平成２９年度教員免許状更新講習

スクラッチ入門

北海道教育大学釧路校会場 平成２９年８月７日

目 次

1 はじめに

最近指導要領に「プログラミング的思考」という単語が現れました。「プログラミング的思考」とは何か、学校では何を教えればよいのかといった疑問があると思います。単純には、文部科学省のサイトにある「プログラミング実践ガイド」

(http://jouhouka.mext.go.jp/school/programming_zirei/) にある事例を参考に指導を行えばよいと言えます。しかし、それらの事例ににどういう力を伸ばすねらいがあるのか、そのためにどうしてその実践に至ったのかなどはなかなか見えてこない場合もあります。本講習では、そういったことについても理解が進むよう、まず「プログラミング的思考」についての概説をします。

続いて、「プログラミング実践ガイド」の 12 事例のうち 2 事例で使用されてもいるプログラミング言語「Scratch」の基礎を解説します。文部科学省が開発したプログラミング言語「プログラミン」も「プログラミング実践ガイド」に 1 事例がありますが、その「プログラミン」も「Scratch」を参考に作られています。

図 1: 「プログラミン」は「Scratch」を参考に開発された

本講習では、「Scratch」を用いて実際に授業を作ってみるという所までは踏み込めませんが、どういうことをできるプログラミング言語なのかということは、ある程度解説します。

この講習が、わずかでも日頃の教育活動のお役に立てばさいわいです。

2　プログラミング入門

2.1　プログラミング的思考

平成 28 年 6 月 16 日、「小学校段階における論理的思考力や創造性、問題解決能力等の育成とプログラミング教育に関する有識者会議」による、「小学校段階におけるプログラミング教育の在り方について (議論の取りまとめ)」(以下「議論の取りまとめ」(http://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/houkoku/1372522.htm) が出されており、「3. 学校教育におけるプログラミング教育の在り方とは」の「(2) 学校教育として実施するプログラミング教育は何を目指すのか」に、「プログラミング的思考」の定義が載っています。

図 2: プログラミング的思考（「取りまとめ」3. (2) より）

【思考力・判断力・表現力等】
・　発達の段階に即して、「プログラミング的思考」（自分が意図する一連の活動を実現するために、どのような動きの組合せが必要であり、一つ一つの動きに対応した記号を、どのように組み合わせたらいいのか、記号の組合せをどのように改善していけば、より意図した活動に近づくのか、といったことを論理的に考えていく力） [5]を育成すること。

---

[5]いわゆる「コンピュテーショナル・シンキング」の考え方を踏まえつつ、プログラミングと論理的思考との関係を整理しながら提言された定義である。

この定義には、「コンピュータ」や「プログラム」の語が脚注にしかないことに注意しておきます。ただし、小学校学習指導要領の総則には、下のように、プログラミングを体験すると記されています。

図 3: 小学校学習指導要領の総則

(3) 第２の２の(1)に示す情報活用能力の育成を図るため，各学校において，コンピュータや情報通信ネットワークなどの情報手段を活用するために必要な環境を整え，これらを適切に活用した学習活動の充実を図ること。また，各種の統計資料や新聞，視聴覚教材や教育機器などの教材・教具の適切な活用を図ること。

あわせて，各教科等の特質に応じて，次の学習活動を計画的に実施すること。

ア　児童がコンピュータで文字を入力するなどの学習の基盤として必要となる情報手段の基本的な操作を習得するための学習活動

イ　児童がプログラミングを体験しながら，コンピュータに意図した処理を行わせるために必要な論理的思考力を身に付けるための学習活動

「取りまとめ」に戻ると、冒頭の「有識者会議における議論の視野」において、次のように述べられています。

図 4: コーディングが目的ではない（「取りまとめ」有識者会議における議論の視野より）

○ 小学校段階におけるプログラミング教育については、学校と民間が連携した意欲的な取組が広がりつつある一方で、コーディング（プログラミング言語を用いた記述方法）を覚えることがプログラミング教育の目的であるとの誤解が広がりつつあるのではないかとの指摘もある。“小さいうちにコーディングを覚えさせないと子供が将来苦労するのではないか”といった保護者の心理からの過熱ぶりや、反対に“コーディングは時代によって変わるから、プログラミング教育に時間をかけることは全くの無駄ではないか”といった反応も、こうした誤解に基づくものではないかと考えられる。

○ プログラミング教育とは、子供たちに、コンピュータに意図した処理を行うよう指示することができるということを体験させながら、将来どのような職業に就くとしても、時代を超えて普遍的に求められる力としての「プログラミング的思考」などを育むことであり、コーディングを覚えることが目的ではない。こうしたプログラミング教育についての考え方や、小学校段階における具体的な在り方等を、下記3.や4.において示している。

さらには、次のような記述もあります。

図 5: アンプラグドコンピュータサイエンス（「取りまとめ」4. (2) より）

○ プログラミング教育を実施することとなった教科等においては、上記の指導事例集等を参考に、各教科等の指導内容を学びながら、コンピュータに意図した処理を行うよう指示することができるということを体験[10]することを、各教科等の特質に応じた見方・考え方を働かせた「主体的･対話的で深い学び」の中で実現し、各教科等における教育の強みとプログラミング教育のよさが相乗効果を生むような指導内容を具体化していくことが望まれる。

[10]こうした体験については、コンピュータを活用して行うことが原則になると考えられるが、「アンプラグドコンピュータサイエンス」の考え方のもと、コンピュータを使わずに紙と鉛筆で行う教育も提案されているところであり、小学校段階における具体的な教材や指導方法、その効果等について検討が求められる。

「アンプラグドコンピュータサイエンス」は比較的新しい考え方で [BWF]、これは、一切コンピュータを用いずに情報教育を行おうという試みです。教える側からすると、コンピュータ等の知識を新たに学ばなくとも、あるいは、様々な障害を克服してコンピュータ教育の環境を整えることをしなくとも「プログラミング的思考」を教え始められる可能性があります。アンプラグドコンピュータサイエンスの考え方とは違うのですが、コンピュータを用いずに「プログラミング的思考」の教育を行う研究を既に始めている小学校もあります（校名は伏せます）。また、福岡教育大学附属小学校では、アンプラグドコンピュータサイエンスの考え方に基づいて、学び始めからしばらくはコンピュータを用いず、最後の方で少しだけコンピュータを用いるというよう方法での授業を既に研究しています。

他方、見方を変えると、コンピュータを用いただけでは、（文部科学省の目指す）コンピュータ教育にはならないということでもあり、やはり、「プログラミング的思考」をしっかり理解することが重要といえます。

「取りまとめ」には、各学校でのプログラミング教育の在り方が示されています。

図 6: 小中高でのプログラミング教育の在り方（「取りまとめ」 4. (1) より）

○ 小学校におけるプログラミング教育が目指すのは、前述のように、子供たちが、コンピュータに意図した処理を行うよう指示することができるということを体験しながら、身近な生活でコンピュータが活用されていることや、問題の解決には必要な手順があることに気付くこと、各教科等で育まれる思考力を基盤としながら基礎的な「プログラミング的思考」を身に付けること、コンピュータの働きを自分の生活に生かそうとする態度を身に付けることである。
○ 中学校や高等学校の段階では、簡単なプログラムの作成や、コンピュータの働きの科学的な理解などを目指し、技術・家庭科や情報科において構造化された内容を体系的に学んでいくことが必要となる。一方で、小学校におけるプログラミング教育が目指す、身近な生活の中での気付きを促したり、各教科等で身に付いた思考力を「プログラミング的思考」につなげたり、コンピュータの働きが身近な様々な場面で役立っていることを実感しながら自分の生活に生かそうとしたりするためには、学級担任制のメリットを生かしながら、教育課程全体を見渡した中で、プログラミング教育を行う単元を各学校が適切に位置付け、実施していくことが効果的であると考えられる[7]。

中学校・高等学校ではプログラムの作成は求められているようです。また、次のように、生徒がコンピュータ教育を受ける機会は大幅に増えることになります。

図 7: 中高でのプログラミング教育の今後（「取りまとめ」 3. (3) より）

○　具体的には、中学校技術・家庭科技術分野の「情報に関する技術」において、計測・制御に関するプログラミングだけではなく、コンテンツに関するプログラミングを指導内容に盛り込むことによって、プログラミングに関する内容を倍増させること、高等学校情報科に共通必履修科目を新設し、全ての高校生[6]がプログラミングを問題解決に活用することを学べるようにすることが検討されている。

また、次のように、「主体的・対話的で深い学び」にプログラミング教育も貢献することが求められています。また、当たり前ですが楽しいだけでもいけないことが指摘されています。

図 8: 主体的・対話的で深い学び（「取りまとめ」 4. (1) より）

○ プログラミング教育の実施に当たっては、コーディングを覚えることが目的ではないこと[8]を明確に共有していくことが不可欠である。また、「主体的・対話的で深い学び」の実現に資するプログラミング教育とすることが重要であり、一人で黙々とコンピュータに向かっているだけで授業が終わったり、子供自身の生活や体験と切り離された抽象的な内容に終始したりすることがないよう、留意が必要である。楽しく学んでコンピュータに触れることが好きになることが重要であるが、一方で、楽しいだけで終わっては学校教育としての学習成果に結びついたとは言えず、子供たちの感性や学習意欲に働きかけるためにも不十分である。学習を通じて、子供たちが何に気付き、何を理解し、何を身に付けるようにするのかといった、指導上のねらいを明確にする必要がある。

では、どのような授業を展開すればよいのかということになりますが、算数における留意点が示されていますので、少し長いですが下に引用します。

図 9: 算数での留意点（「取りまとめ」 4. (2) より）

【算数】
・「計算する」という過程は、算数・数学の学習においても、日常生活においても、繰り返し行うことが必要となる場面である。繰り返し行うことが必要となる場面というものは、プログラミングで実行する必要性につながりやすいため、“計算することをプログラミングで教えればいいのではないか”と考えられる可能性ある。
・しかしながら、私たちが計算するときには、プログラミングで表現しなくても、人間の文明が生み出した遺産である「筆算」で計算することができる。小学校で筆算を学習するということは、計算の手続を一つ一つのステップに分解し、記憶し反復し、それぞれの過程を確実にこなしていくということであり、これは、プログラミングの一つ一つの要素に対応する[12]。つまり、筆算の学習は、プログラミング的思考の素地（そじ）を体験していることであり、プログラミングを用いずに計算を行うことが、プログラミング的思考につながっていく。
・算数において、プログラミングの体験をどこに位置付けていくかについては、こうしたことを踏まえながら、効果的な場面を考えていかなければならない。例えば、図の作成等において、プログラミングを体験しながら考え、プログラミング的思考と数学的な思考の関係やそれらのよさに気付く学びを取り入れていくことなどが考えられる。
・実施に当たっては、プログラミングを体験することが、算数における学びの本質である数学的活動として適切に位置付けられるようにすることとともに、子供一人一人に探究的な学びが実現し、一層充実するものとなるように十分配慮することが必要である。プログラミングを体験することによる数学的活動が、算数における「深い学び」の達成に寄与するものになることが求められる。
・なお、言うまでもないが、算数における文章題の解決は、文章から数量の関係について情報を読み取り、それらの関係を明らかにし、解決の方法を立案して解決するという過程を体験する活動であり、文章題のストーリーをプログラミングによって単にアニメーション化するようなことは、数学的活動とはならないことなどは、改めて確認しておく必要がある。

[12]コンピュータ科学等でも用いられる「アルゴリズム」とは、筆算といった計算の手続も含む、問題を解決する手順を定式化して表したものを指す。筆算は数学の歴史の中で初期から存在したものではなく、長い年月をかけて人類が生み出したアルゴリズムであり、そうしたものを生み出す人間の数学的な思考が、人工知能の動きや働きなどを支えるおおもととなっている。これからの算数では、筆算が所与のものではなく、こうした意義を持つものであることなどを学ばせることも重要ではないかと考えられる。

最後に「アルゴリズム」の語が出てきましたが、「プログラミング的思考」は「プログラム」よりも「アルゴリズム」の方がぴったりくると言えます（しかし「アルゴリズム的思考」を採用するのは勇気が必要な決断だと思います）。筆算を学ぶことが「プログラミング的思考」を学ぶことになるかも知れません。そのためには、計算練習をする方向ではなく、アルゴリズムにあたる要素を抽出して、なぜこれで計算できるのか、この方法で本当にすべての場合の計算が可能なのか、複数回

