 */
    x = rand () % 10;
    y = rand () % 7;
    while (sb[x][y] != N || sb[x][y + 1] != N || sb[x][y + 2] != N
           || sb[x][y + 3] != N)
      {
        x = rand () % 10;
        y = rand () % 7;
      }
    sb[x][y] = JK;
    sb[x][y + 1] = JK;
    sb[x][y + 2] = JK;
    sb[x][y + 3] = JK;
  }
```

```
if (rand () % 2 == 0)
  {                                 /* 駆逐艦縦方向 */
    y = rand () % 10;
    x = rand () % 8;
    while (sb[x][y] != N || sb[x + 1][y] != N || sb[x + 2][y] != N)
      {
        y = rand () % 10;
        x = rand () % 8;
      }
    sb[x][y] = KK;
    sb[x + 1][y] = KK;
    sb[x + 2][y] = KK;
  }
else
  {                                 /* 駆逐艦横方向 */
    x = rand () % 10;
    y = rand () % 8;
    while (sb[x][y] != N || sb[x][y + 1] != N || sb[x][y + 2] != N)
      {
        x = rand () % 10;
        y = rand () % 8;
      }
    sb[x][y] = KK;
    sb[x][y + 1] = KK;
    sb[x][y + 2] = KK;
  }

if (rand () % 2 == 0)
  {                                 /* 潜水艦縦方向 */
    y = rand () % 10;
    x = rand () % 8;
    while (sb[x][y] != N || sb[x + 1][y] != N || sb[x + 2][y] != N)
      {
        y = rand () % 10;
        x = rand () % 8;
      }
    sb[x][y] = SM;
    sb[x + 1][y] = SM;
    sb[x + 2][y] = SM;
  }
else
  {                                 /* 潜水艦横方向 */
    x = rand () % 10;
```

```
      y = rand () % 8;
      while (sb[x][y] != N || sb[x][y + 1] != N || sb[x][y + 2] != N)
        {
          x = rand () % 10;
          y = rand () % 8;
        }
      sb[x][y] = SM;
      sb[x][y + 1] = SM;
      sb[x][y + 2] = SM;
    }

  if (rand () % 2 == 0)
    {                                   /* 輸送船縦方向 */
      y = rand () % 10;
      x = rand () % 9;
      while (sb[x][y] != N || sb[x + 1][y] != N)
        {
          y = rand () % 10;
          x = rand () % 9;
        }
      sb[x][y] = YS;
      sb[x + 1][y] = YS;
    }
  else
    {                                   /* 輸送船横方向 */
      x = rand () % 10;
      y = rand () % 9;
      while (sb[x][y] != N || sb[x][y + 1] != N)
        {
          x = rand () % 10;
          y = rand () % 9;
        }
      sb[x][y] = YS;
      sb[x][y + 1] = YS;
    }
}

void
dispsb (int sb[][10])
{
  int i, j;

  printf ("   1 2 3 4 5 6 7 8 9 10\n");
```

```
  printf ("  ----------------------\n");

  for (i = 0; i < 10; i++)
    {
      printf ("%2d", i + 1);
      for (j = 0; j < 10; j++)
        {
          putchar ('|');
          switch (sb[i][j])
            {
            case BM:
              putchar ('O');
              break;
            case BH:
              putchar ('X');
              break;
            default:
              putchar (' ');
              break;
            }
        }
      printf ("|\n");
    }

  printf ("  ----------------------\n");
}

int
zanteki (int skp, int jkp, int kkp, int smp, int ysp)
{
  if (skp == 0 && jkp == 0 && kkp == 0 && smp == 0 && ysp == 0)
    {
      printf ("全艦撃沈！\n");
      return 1;
    }

  printf ("残敵＝");
  if (skp > 0)
    {
      printf ("戦艦 ");
    }
  if (jkp > 0)
    {
```