繰り上がりが発生する場合のこともアルゴリズムとして記述できているか、などを探究する方向で学ぶことになるでしょう（簡単なことではなさそうです）。

「取りまとめ」からの引用の最後として、教員の養成・研修についての部分を記します。

図 10: 教員の養成・研修（「取りまとめ」 5. (2) より）

○ 教員の養成・研修に当たっては、ICTやアプリケーションの使い方そのものが目的ではなく、プログラミングを経験させるなど、子供たちに育む「プログラミング的思考」の意義や、質の高いプログラミング教育を実現するための授業の工夫や在り方等についての研修が図られるべきであること、また、コンピュータ科学分野の高度な知識が必要というわけではないこと等に留意が必要である。

というわけで、本講習では、前半で「プログラミング的思考」に触れた後、後半でプログラミング言語「Scratch」を紹介し、実際にプログラムを作成することも行ってみようと思います。

2.2 学習指導要領・学習指導要領解説

小学校学習指導要領には、上述した総則の部分の他にも、算数での内容の取り扱いとして、正多角形の作図がプログラミングの体験の一例として示されています。

図 11: 算数、第 5 学年、図形

(2) 数量や図形についての感覚を豊かにしたり，表やグラフを用いて表現する力を高めたりするなどのため，必要な場面においてコンピュータなどを適切に活用すること。また，第 1 章総則の第 3 の 1 の(3)のイに掲げるプログラミングを体験しながら論理的思考力を身に付けるための学習活動を行う場合には，児童の負担に配慮しつつ，例えば第 2 の各学年の内容の〔第 5 学年〕の「B 図形」の(1)における正多角形の作図を行う学習に関連して，正確な繰り返し作業を行う必要があり，更に一部を変えることでいろいろな正多角形を同様に考えることができる場面などで取り扱うこと。

また、引用はしませんが、理科や総合的な学習の時間においても、例示がなされています。

さらに、小学校学習指導要領解説算数編には、「取りまとめ」や学習指導要領の記述を受けて、筆算や正多角形について触れられています。

図 12: 指導計画の作成と内容の取扱い

その際，小・中・高等学校を見通した学びの過程の中で，「主体的･対話的で深い学び」の実現に資するプログラミング教育とすることが重要である。小学校においては，教科等における学習上の必要性や学習内容と関連付けながらプログラミング教育を行う単元を位置付け，身近な生活でコンピュータが活用されていることや，問題の解決には必要な手順があることに気付くことを重視する。

算数科において，プログラミングを体験しながら論理的思考力を身に付けるための活動を行う場合には，算数科の目標を踏まえ，数学的な思考力・判断力・表現力等を身に付ける活動の中で行うものとする。

算数科においては，問題解決したのち，問題解決の仕方を振り返り，問題解決の方法をより簡潔・明瞭・的確なものに高めたり，それを手順としてまとめたりするという学習活動が多く行われる。例えば，整数などの計算の仕方を考えた後，計算の仕方を簡潔・明瞭・的確なものとしていく中で，筆算という形式で表し，計算の仕方を筆算の手順としてまとめていく。筆算として計算の仕方をまとめた後は，手順通りに間違いなく筆算を行うことが大切になる。これは技能である。

このように算数科の学習は，問題の解決には必要な手順があることに気付くことに資するものである。

「プログラミング的思考」とは，自分が意図する一連の活動を実現するために，どのような動きの組み合わせが必要か，どのように改善していけばより意図した活動に近づくのかということを論理的に考えていく力の一つである。

算数科においては，「例えば第２の各学年の内容の〔第５学年〕の「Ｂ図形」の(1)における正多角形の作図を行う学習に関連して，正確な繰り返し作業を行う必要があり，更に一部を変えることでいろいろな正多角形を同様に考えることができる場面などで取り扱うこと。」と示されている。

正多角形の学習では「正多角形は円に内接すること」を基に定規とコンパスなどを用いてかくことを指導する。コンピュータを用いると，「正多角形は全ての辺の長さや角の大きさが等しいこと」を基に簡単にかつ正確にかくことができる。また，辺の長さや角の大きさを適切に変えれば，ほかの正多角形もすぐにかくことができる。

辺の長さ分だけ線を引き，角の大きさ分向きを変え，これらのことを繰り返すことで正多角形がかける。正方形は90度向きを変えればよいが，正六角形は何度にすればいいのかを考えていく。線の動きを示す指示として「線を引く」「○度向きを変える」「繰り返す」などの最小限の指示を指定することで，正多角形をかくことができるのである。

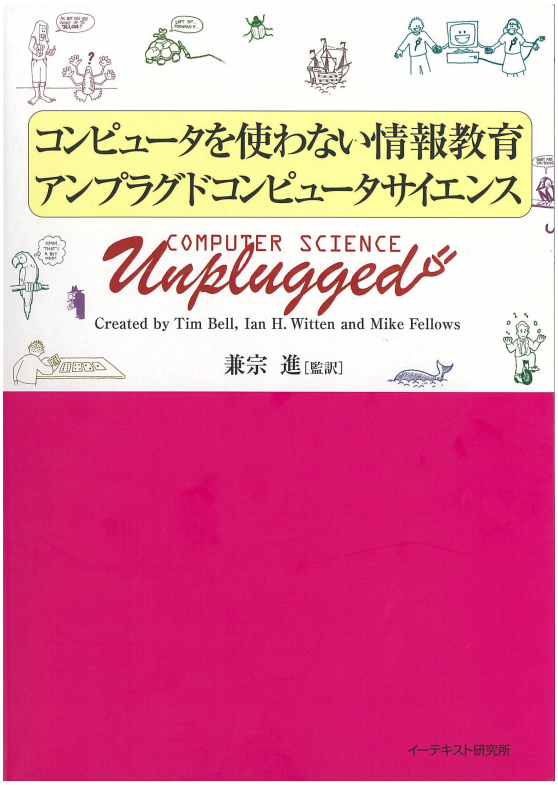
算数科ではこのような活動を行うことで，問題の解決には必要な手順があることと，正確な繰り返しが必要な作業をする際にコンピュータを用いるとよいことに気付かせることができる。

中学校になると、技術・家庭で触れられているのみで、数学での記述はないように思われます。高校数学は、まだ次期指導要領が出来ていませんが、現行のものですと、ユークリッドの互除法において、かつて「数値計算とコンピュータ」に配置されていたという書き方がされているのみです。

2.3 アンプラグドコンピュータサイエンス

上で出てきた「アンプラグドコンピュータサイエンス」について、[BWF] を紹介してみようと思います。これは、ニュージーランドで書かれた本で和訳も 10 年程前に出ています。本も購入できますし、pdf も無料でダウンロードできます (`http://csunplugged.org/books/`)。

図 13: アンプラグドコンピュータサイエンス



目次

2.3.1 2進法

まず、2進法の学習についてを紹介します。これは、コンピュータの基礎知識といってよいものです。最初から2ページにある部分は、恐らく指導案にあたる部分と思われます（米国のを見たことがありますがこのような感じでした)。

学習１　点を数える

２進数

概要

コンピュータの中のデータは０と１の列の形で格納され、転送されます。たった２つの記号を使って、どうやったら文字や数字を表現できるのでしょう？

教科学習との関連

・Mathematics: Number Level 2 and up. Exploring numbers in other bases. Representing numbers in base two.

※［数と計算］２進法や他の記数法で数を表すこと

・Mathematics: Algebra Level 2 and up. Continue a sequential pattern, and describe a rule for this pattern. Patterns and relationships in powers of two.

※［数列］はじめの数列の何項かをみて続きを書いたり、数列を生成している規則を記述したりすること。特に公比２の等比数列でのパターンと関係。

技能

・Counting　数を数える。

・Matching　他の情報の表しかたに対応させる。

・Sequencing　順番に並べる。

年齢

・７歳以上

教材

・説明のために、５種類の２進数カード (6 ページを参照) を作る必要があります。点の代りにスマイルマークのシールが貼ってある A4 のカードも効果的です。

＜子どもごとに必要なもの＞

・５種類のカード

・２進数が印刷されたカード (6 ページ) をコピーして、切り取ってください。

・ワークシート学習：２進数 (5 ページ)

＜必要に応じてさらに学習を選ぶことができます＞

・ワークシート学習：２進数を計算する (7 ページ)

・ワークシート学習：秘密のメッセージを送ろう (8 ページ)

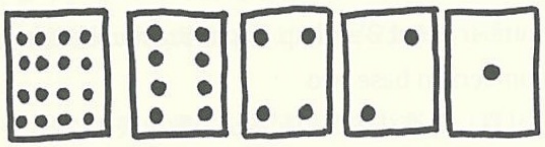
・ワークシート学習：電子メールとモデム (9 ページ)

・ワークシート学習：31 より大きい数を数える (10 ページ)

・ワークシート学習：２進数のあれこれ (11 ページ)

基本的原理

5ページのワークシートに進む前に、基本的な原理を説明しておきます。この学習では、ここに示す5種類のカードが必要です。片面だけに点が描かれています。教室の前でカードを持ってもらうために、5人の子どもを選びましょう。カードは次の順番に並べる必要があります。



討論

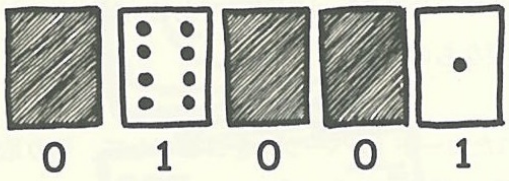
カードの点の数に気づいたでしょうか（それぞれのカードは、右のカードの倍の数になっています）。左にもう一枚加えるとしたら、点はいくつ必要ですか（32）。その次は？

このカードをめくることで、数を作ることができます。子どもに6を作らせてみてください（4と2のカード）。次に15(8と4と2と1のカード)、次に21(16と4と1)、...

次に、0から順に数え上げさせてみましょう。

見ている子どもたちにカードがどのようなパターンで反転するかを観察させます（それぞれのカードは、その右のカードの半分の頻度で反転します）。2つ以上のグループで試してもよいでしょう。

カードは、表が見えていないときは0を表し、見えているときは1を表します。これが2進法の仕組みです。



子どもに01001を作らせます。10進数だと何の数になりますか（9）。17は2進数でどう書きますか（10001）。子どもが理解するまで、いくつかの課題を与えます。

理解を充実するために、5個の選択学習が用意されています。子どもが理解できるまで行いましょう。

解説シート

２進数

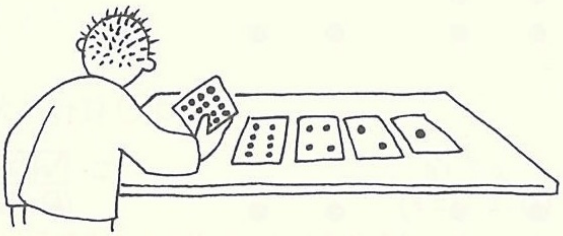
数え方を学ぼう

数え方は知っている？そうですね、ここでは新しいやり方を学びましょう！

コンピュータは０と１しか使えないことを知っていましたか？コンピュータで見えるものや聞こえるもの、たとえば文字、絵、数、動画、音などは、すべて０と１だけで表現されています。今回の学習を体験すれば、コンピュータと同じやり方で、友達に秘密のメッセージが送れるようになります。

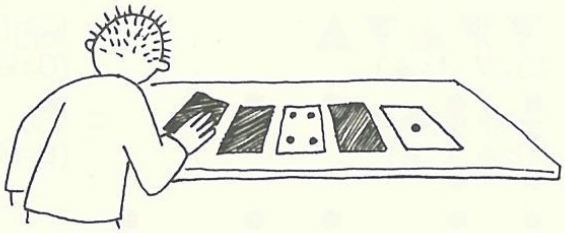
説明

カードをシートから切りぬいて、16 個の点のカードを左にして、図のように順に並べます。



カードはきちんと図の順番に並べるように気をつけましょう。

次に、カードを裏返して、合わせて５個の点だけが見えるようにします。カードの順番は変えてはいけません！



３と 12 と 19 を作ってみよう。

ある数を表すのに、２通り以上の並べ方はありますか？

これらを使って表せるいちばん大きい数は何ですか？

いちばん小さな数は？

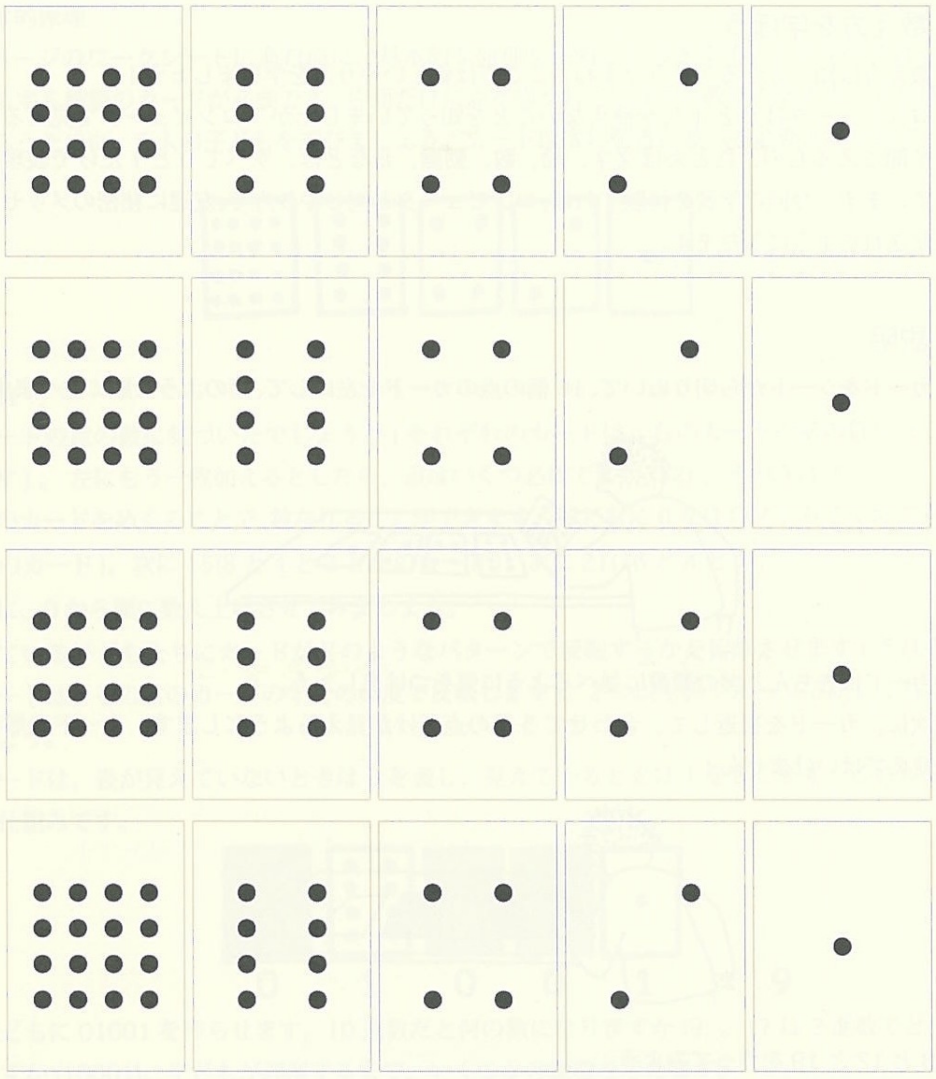
いちばん小さい数といちばん大きい数の間で、表せない数はありますか？

応用問題

1、2、3、4 の数を順に作ってみよう。

数を１ずつ増やしたときに、カードをどのように裏返せばよいかという規則を見つけられますか？

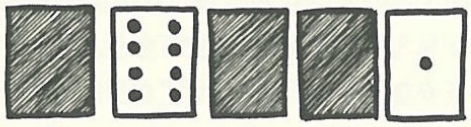
複写用シート



ワークシート

２進数を計算する

２進法では、カードが向いている面を０と１で表します。０はカードが裏向きで、１はカードが表向きで数が見える状態です。例を見ましょう。



0　1　0　0　1　＝ 9

10101 を計算できますか？11111 はどうでしょう？

自分の誕生月を２進数で書いてみましょう。次に、友だちの誕生日を２進数で書いてみましょう。

記号で書かれた数を計算しよう

☒☑☒☒☑ ＝
(☑=1, ☒=0)

⇧⇩⇧ ＝
(⇧=1, ⇩=0)

○○○○○ ＝
(⊙=1, ○=0)

 ＝
(=1, =0)

☹ ＝
(☺=1, ☹=0)

 ＝
(=1, =0)

✚✚✖✚ ＝
(✚=1, ✖=0)

↻↻↻↻↻ ＝
(↻=1, ↺=0)

▲▼▲▼▼ ＝
(▲=1, ▼=0)

♠♠♠♠♠ ＝
(♠=1, ♣=0)

応用問題

長さが 1、2、4、8、16 の棒を使って、31 までの長さの棒を作ってみよう。

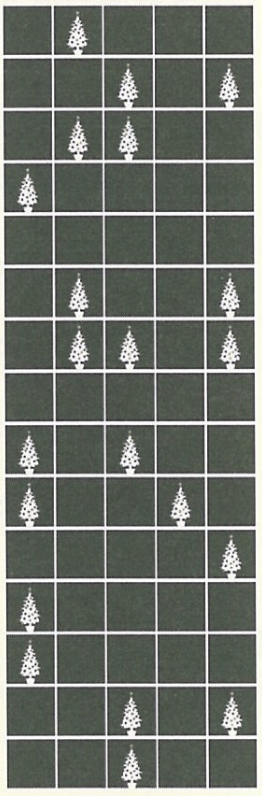
ワークシート

秘密のメッセージを送ろう

トムはデパートの最上階に閉じ込められてしまった。季節はそろそろクリスマスで家に帰りたい。どうしたらいいだろう？

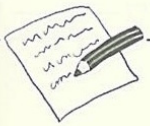
声を出して叫んでみたが、誰も近くにいないようだ。道の向かい側では、夜遅くまでコンピュータの仕事をしている人たちが見えた。彼らに気づいてもらう方法はあるだろうか。トムは使えそうなものがないか探してみた。そして、アイデアが浮かんだ。クリスマスツリーの明かりでメッセージを送ろう！

彼はツリーの明かりを点けたり消したりできるように、コンセントを探した。そして道の向こうの女性が理解してくれそうな、簡単な２進数のコード（符号）を使った。うまく伝わるだろうか？



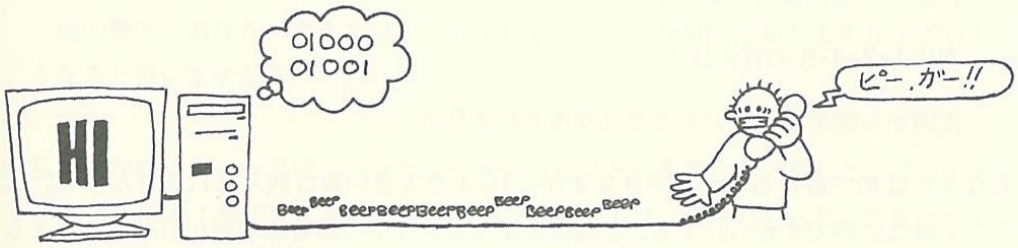
1	2	3	4	5	6	7	8	9	10	11	12	13
a	b	c	d	e	f	g	h	i	j	k	l	m
14	15	16	17	18	19	20	21	22	23	24	25	26
n	o	p	q	r	s	t	u	v	w	x	y	z

ワークシート



電子メールとモデム

コンピュータはモデムでインターネットに接続し、同じように２進法を使ってメッセージを送ることができます。違いは「ビー」という音を使うことだけです。高い音は１を表し、低い音は０を表します。これらの音はとても速く繰り返されるので、私たちに聴こえるのは耳障りな高い音です。もし聴いたことがなければ、インターネットに接続されたモデムの音を聴いてみましょう。または、ファクシミリに電話をかけましょう。ファクシミリも情報を送るためにモデムを使っています。



トムがデパートで使ったコード（符号）を使って、友だちに電子メールのメッセージを送ってみよう。ゆっくりで構いません。本物のモデムのような速さである必要はありませんから。

ワークシート

31 より大きい数を数える

もういちど２進数のカードを見てみましょう。もう１枚カードを加えるとしたら、何個の点のカードが必要でしょう？その次のカードは？新しいカードの点はどんな規則でしょう？これから見るように、大きな数も数枚のカードで表すことができます。

カードの並びをよく見ると、おもしろい規則がわかるでしょう。

1, 2, 4, 8, 16...

1+2+4 の答えは何でしょう？

次に 1+2+4+8 の答えは？

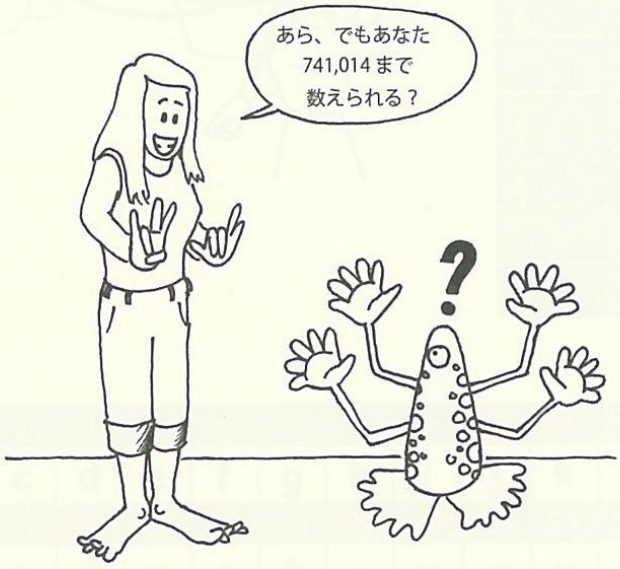
先頭から数を足していくとどうなるでしょう？

みなさんは指で数えることができますが、10 より大きい数は数えられませんよね。エイリアンではないのですから。でも、２進法を使って、片手の指を点の書かれたカードとして使えば、みなさんは０から 31 の数を表すことができます。32 個の数ですね。(0 も数です！)

指を使って順に数えてみましょう。指が伸びていたら 1、閉じていたら０です。

両手を使うと、実は０から 1023 の数を数えられます。1024 個の数ですね！

もし足の指まで使えたら (宇宙人になる必要があるかもしれませんね)、もっと大きな数を数えられるでしょう。片手の指で 32 個を数えられれば、両手では 32 × 32=1024 個を数えられます。さて、足の指を自由に曲げられる女の子は、両手と両足を使っていくつの数まで数えられるでしょう？。



ワークシート

２進数のあれこれ

① ２進数のおもしろい性質を見ましょう。右側に０を加えるとどうなりますか。普段使っている 10 進数では、右に０を加えると、元の数の 10 倍になりました。たとえば、９は 90 になるし、30 は 300 になります。
では、２進数で右に０を加えるとどうなりますか？試してみてください。

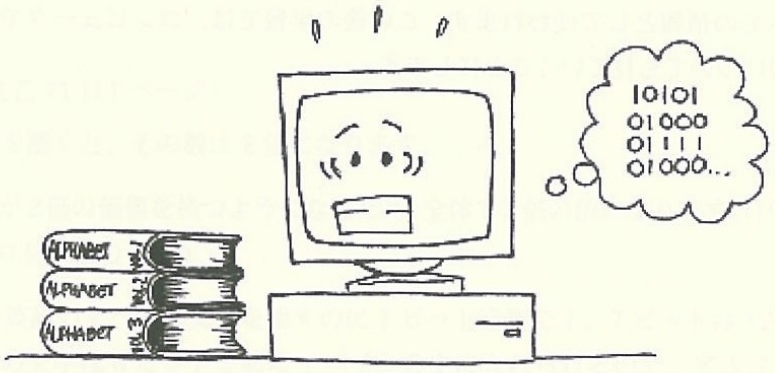
$$1001 \rightarrow 10010$$
$$(9) \qquad (?)$$