```
      printf ("巡洋艦 ");
    }
  if (kkp > 0)
    {
      printf ("駆逐艦 ");
    }
  if (smp > 0)
    {
      printf ("潜水艦 ");
    }
  if (ysp > 0)
    {
      printf ("輸送船");
    }
  putchar ('\n');
  return 0;
}

void
shoot (int sb[][10], int i, int *x, int *y)
{
  int tate, yoko;

  while (1)
    {
      printf ("次弾(%d)発射位置？ 縦,横 ＝ ", i);
      fgets (line, sizeof (line), stdin);
      sscanf (line, "%d,%d", &tate, &yoko);
      if (tate >= 1 && tate <= 10 && yoko >= 1 && yoko <= 10)
        {
          *x = tate - 1;
          *y = yoko - 1;
          return;
        }
    }
}

int
meichup (int sb[][10], int x, int y)
{
  int a;

  a = sb[x][y];
```

```
  if (a == SK)
    {
      sb[x][y] = BM;
      skp--;
      return SK;
    }
  if (a == JK)
    {
      sb[x][y] = BM;
      jkp--;
      return JK;
    }
  if (a == KK)
    {
      sb[x][y] = BM;
      kkp--;
      return KK;
    }
  if (a == SM)
    {
      sb[x][y] = BM;
      smp--;
      return SM;
    }
  if (a == YS)
    {
      sb[x][y] = BM;
      ysp--;
      return YS;
    }
  if (a == BM)
    {
      return BH;
    }
  sb[x][y] = BH;
  return BH;
}

void
meichud (int k)
{
  if (k == BH)
```

```
  {
    printf ("ハズレ！\n");
  }
else if (k == SK)
  {
    printf ("命中！");
    if (skp <= 0)
      {
        printf ("戦艦沈没！");
      }
    putchar ('\n');
  }
else if (k == JK)
  {
    printf ("命中！");
    if (jkp <= 0)
      {
        printf ("巡洋艦沈没！");
      }
    putchar ('\n');
  }
else if (k == KK)
  {
    printf ("命中！");
    if (kkp <= 0)
      {
        printf ("駆逐艦沈没！");
      }
    putchar ('\n');
  }
else if (k == SM)
  {
    printf ("命中！");
    if (smp <= 0)
      {
        printf ("潜水艦沈没！");
      }
    putchar ('\n');
  }
else if (k == YS)
  {
    printf ("命中！");
    if (ysp <= 0)
```