他の数で、あなたの考えを試してみましょう。どんな規則がありますか？どうしてそうなると思いますか？

② 私たちが使ったカードは、コンピュータの「ビット」を表していました。ビットというのは２進数の１桁です。英語で使うアルファベットは、５枚のカード、または５ビットがあれば表すことができます。しかし、コンピュータは大文字、小文字、数字、句読点、「$」や「~」のような記号などを区別できる必要があります。

キーボードを見ながら、コンピュータは英語のいくつの文字を表現できる必要があるのかを考えてみましょう。それらの文字を表現するために、コンピュータは何ビットが必要でしょう？

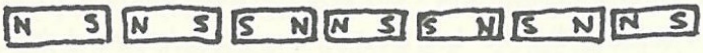
多くのコンピュータは、文字をそれぞれのビットで表現する ASCII と呼ばれる形で表現します。しかし、英語以外の国では、もっと長いコードを使う必要があります。



実際のコンピュータでは

現在のコンピュータは、情報を2進法で扱っています。2個の数字を使うので2進と呼ばれます。人間は普通、10進を使います。1桁の0か1をビットと呼びます。ビットはコンピュータのメモリの中で、トランジスタがオンかオフか、またはコンデンサが充電されているかどうかで表現されます。

データが電話線や電波で送信されるときは、高い音と低い音が1と0として使われます。フロッピーディスクやハードディスクなどの磁気ディスクや磁気テープでは、ビットは表面に塗られた磁性体のNとSの向きで表されます。



音楽CDやCD-ROM、DVDでは、表面で光が反射するかしないかによってビットを表します。1個ずつのビットではたくさんの情報を表現できないため、普通は8個をひとまとまりにして0から255の値を表現します。この8ビットをバイトと呼びます。

コンピュータの速さは、いちどに計算できるビット数で変ります。たとえば、32ビットのコンピュータはいちどに32ビットを計算します。16ビットのコンピュータは32ビットの数をいくつかに分けて計算する必要があるため遅くなります。

つまり、ビットやバイトは、コンピュータが記憶したり通信したりするための数値や文章、そしてすべての情報として使われます。この後の学習では、コンピュータで使われるその他の情報の表現についても見ていくことにします。

解答とヒント

２進数 (5 ページ)

3 は 1 と 2 のカードが必要です。
12 は 8 と 4 のカードが必要です。
19 は 16 と 2 と 1 のカードが必要です。

数を作る方法は 1 通りしかありません。

作ることができるいちばん大きい数は 31 です。いちばん小さい数は 0 です。その間の数はすべて作れます。数を作る方法は 1 通りしかありません。

応用問題
どんな数でも、1 増やすためには、右から順に 1 枚を表にするまで裏返していきます。

２進数で解く (7 ページ)

10101 = 21
11111 = 31

秘密のメッセージを送る (8 ページ)

コード (符号) になったメッセージ : help i(')m trapped

31 より大きい数を数える (10 ページ)

いちばん小さい桁から足していくと、ある桁までの和は、その次の桁の数より 1 だけ小さい数になります。たとえば、１＋２＋４は７で、次の８より１だけ小さいです。

足の指を自由に曲げられる女の子は、1024 × 1024=1,048,576 通り (0 から 1,048,575 まで！) を数えられます。

２進数のあれこれ (11 ページ)

２進数の右に０を置くと、その数は２倍になります。

すべての桁の１が２倍の価値を持つようになるため、全体で２倍の値になります。(10 進数では、右に０を置くと 10 倍になります)

コンピュータは英語のすべての文字を表すのに７ビット必要です。７ビットは 128 文字を表せます。７ビットの文字は８ビットである１バイトの中に入れられるので、アルファベットしか使わない場合は１ビットが無駄になっていることになります。

2.3.2　整列アリゴリズム

次に、整列（ソート）についての部分を紹介します。これは、アルゴリズムの学習と言えばまず最初の大きな目標になるものと言えます。

学習 7 いちばん軽いといちばん重い

整列アルゴリズム

概要

名前を五十音順にしたり、予定や電子メールを日付順にしたり、商品を値段順にするような場合に、コンピュータを使った並びの整列 (ソート) が使われます。順序よく並べておくと、そこから物を探しやすくなりますし、値を見やすくなります。もしクラスのテストの点を順番に並べておくと、いちばん小さい点数といちばん大きい点数が一目でわかります。

正しい方法を使わないと、速いコンピュータでも大きい並びを整列するときに時間がかかってしまいます。幸いなことに、高速な整列手法がいくつか知られています。この学習で、子どもは何種類かの整列手法を知り、どのように速く処理できるか良い方法を学びます。

教科学習との関連

・Mathematics: Measurement Level 2 and up. Carrying out practical weighing tasks.

※［量と測定］小３で重さを扱います。

技能

・Using balance scales　天秤を使う。
・Ordering　順番に並べる。
・Comparing　比較する。

年齢

・８歳以上

教材

＜グループごとに必要なもの＞

・大きさは同じだが重さが違う８個の入れ物
　(牛乳パックやフィルムケースに砂を入れたものなど)
・天秤ばかり
・ワークシート（66、67 ページ）

いちばん軽いといちばん重い

話し合おう

コンピュータを使った整列はよく使われます。どんな場面で重要になるかを話し合いましょう。もし整列されていなかったら、どんなことが起きるでしょう？

コンピュータは普通、値を一度に２個ずつ比較します。次のページの学習ではこの制約を使い、子どもに何と似ているかを伝えます。

学習

① 子どもをグループに分けます。

② グループごとに、66 ページの学習シートと、重りとはかりが必要です。

③ 学習の後、結果を話し合わせます。

ワークシート

重りの整列

ねらい

重さがわからないものを重さの順に並べるよい方法を見つける

必要なもの

砂か水、8個の区別できない入れ物、天びん

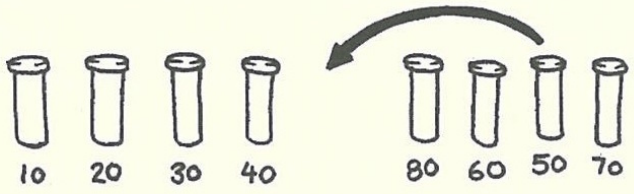
やること

① 入れ物に砂か水を違った量だけ入れ、こぼれないように封をします。
② 順番を入れ替えて、重さの順番がわからないようにします。
③ いちばん軽いものを見つけなさい。簡単に見つけるにはどうしますか？
　注意：入れ物の重さははかりで調べます。いちどに2個の重さを比べられます。
④ 適当に3個の重りを選び、はかりを使って、いちばん軽いものからいちばん重いものまで順に並べなさい。どうやってやりましたか？比べる回数を少なくできたのはどんなやり方でしたか？それはなぜですか？
⑤ 次に、重り全体を、いちばん軽いものからいちばん重いものまで順に並べなさい。

整列が終ったら、並んでいる隣同士をもう一度比べて、正しい順に並んでいることを確かめなさい。

選択ソート

選択ソートはコンピュータが使う方法の1つです。どのように動くのかを見てみましょう。最初に、いちばん軽い重りを見つけて、別の場所に移します。これを、重りがなくなるまで繰り返します。



自分が何回重さを比べたかを数えておきましょう。

応用問題

8個を整列するために、何回比較する必要があるかを計算しましょう。
9個ならどうですか？20個なら？

ワークシート

分けて作業する

クイックソート

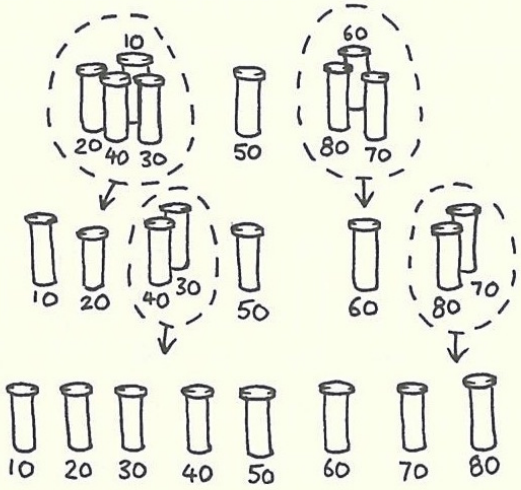
クイックソートは、最も速い手法の１つで、特に並びが大きいときは選択ソートよりずっと高速です。どのように動くかを見てみましょう。

① 適当に１個の重りを選び、天びんの片方に乗せます。

② 次に、残りの重りと比べます。その重りより軽いものは左に、同じものは真ん中に、重いものは右に置きましょう。
（どちらかの数が多くなってしまうことがあるかもしれません）

③ それぞれの組の中で、同じことをやりましょう。
選んだ１個を必ず真ん中に置くのを忘れずに。

これを、それぞれの組の中が１個になるまで繰り返します。最後にすべての組が１個ずつに分けられたら、小さいものから順に並んでいるはずです。



大きさを何回比べる必要がありましたか？

いちばん小さい値やいちばん大きい値を偶然選んでしまわない限り、クイックソートは選択ソートより効率がよいことがわかると思います。選択ソートの比較回数は 28 回ですが、クイックソートでは、運よく真ん中の値を選べたときは 14 回で済みます。運が悪くても選択ソートと同じくらいで、ほとんどの場合は選択ソートよりずっと速いです！

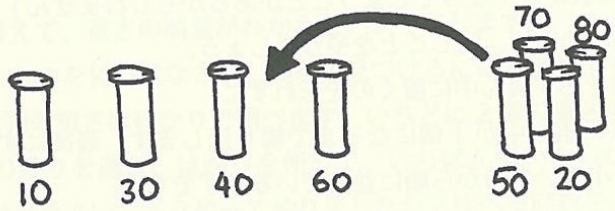
応用問題

常にいちばん小さい値を選んでしまったときは、クイックソートでは何回の比較が必要でしょう？

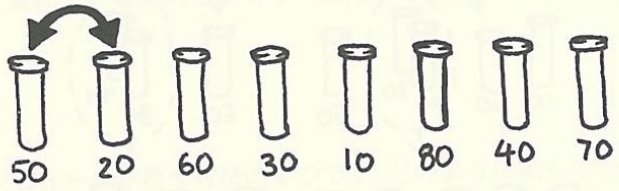
発展と応用

いろいろなソートの方法が発明されています。いくつかを試してみましょう。

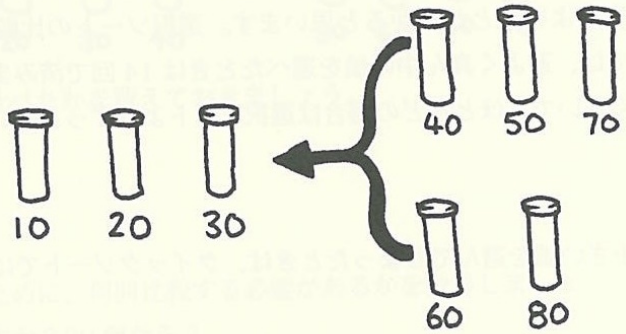
挿入ソートでは、図のように1個ずつ正しい並びの位置に入れていきます。まだ並べられていないものは減っていき、並べられたものが増えていきます。トランプをする人は、よくこの方法でカードを並べます。



バブルソートでは、並びを何度も行き来して、図のように順番が逆のものを交換します。作業は並び全体で交換するものがなくなったときに終わります。この方法はそれほど効率がよくありませんが、理解しやすいと思う人がいるかもしれません。



マージソートは「分けて作業する」方法のひとつです。まず、全体を同じ数になるように2つに分けます(奇数の場合にはどちらかが1個多くなります)。そして、それぞれを整列した後で、2つの並びを統合します。2つの整列された並びを統合するには、並びの先頭を比べて小さいほうを取り出せばよいので簡単です。図では40gと60gが並びの先頭にあるので、40gのほうを取り出して新しい並びに加えています。



実際のコンピュータでは

順に並んでいると情報を探すのが簡単になります。もし電話帳や辞書、本の索引が読みの順に並んでいなかったら、生活はかなり不便になるでしょう。支出額のような数字の並びが整列されていれば、極端な値は最初か最後にあるので見つけるのが簡単になります。同じ値は並ぶので、重複を調べることも簡単です。

コンピュータはたくさんのデータを整列する必要があるため、コンピュータ科学者は効率のよい整列方法を見つけようとしてきました。挿入ソート、選択ソート、バブルソートなどの遅い方法が役立つこともありますが、普通はクイックソートのような高速なものが使われます。

クイックソートでは再帰の考え方が使われています。この考え方では、並びを小さな並びに分けて、それぞれに対して同じように作業を進めます。この方法は「分けて作業する (分割して統治せよ)」と呼ばれます。並びは、作業するために適した大きさになるまで、繰り返し分割されます。クイックソートでは、それぞれの並びは要素が 1 個になるまで分解されます！この考え方は複雑に見えますが、他の方法よりずっと高速です。

70　第2部　コンピュータを働かせる：アルゴリズム

解答とヒント

① いちばん小さな値を見つける最もよい方法は、すべての要素を見て、それまでのいちばん小さな値を覚えておくことです。そのために、2つの値を比較して、小さなほうの値を覚えます。次に、もう1つと比較します。そしてすべての要素を調べます。

② 天秤で3個の重さを比べましょう。3回の比較が必要ですが、子どもが推移則に気づいたときは2回で済むことがあります。(AがBより小さくBがCより小さいなら、AはCより小さいことがわかります)

上級者向け

選択ソートで比較する回数を楽に計算する方法を紹介します。

2個の要素から小さいほうを見つけるために、1回の比較が必要です。3個では2回、4個では3回が必要です。8個の要素を考えると、最初の1個を見つけるためにに7回の比較を行い、次の1個のために6回の比較を行い、次に5回の比較、というように、次の比較が必要です。

$$7 + 6 + 5 + 4 + 3 + 2 + 1 = 28$$ 回の比較。

n個の要素を整列するためには、$1 + 2 + 3 + 4 + \ldots + n - 1$ 回の比較が必要です。これらの和は、順序を変えることで簡単に計算できます。

たとえば $1 + 2 + 3 + \ldots + 20$ の和は、次のように数の組み合わせを作れます。

$$(1 + 20) + (2 + 19) + (3 + 18) + (4 + 17) + (5 + 16) +$$
$$(6 + 15) + (7 + 14) + (8 + 13) + (9 + 12) + (10 + 11)$$
$$= 21 \times 10$$
$$= 210$$

一般に、和は次のように計算できます。$1 + 2 + 3 + 4 \ldots + n - 1 = n(n - 1)/2$

2.4　いろいろなプログラミング言語

現在多くのプログラミング言語があります。教育向けのものを多めに、主なものとその特徴を掲げてみます。

分類	言語	特徴
本格派	C/C++	現在主流の開発言語。「事例」にある。
	Java	プラットフォーム非依存（?）オブジェクト指向言語。「事例」にある。
スクリプト言語	Perl	スクリプト言語のかつての主流。
	Python	現在の主流。Perl の影響が大きい。
	Ruby	現在の対抗。Perl の影響が大きい。
	Javascript	ブラウザで利用される。Google マップもこれ。
オブジェクト指向	Smalltalk	最初期のオブジェクト指向言語。
	Squeak	Smalltalk をベースとする。ビジュアルな開発環境もある。
	Scratch	Squeak をベースとする。とりあえず猫を動かす。
	プログラミン	Scratch をベースとする。教育用。文部科学省謹製。
	Blockly	Google 謹製。Scratch にそっくりだが用途は違う。
その他教育用	LOGO	教育用。タートルグラフィックス。
	Viscuit	教育用ビジュアル言語。「事例」にある。
	ドリトル	教育用。日本語でプログラム。「事例」にある。
	MindStorm	LEGO 社。
化石	Fortran	数値計算のかつての主流。
	Pascal	厳格な C。元は教育用。Macintosh の初期の開発言語でもあった。
	Basic	VisualBasic として生存している気もする。
	PHP	ウェブサイトの作成用。
	LISP	最初期の言語。当初は理論上の言語であったがすぐに処理系が実装される。究極的に簡単な構文と強力なマクロシステムを備え、オブジェクト指向拡張も行われている。大規模なシステムであっても、実行中に一部を変更可能であるなど、柔軟性は特筆に値する。現在でもトップクラスの機能と実行性能を誇る言語である。

2.4.1 Scratch

起動すると表示される猫のキャラクターが印象的で、子供にも親しみやすくできています。ブロックを組み合わせてプログラムを作成するという特徴があり、ほぼマウスのみでプログラムを作成できます。後に詳しく説明しますので、ここまでに留めます。

図 14: Scratch

2.4.2 プログラミン

Scratch をベースに文部科学省が開発した言語で、やはりブロックの組み合わせでプログラムを作成します。Scratch を習得していればすんなりと利用できます。ただ、Scratch で十分扱いやすいため、プログラミンが日本の教育現場で主流にな

るかどうかは不透明です。

図 15: 「プログラミン」ホームページ

プログラミンの実行には Flash が必要です。プログラミンのサイトには、いくつかの「おてほんプログラム」があり、すぐに試してみることができます。

図 16: おてほんプログラム

図 17: プログラム作成画面

図 18: ブロックでプログラム作成

2.4.3 Blockly

Google が開発した言語で、Scratch を参考にしています。プログラミンよりも Scratch に似ていて、ブロックでプログラムを作成します。ブロックで作成したプログラムを文字で作成された Javascript のプログラムに変換できるという特徴があります。

Scratch との一番の違いは、学習者が Blockly で直接プログラムを作成するのではなく、学習者がプログラムを作成する環境を作るための「教育向けビジュアルプログラミングライブラリ」である所です。

教えたいことに特化した教材を作成できるため、学習者が言語について知っておくべき知識を相当減らすことができます。もちろん、その分教材作成の手間は増えます。

図 19: Blockly ホームページ

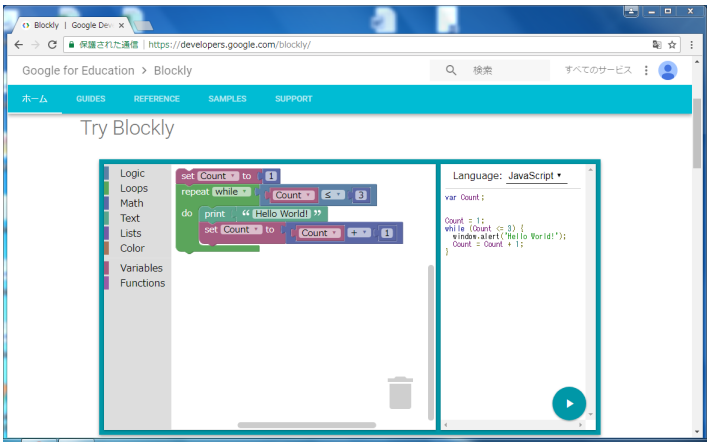


図 20: Blockly のサンプルたち

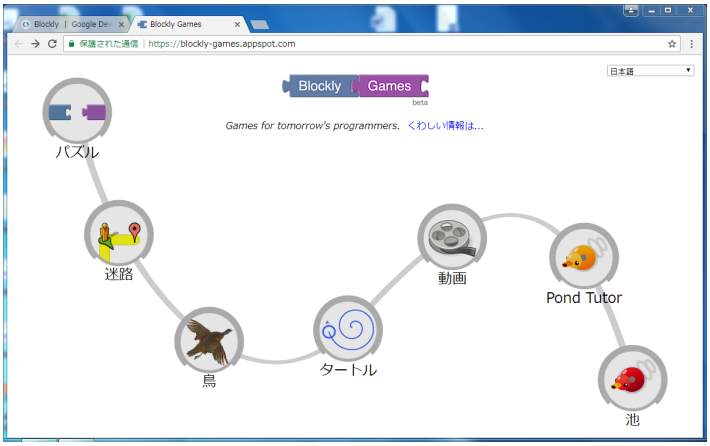
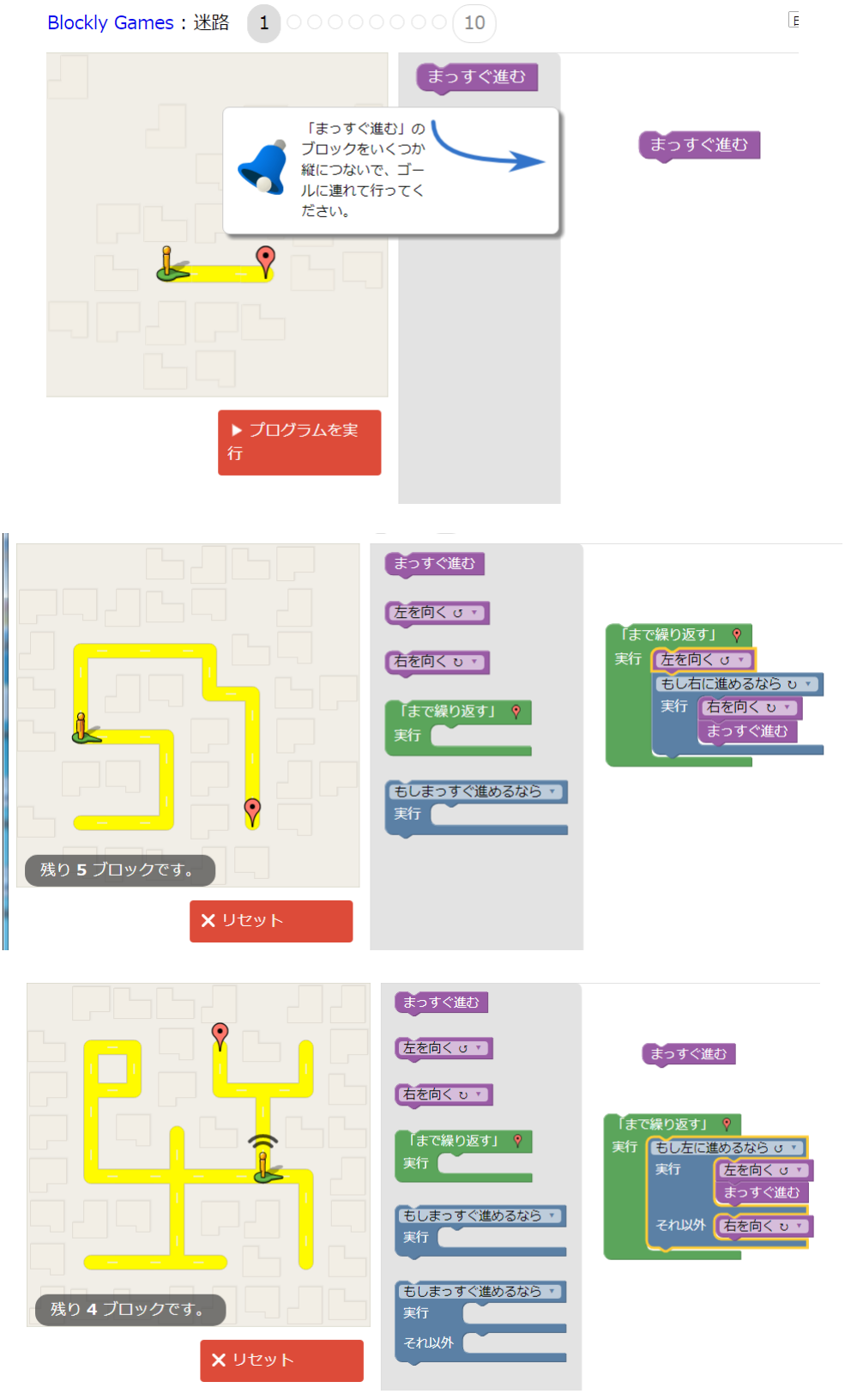


図 21: 迷路


Blockly Games：迷路
1
10
まっすぐ進む
「まっすぐ進む」のブロックをいくつか縦につないで、ゴールに連れて行ってください。
まっすぐ進む
▶ プログラムを実行
まっすぐ進む
左を向く
右を向く
「まで繰り返す」
実行
もしまっすぐ進めるなら
実行
「まで繰り返す」
実行
左を向く
もし右に進めるなら
実行
右を向く
まっすぐ進む
残り 5 ブロックです。
リセット
まっすぐ進む
左を向く
右を向く
「まで繰り返す」
実行
もしまっすぐ進めるなら
実行
もしまっすぐ進めるなら
実行
それ以外
まっすぐ進む
「まで繰り返す」
実行
もし左に進めるなら
実行
左を向く
まっすぐ進む
それ以外
右を向く
残り 4 ブロックです。
リセット