```
        {
          printf ("輸送船沈没！");
        }
      putchar ('\n');
    }
}
```

D.4 Verification

D.4.1 senkan001.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "senkan.h"

char line[100];

void haichi ();
void dispsb ();
int zanteki ();
void shoot ();
int meichup ();
int meichud ();

int tsb[10][10], sb[10][10];
int skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;
int gx=0, gy=0;
int i, kekka, k;
int bx, by;
int fun_c=0, at_flg=0;
int up, rig, dow, lef;

int
main (void)
{
  int tate, yoko;

  haichi ();
  for (tate = 0; tate < 10; tate++)
    {
      for (yoko = 0; yoko < 10; yoko++)
        {
          sb[tate][yoko] = N;
        }
    }

  dispsb ();
  zanteki ();
```

```
  i = 1;
  shoot ();
  getchar ();

  for (i = 2; i <= 101; i++)
    {
      k = meichup ();
      printf("%d\n", k);
      dispsb ();
      kekka = meichud ();
      if (zanteki () || (i == 101))
        {
          break;
        }
      shoot ();
      // printf("%d %d", x+1, y+1);
      getchar ();
    }
  printf ("攻撃回数＝%d\n", --i);
  printf ("Press any key to continue...\n");
  getchar ();
}

void
haichi ()
{
  int tate, yoko, cx, cy;

  srand ((unsigned int) time (NULL));

  for (tate = 0; tate < 10; tate++)
    {
      for (yoko = 0; yoko < 10; yoko++)
        {
          tsb[tate][yoko] = N;
        }
    }

  if (rand () % 2 == 0)
    {                                   /* 戦艦縦方向 */
      cy = rand () % 10;
      cx = rand () % 6;
      tsb[cx][cy] = SK;
```

```
    tsb[cx + 1][cy] = SK;
    tsb[cx + 2][cy] = SK;
    tsb[cx + 3][cy] = SK;
    tsb[cx + 4][cy] = SK;
  }
else
  {                                /* 戦艦横方向 */
    cx = rand () % 10;
    cy = rand () % 6;
    tsb[cx][cy] = SK;
    tsb[cx][cy + 1] = SK;
    tsb[cx][cy + 2] = SK;
    tsb[cx][cy + 3] = SK;
    tsb[cx][cy + 4] = SK;
  }

if (rand () % 2 == 0)
  {                                /* 巡洋艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 7;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N
           || tsb[cx + 3][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 7;
      }
    tsb[cx][cy] = JK;
    tsb[cx + 1][cy] = JK;
    tsb[cx + 2][cy] = JK;
    tsb[cx + 3][cy] = JK;
  }
else
  {                                /* 巡洋艦横方向 */
    cx = rand () % 10;
    cy = rand () % 7;
    while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N
           || tsb[cx][cy + 3] != N)
      {
        cx = rand () % 10;
        cy = rand () % 7;
      }
    tsb[cx][cy] = JK;
    tsb[cx][cy + 1] = JK;
```

```
    tsb[cx][cy + 2] = JK;
    tsb[cx][cy + 3] = JK;
  }

if (rand () % 2 == 0)
  {                                 /* 駆逐艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 8;
      }
    tsb[cx][cy] = KK;
    tsb[cx + 1][cy] = KK;
    tsb[cx + 2][cy] = KK;
  }
else
  {                                 /* 駆逐艦横方向 */
    cx = rand () % 10;
    cy = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
      {
        cx = rand () % 10;
        cy = rand () % 8;
      }
    tsb[cx][cy] = KK;
    tsb[cx][cy + 1] = KK;
    tsb[cx][cy + 2] = KK;
  }

if (rand () % 2 == 0)
  {                                 /* 潜水艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 8;
      }
    tsb[cx][cy] = SM;
    tsb[cx + 1][cy] = SM;
    tsb[cx + 2][cy] = SM;
```

```
    }
  else
    {                                   /* 潜水艦横方向 */
      cx = rand () % 10;
      cy = rand () % 8;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
        {
          cx = rand () % 10;
          cy = rand () % 8;
        }
      tsb[cx][cy] = SM;
      tsb[cx][cy + 1] = SM;
      tsb[cx][cy + 2] = SM;
    }

  if (rand () % 2 == 0)
    {                                   /* 輸送船縦方向 */
      cy = rand () % 10;
      cx = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N)
        {
          cy = rand () % 10;
          cx = rand () % 9;
        }
      tsb[cx][cy] = YS;
      tsb[cx + 1][cy] = YS;
    }
  else
    {                                   /* 輸送船横方向 */
      cx = rand () % 10;
      cy = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N)
        {
          cx = rand () % 10;
          cy = rand () % 9;
        }
      tsb[cx][cy] = YS;
      tsb[cx][cy + 1] = YS;
    }
}

void
dispsb ()
```

```
{
  int tate, yoko;

  printf ("   1 2 3 4 5 6 7 8 9 10\n");
  printf ("  ---------------------\n");

  for (tate = 0; tate < 10; tate++)
    {
      printf ("%2d", tate + 1);
      for (yoko = 0; yoko < 10; yoko++)
        {
          putchar ('|');
          switch (sb[tate][yoko])
            {
            case BM:
              putchar ('O');
              break;
            case BH:
              putchar ('X');
              break;
            default:
              putchar (' ');
              break;
            }
        }
      printf ("|\n");
    }

  printf ("  ---------------------\n");
}

int
zanteki ()
{
  if (skp <= 0 && jkp <= 0 && kkp <= 0 && smp <= 0 && ysp <= 0)
    {
      printf ("全艦撃沈！\n");
      return 1;
    }

  printf ("残敵＝");
  if (skp > 0)
    {
```

```
      printf ("戦艦 ");
    }
  if (jkp > 0)
    {
      printf ("巡洋艦 ");
    }
  if (kkp > 0)
    {
      printf ("駆逐艦 ");
    }
  if (smp > 0)
    {
      printf ("潜水艦 ");
    }
  if (ysp > 0)
    {
      printf ("輸送船");
    }
  putchar ('\n');
  return 0;
}


int
meichup ()
{
  int a, b;

  a = tsb[gx][gy];
  b = sb[gx][gy];

  if (b != N)
    {
      return BH;
    }
  if (a == SK)
    {
      sb[gx][gy] = BM;
      skp--;
      return SK;
    }
  if (a == JK)
    {
```

```
      sb[gx][gy] = BM;
      jkp--;
      return JK;
    }
  if (a == KK)
    {
      sb[gx][gy] = BM;
      kkp--;
      return KK;
    }
  if (a == SM)
    {
      sb[gx][gy] = BM;
      smp--;
      return SM;
    }
  if (a == YS)
    {
      sb[gx][gy] = BM;
      ysp--;
      return YS;
    }
  sb[gx][gy] = BH;
  return BH;
}

int
meichud ()
{
  if (k == BH)
    {
      printf ("ハズレ！\n");
      return BH;
    }
  else if (k == SK)
    {
      printf ("命中！");
      if (skp <= 0)
        {
          printf ("戦艦沈没！\n");
          return SK;
        }
      putchar ('\n');
```

```
    }
  else if (k == JK)
    {
      printf ("命中！");
      if (jkp <= 0)
        {
          printf ("巡洋艦沈没！\n");
          return JK;
        }
      putchar ('\n');
    }
  else if (k == KK)
    {
      printf ("命中！");
      if (kkp <= 0)
        {
          printf ("駆逐艦沈没！\n");
          return KK;
        }
      putchar ('\n');
    }
  else if (k == SM)
    {
      printf ("命中！");
      if (smp <= 0)
        {
          printf ("潜水艦沈没！\n");
          return SM;
        }
      putchar ('\n');
    }
  else if (k == YS)
    {
      printf ("命中！");
      if (ysp <= 0)
        {
          printf ("輸送船沈没！\n");
          return YS;
        }
      putchar ('\n');
    }
  return BM;
}
```