実際に迷路の例を見てみますが、ブロックでのプログラム作成の予備知識がまったくなくても、迷路脱出の方法を考えることに集中できるのがわかると思います。

図 22: 多角形

3　スクラッチ入門

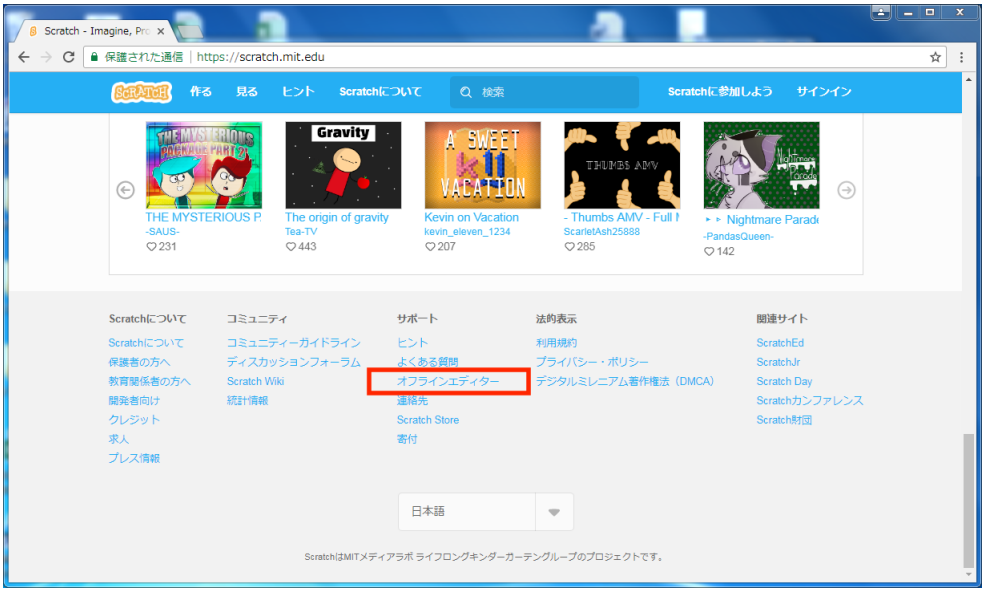
3.1　インストール

Scratch の最新バージョンは 2.0 です。ブラウザ上でも使用できますが、プログラムを保存するためには、アカウントを作成する必要があります。また、Flash が必要です。

ここではアカウントの必要のない「オフラインエディター」のインストール方法を記します。

まず、Scratch ホームページ (`https://scratch.mit.edu`) 下部の「オフラインエディター」をクリックします。

図 23: ホームページ下部



すると「Adobe AIR」のインストールが促されます。「Adobe AIR」がないとオフラインエディターは動作しません。「Adobe AIR」のインストールが済んだら、オフラインエディターをインストールします。

図 24: インストールのページ

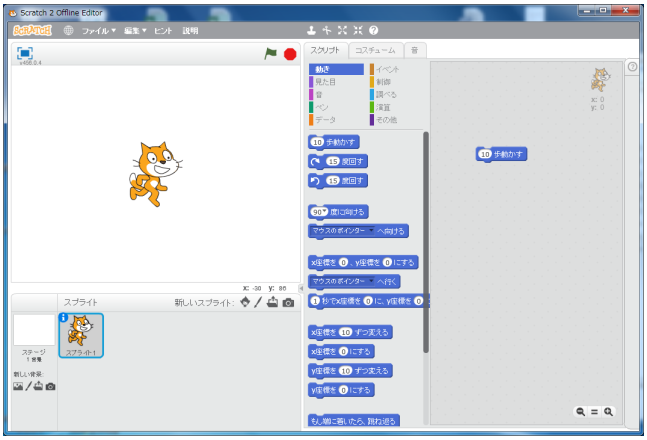
本講習では、使用するコンピュータに Adobe AIR をインストールすることが困難なため、古いバージョンの Scratch 1.4 を使用します。そのため画面構成などが異なります。本資料では新しい Scratch 2.0 の画面写真を掲載します。

同様の理由で Scratch 1.4 をインストールする必要がある場合は、上の画面の下部に、「Scratch のその他のバージョン」という項目があるので、そこから Scratch 1.4 のページへ辿れます。

3.2　プログラム作成の基本

起動すると下のような画面が現れます。

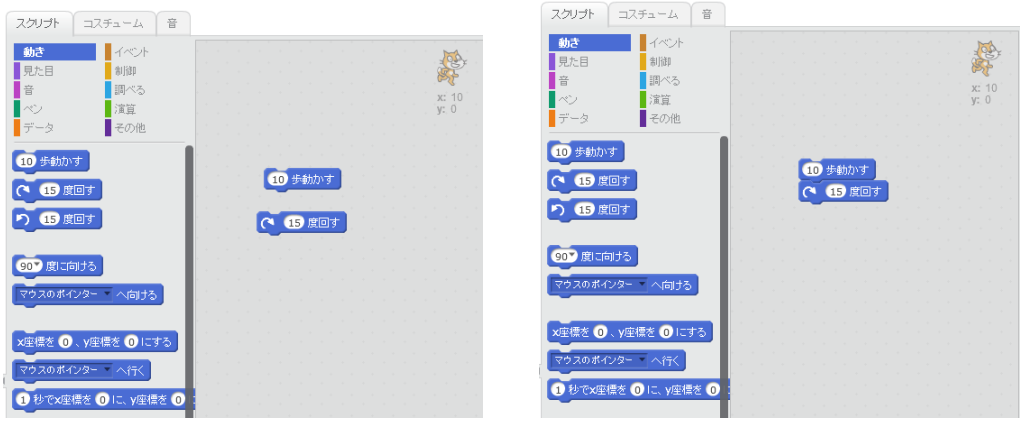
図 25: Scratch の初期画面./img/scratch24.png



猫のキャラクターのような動かせる対象（オブジェクト）は「スプライト」と呼ばれます。ドラッグすると動かすことができます。また、画面中央のブロックが並んでいる場所から、ブロックを画面右の領域に配置し、クリックすると実行できます。こうしてブロックを置いて組み合わせていくのが、Scratch でのプログラミングです。

下図左のように 2 つ置けば、クリックした方が実行されます。下図右のように 2 つのブロックを連結すると、上側のブロックをクリックすれば上から順番に実行されます。こうして、簡単な命令を組み合わせて、複雑な動作をさせるのが、Scratch におけるプログラミングの基本です。

図 26: 2 つ置いて、連結する



不要になったブロックは、元のブロック置き場にドラッグするとなくなります。ブロックを右クリックすると、削除や複製もできます。また、作成したプログラ

ムがなくなってしまった思ったときは、

(1) **画面左下のスプライトの猫が選択されているか、**

(2) **画面中央上のタブでスクリプトが選択されているか**

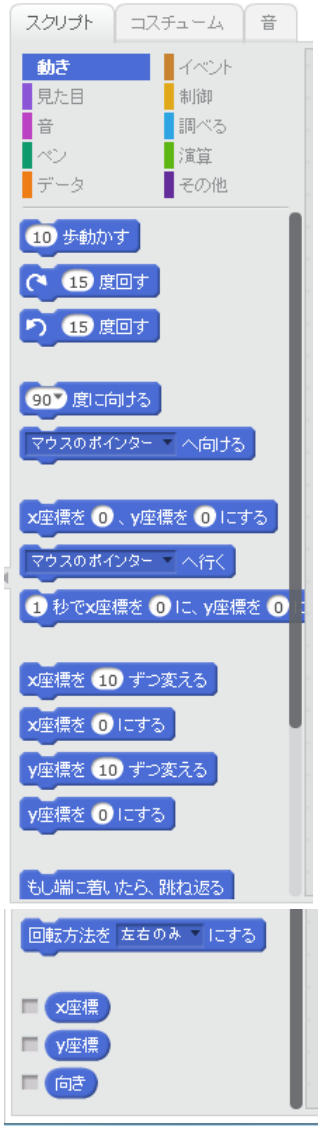
をまずは確認して下さい。

3.3 「スクリプト」タブ

このタブにはプログラムに使ういろいろなブロックが機能によって分類して置いてあります。分類ごとに主なブロックの説明をします。

3.3.1 「動き」ブロック

図 27: 動き

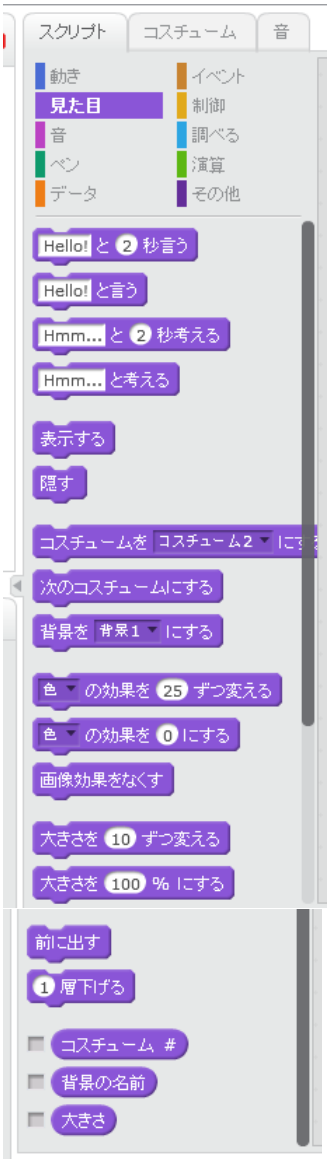


- 「10 歩動かす」、「15 度回す」は、猫が現在向いている方向に歩いたり、向きを変えたりします。値はクリックすると変更できます。
- 「90 度に向ける」は向きを直接指定します。時計のように真上を 0 度として右回りに測ります。
- 「x 座標を 0、y 座標を 0 にする」は猫の位置を直接指定します。座標は猫のいる領域の右下に表示されています。

- 猫の座標や向きは、下の方の「x 座標」、「y 座標」、「向き」という変数（詳細は後述）としても用意されています。
- 「x 座標を 10 ずつ変える」、「x 座標を 10 にする」なども猫の位置を指定しますが、前者は現在位置からどれだけ移動するかを意味し（相対移動）後者は変更後の座標を直接指定することを意味します（絶対移動）。この言葉遣いは、他のブロックでも同じ規則です。
- 「もし端に着いたら、跳ね返る」は、移動中に領域境界に到達したら、反射するように向きを変えます。これで、領域外に出ないように運動させることができます。

3.3.2 「見た目」ブロック

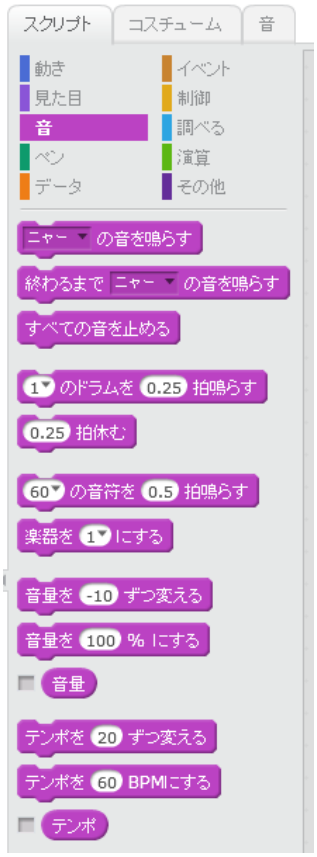
図 28: 見た目



- 「Hello!と言う」などは、猫が吹き出しに文章を表示します。
- 「次のコスチュームにする」などは、少し違う表示の画像に変えます。これを繰り返すことで歩く様子などをアニメーションにすることができます。
- 「色の効果を 25 ずつ変える」などは、スプライトの見た目をいろいろ変えます。「色」の部分をクリックすると、さまざまな画像効果に変更できます。
- 「大きさを 10 ずつ変える」、「大きさを 100%にする」は、それぞれ、大きさを相対的、絶対的に変更します。
- 「前に出す」、「1 層下げる」は、複数のスプライトがある場合の重なり方を変更します。