D.5 Score

D.5.1 senkan100.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "senkan.h"

char line[100];

void haichi ();
void dispsb ();
int zanteki ();
void shoot ();
int meichup ();
int meichud ();

int tsb[10][10], sb[10][10];
int skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;
int gx=0, gy=0;
int i, kekka, k;
int bx, by;
int at_flg=0;

int
main (void)
{
  int j, a, b, total = 0;

  for (j = 1; j <= 100; j++)
    {
      skp = 5, jkp = 4, kkp = 3, smp = 3, ysp = 2;
      at_flg = 0;

      haichi ();
      for (a = 0; a < 10; a++)
        {
          for (b = 0; b < 10; b++)
            {
              sb[a][b] = N;
            }
        }
```

```
      zanteki ();
      i = 1;
      shoot ();

      for (i = 2; i <= 101; i++)
        {
          k = meichup ();
          kekka = meichud ();
          if (zanteki () || i == 101)
            {
              break;
            }
          shoot ();
        }
      i--;
      total = total + i;
      printf ("%3d 回目のプレイ：攻撃回数 %3d 回\n", j, i);
    }
  printf ("平均攻撃回数＝ %6.2f\n", total / 100.0);
}

void
haichi ()
{
  int tate, yoko, cx, cy;
  static unsigned int seed = 1;

  srand ((unsigned int) time (NULL) * seed);

  for (tate = 0; tate < 10; tate++)
    {
      for (yoko = 0; yoko < 10; yoko++)
        {
          tsb[tate][yoko] = N;
        }
    }

  if (rand () % 2 == 0)
    {                                   /* 戦艦縦方向 */
      cy = rand () % 10;
      cx = rand () % 6;
      tsb[cx][cy] = SK;
```

```
    tsb[cx + 1][cy] = SK;
    tsb[cx + 2][cy] = SK;
    tsb[cx + 3][cy] = SK;
    tsb[cx + 4][cy] = SK;
  }
else
  {                                /* 戦艦横方向 */
    cx = rand () % 10;
    cy = rand () % 6;
    tsb[cx][cy] = SK;
    tsb[cx][cy + 1] = SK;
    tsb[cx][cy + 2] = SK;
    tsb[cx][cy + 3] = SK;
    tsb[cx][cy + 4] = SK;
  }

if (rand () % 2 == 0)
  {                                /* 巡洋艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 7;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N
           || tsb[cx + 3][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 7;
      }
    tsb[cx][cy] = JK;
    tsb[cx + 1][cy] = JK;
    tsb[cx + 2][cy] = JK;
    tsb[cx + 3][cy] = JK;
  }
else
  {                                /* 巡洋艦横方向 */
    cx = rand () % 10;
    cy = rand () % 7;
    while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N
           || tsb[cx][cy + 3] != N)
      {
        cx = rand () % 10;
        cy = rand () % 7;
      }
    tsb[cx][cy] = JK;
    tsb[cx][cy + 1] = JK;
```

```
    tsb[cx][cy + 2] = JK;
    tsb[cx][cy + 3] = JK;
  }

if (rand () % 2 == 0)
  {                                 /* 駆逐艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 8;
      }
    tsb[cx][cy] = KK;
    tsb[cx + 1][cy] = KK;
    tsb[cx + 2][cy] = KK;
  }
else
  {                                 /* 駆逐艦横方向 */
    cx = rand () % 10;
    cy = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
      {
        cx = rand () % 10;
        cy = rand () % 8;
      }
    tsb[cx][cy] = KK;
    tsb[cx][cy + 1] = KK;
    tsb[cx][cy + 2] = KK;
  }

if (rand () % 2 == 0)
  {                                 /* 潜水艦縦方向 */
    cy = rand () % 10;
    cx = rand () % 8;
    while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N || tsb[cx + 2][cy] != N)
      {
        cy = rand () % 10;
        cx = rand () % 8;
      }
    tsb[cx][cy] = SM;
    tsb[cx + 1][cy] = SM;
    tsb[cx + 2][cy] = SM;
```

```
    }
  else
    {                                   /* 潜水艦横方向 */
      cx = rand () % 10;
      cy = rand () % 8;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N || tsb[cx][cy + 2] != N)
        {
          cx = rand () % 10;
          cy = rand () % 8;
        }
      tsb[cx][cy] = SM;
      tsb[cx][cy + 1] = SM;
      tsb[cx][cy + 2] = SM;
    }

  if (rand () % 2 == 0)
    {                                   /* 輸送船縦方向 */
      cy = rand () % 10;
      cx = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx + 1][cy] != N)
        {
          cy = rand () % 10;
          cx = rand () % 9;
        }
      tsb[cx][cy] = YS;
      tsb[cx + 1][cy] = YS;
    }
  else
    {                                   /* 輸送船横方向 */
      cx = rand () % 10;
      cy = rand () % 9;
      while (tsb[cx][cy] != N || tsb[cx][cy + 1] != N)
        {
          cx = rand () % 10;
          cy = rand () % 9;
        }
      tsb[cx][cy] = YS;
      tsb[cx][cy + 1] = YS;
    }
  seed = rand ();
}

void
```

```
dispsb ()
{
  int i, j;

  printf ("   1 2 3 4 5 6 7 8 9 10\n");
  printf ("  ---------------------\n");

  for (i = 0; i < 10; i++)
    {
      printf ("%2d", i + 1);
      for (j = 0; j < 10; j++)
        {
          putchar ('|');
          switch (sb[i][j])
            {
            case BM:
              putchar ('O');
              break;
            case BH:
              putchar ('X');
              break;
            default:
              putchar (' ');
              break;
            }
        }
      printf ("|\n");
    }

  printf ("  ---------------------\n");
}

int
zanteki ()
{
  if (skp <= 0 && jkp <= 0 && kkp <= 0 && smp <= 0 && ysp <= 0)
    {
      /* printf("全艦撃沈！\n"); */
      return 1;
    }
/*
        printf("残敵＝");
        if(skp > 0) {
```

```
                printf("戦艦 ");
        }
        if(jkp > 0) {
                printf("巡洋艦 ");
        }
        if(kkp > 0) {
                printf("駆逐艦 ");
        }
        if(smp > 0) {
                printf("潜水艦 ");
        }
        if(ysp > 0) {
                printf("輸送船");
        }
        putchar('\n');
        */
  return 0;
}

int
meichup () /* x->gx, y->gy change */
{
  int a, b;

  a = tsb[gx][gy];
  b = sb[gx][gy];

  if (b != N)
    {
      return BH;
    }
  if (a == SK)
    {
      sb[gx][gy] = BM;
      skp--;
      return SK;
    }
  if (a == JK)
    {
      sb[gx][gy] = BM;
      jkp--;
      return JK;
    }
```

```
  if (a == KK)
    {
      sb[gx][gy] = BM;
      kkp--;
      return KK;
    }
  if (a == SM)
    {
      sb[gx][gy] = BM;
      smp--;
      return SM;
    }
  if (a == YS)
    {
      sb[gx][gy] = BM;
      ysp--;
      return YS;
    }
  sb[gx][gy] = BH;
  return BH;
}

int
meichud ()
{
  if (k == BH)
    {
      return BH;
    }
  else if (k == SK && skp <= 0)
    {
      return SK;
    }
  else if (k == JK && jkp <= 0)
    {
      return JK;
    }
  else if (k == KK && kkp <= 0)
    {
      return KK;
    }
  else if (k == SM && smp <= 0)
    {
```

```
      return SM;
    }
  else if (k == YS && ysp <= 0)
    {
      return YS;
    }
  return BM;
}
```

D.6 攻略関数テンプレート

D.6.1 shoot.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "senkan.h"

void
shoot ()
{
    if (!at_flg && kekka == BM)
        {
            bx = gx;
            by = gy;
            at_flg = 1;
        }
    else if (at_flg && kekka == BH)
        {
            at_flg = 0;
        }

  return;
}
```