3.3.3 「音」ブロック

図 29: 音

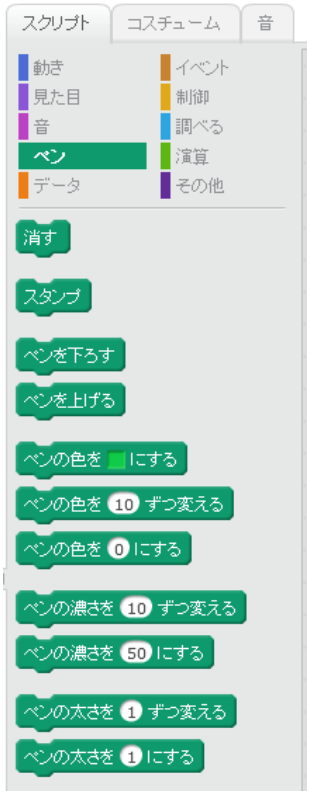


●猫が鳴き声を出したり、音程を指定して音を鳴らすことができます。

3.3.4 「ペン」ブロック

Scratch で図形を描くには、猫がペンを持っていると考え、ペンを下ろして、移動して、ペンを上げる、という手順で描きます。

図 30: ペン



- 「消す」で描いたものが消せます。
- 「スタンプ」は今いる位置にスプライトと同じ画像を描きます。これは、猫が移動してもそのまま残ります
- 「ペンを下ろす」、「ペンを上げる」で、ペンを上げ下げします。ペンを下ろした状態でのみ線が描かれます。
- 「ペンの色を（黒）にする」などで、ペンの状態を変更できます。

3.3.5 「データ」ブロック

これは Scratch 1.4 では「変数」という名前でした。

図 31: データ

●変数やリストを作成します。作成のボタンをクリックすると、下のような画面になり、変数やリストが作成できます。変数自体は既に「動き」ブロックなどで登場していますが、自分でも変数やリストをを作成できます。

図 32: 変数、リストの作成

詳細は後述します。

3.3.6 「イベント」ブロック

イベントとは、主に、マウスのクリックや移動、キーボードからの入力のような、コンピュータ外部からの入力に起因する事象を指します。それらが発生したときに、何かの動作をさせたいときには、「イベント」ブロックを用います。

ここでは、他に、複数のスプライト間の通信として使われる「メッセージ」のやりとりもイベントとして扱っています。

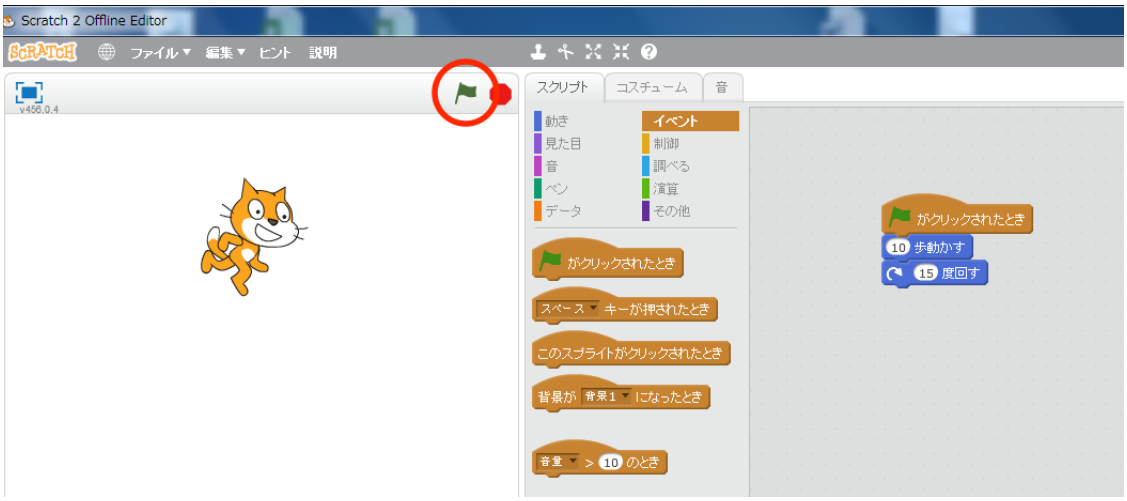
図 33: イベント

- 「(旗) がクリックされたとき」は、少し下の図の丸で囲んだ「旗」がクリックされたときに実行されます。この「旗」は、このスプライトに対するメインのプログラムという意味合いがあります。
また、「旗」の右隣にある赤い印は、クリックすると実行中のプログラムがすべて停止します。

- すぐ下の 2 つの図のように、一連のプログラムの上に、「イベント」ブロックを連結して用います。

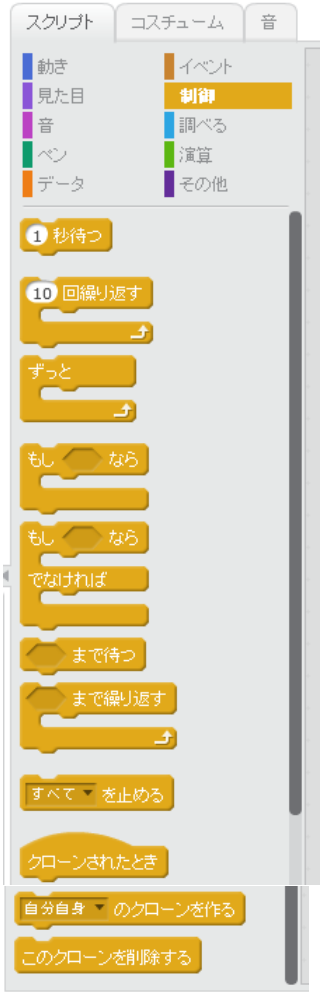
図 34: 「イベント」ブロック

図 35: 旗



3.3.7 「制御」ブロック

図 36: 制御

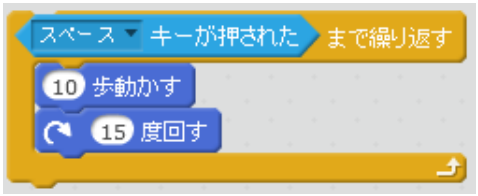


- 「10 回繰り返す」や「ずっと」は、コの字型の内側にブロックを配置すると、その部分を指定回数、あるいは、ずっと繰り返します。回数には変数や後述の数式も置くことができます。

図 37: 反復（回数指定）のブロック

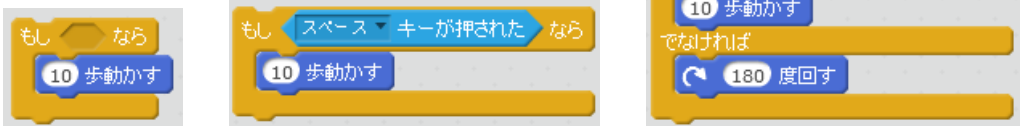
- 「（条件）まで繰り返す」のブロックは条件が成立するまで内部のブロックを反復して実行します。条件には後述の条件式を置きます。

図 38: 反復（条件指定）のブロック



- 「もし」で始まるブロックは、条件が成立した場合に内部のブロックを実行する（そして不成立の場合は何もしない）ものと、条件の成立、不成立に応じて異なるブロックを実行するものがあります。

図 39: 条件分岐のブロック

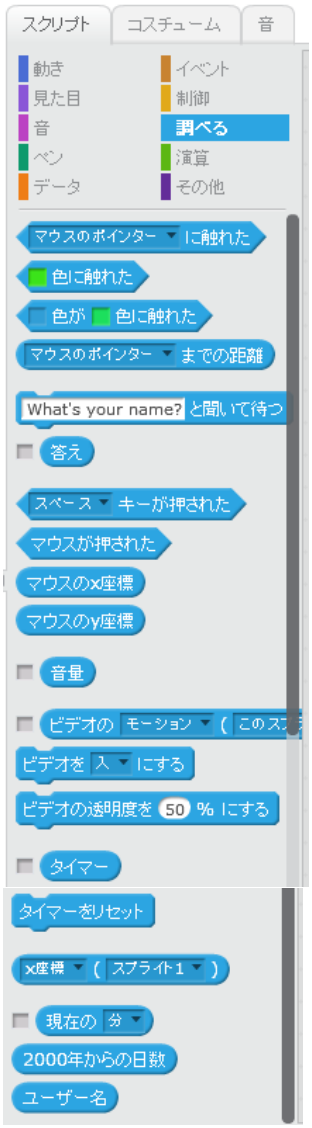


3.3.8 「調べる」ブロック

「調べる」ブロックを用いると、各種の情報を数値や文字列で取得し、変数のように式の一部として使うことができます。

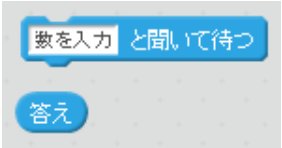
変数のように式の一部として使えるものは左右が丸いブロック、条件式は左右が三角になっているブロックです。

図 40: 調べる



- 「マウスのポインターに触れた」のように下向き三角形が付いている場合は、クリックすると内容を変更できます。
- 「色が色に触れた」の条件を使うと、背景に描いた模様に触れたときに何かを実行させたりすることができます。色を変更するには、色の部分をクリックしてから、目的の色の所をクリックします。
- キーボードから数値や文字列を入力したいときは、「What's your name? と聞いて待つ」ブロックと「答え」ブロックを組で使います。「What's your name? と聞いて待つ」ブロックを実行すると、人間が入力した後に「答え」ブロックに入力結果が設定されます。

図 41: キーボードからの入力



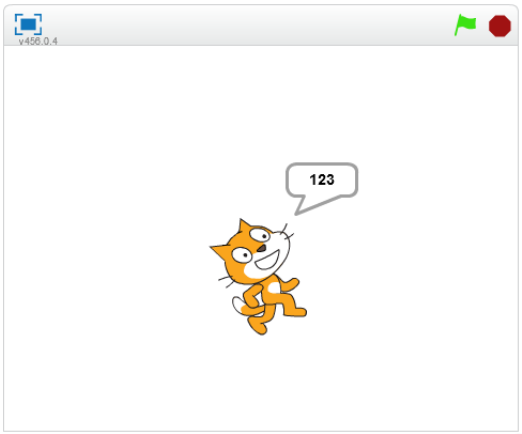
キーボードからの入力の例を見てみます。下のプログラムを実行すると以下のように処理が進みます。

図 42: キーボードからの入力を用いたプログラム例

実行すると入力を促されます

123 と入力してみます

「答え」が 123 になり、猫が 123 と言います



3.3.9 「演算」ブロック

変数どうしで計算したり、反復や条件分岐のブロックで条件式として使えるブロックです。

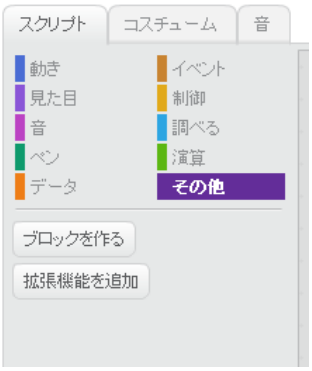
図 43: 演算

- 四則演算、剰余、平方根、乱数といった式があります。
- 不等式や相等、「かつ」などの論理演算があります。
- 「hello と world」ブロックは、文字列の連結を行います。

3.3.10 「その他」ブロック

一連のブロックからなるプログラムに名前を付けて、その名前で呼び出す仕組みがあります。他の言語では「関数」とか「メソッド」と呼ばれるものです。

図 44: その他



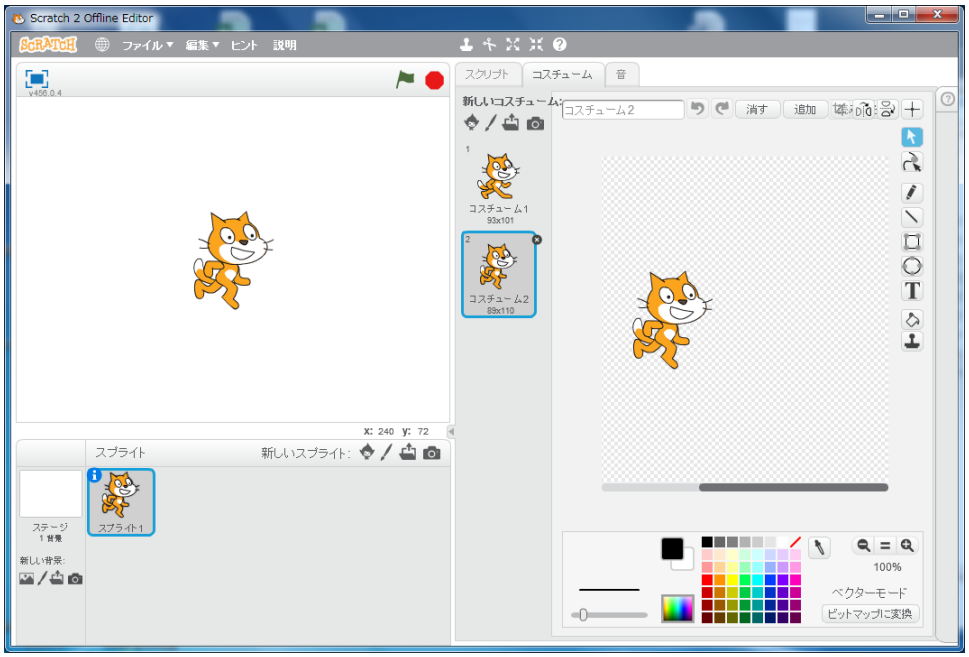
- 「ブロックを作る」ボタンを押すと、プログラムの領域には「定義」ブロックができます。これを、一連のブロックの上に連結すると、その一連のブロックに名前が付き、その名前で一連のブロックを呼び出せるようになります。

図 45: 「定義」を用いたプログラム例

3.4 「コスチューム」タブ

コスチュームとは、1つのスプライトに少し違う見た目を与えるものです。それを短時間で切り替えると、例えば歩いているようなアニメーションのような効果が得られます。もともと用意されていることもありますし、自分で作ることもできます。

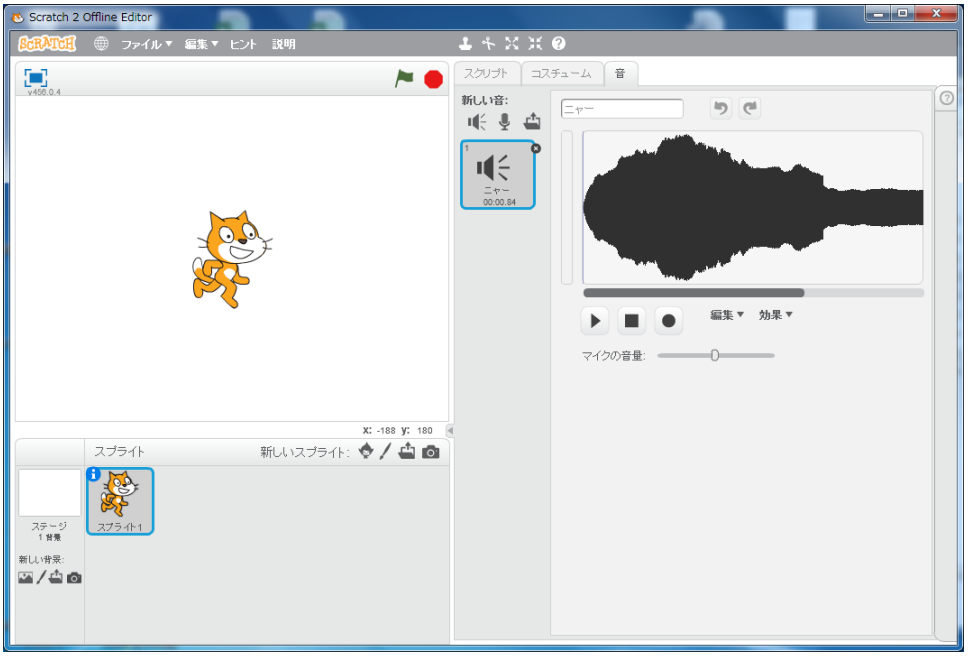
図 46: コスチューム



3.5 「音」タブ

Scratch ではいろいろな音を鳴らせます。詳細は省略します。

図 47: 「音」タブ



4 続・スクラッチ

より進んだプログラムの作成について説明します。

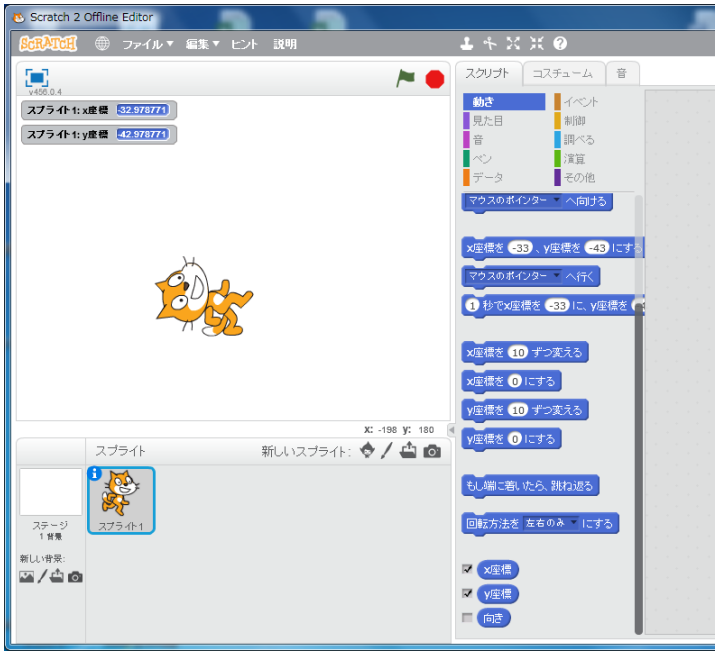
4.1 変数

4.1.1 変数とは

変数とは数値や文字列といった値の入れ物で、左右が丸くなっているブロックで表されます。キーボードからの入力などで、既にいくつかは説明しました。

下の図「x 座標」などは、Scratch にもともと用意されている変数です。変数の左のチェックボックスで、猫のいる領域に変数の値を表示させることもできます。

図 48: 変数



4.1.2 変数の作成

自分で新しい変数を作成することもできます。「変数を作る」ボタンを押して変数名を指定して作成します。すると、変数を操作するいくつかのブロックも用意されます。

図 49: 変数の作成

4.1.3　変数の利用

自分で作成した変数を用いて、プログラムを作成してみます。下のプログラムは、正の数を 5 回入力したときの最大値を求めるプログラムです。

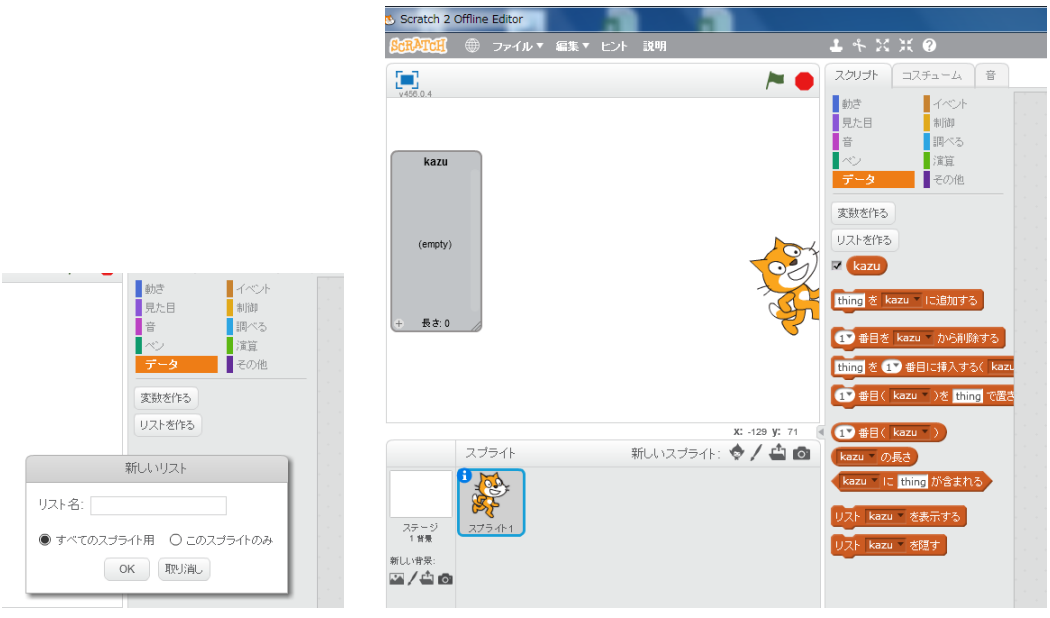
図 50: 変数の利用

4.2　リスト

4.2.1　リストの作成

「リスト」とは、番号付けられた一連の変数の集まりです。他の言語では配列と呼ばれることもあります。「データ」ブロックの一覧で、「リストを作る」ボタンを押してリストの名前を指定すると、リストが作成されます。リストを操作するためのブロックも現れ、リストの内容の一覧が、猫のいる領域に表示されます（まだリストは空です）。

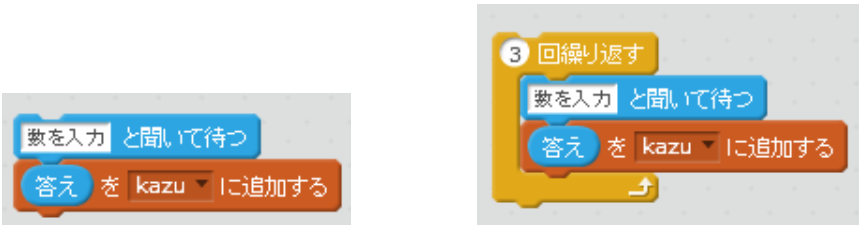
図 51: リストの作成



4.2.2　リストの要素のアクセス・追加・削除

リストに値を格納するにはいくつかの方法がありますが、下図での方法は、「(答え) を (kazu) に追加する」ブロックを用いる方法です。キーボードからの入力をリストに登録しています。右のものは 3 回繰り返すようにしています。

図 52: リストへの要素の追加



4.2.3　リストの利用（最大値）

いくつかの値が格納されたリストが与えられたとき、その最大値を求めるプログラムを考えてみます。

まず最大値を格納する変数を作成します。

図 53: 最大値を格納する変数の作成

まず、リストの 1 番目を暫定的に最大値とします。リストの 2 番目の方がより大きければ、2 番目を最大値とします。

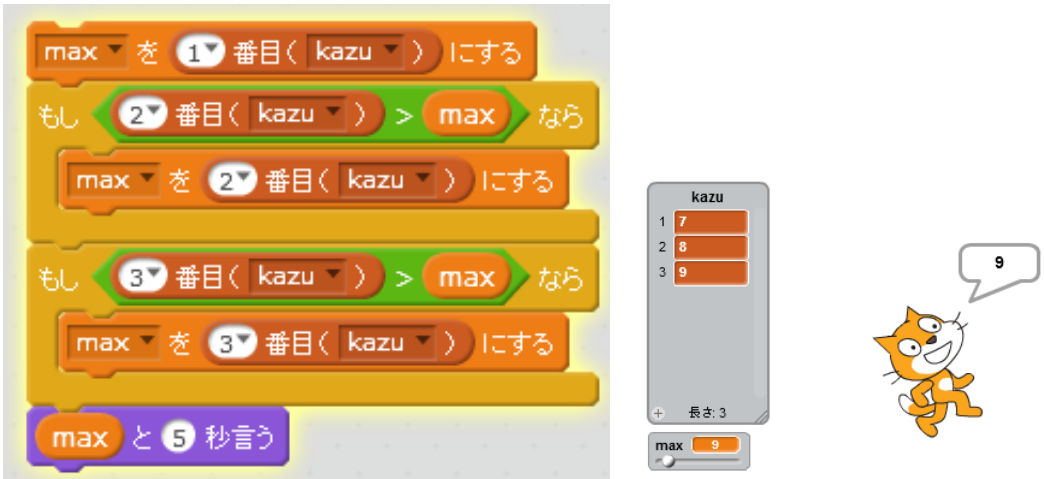
図 54: 2 番目との比較

同様にしてリストの 3 番目の方がより大きければ、3 番目を最大値とします。

図 55: 3 番目との比較

最後に猫に言わせます。

図 56: 実行結果



しかし、これでは 3 要素のリストしか対応できませんし、要素が多くなるとプログラムがどんどん長くなります。

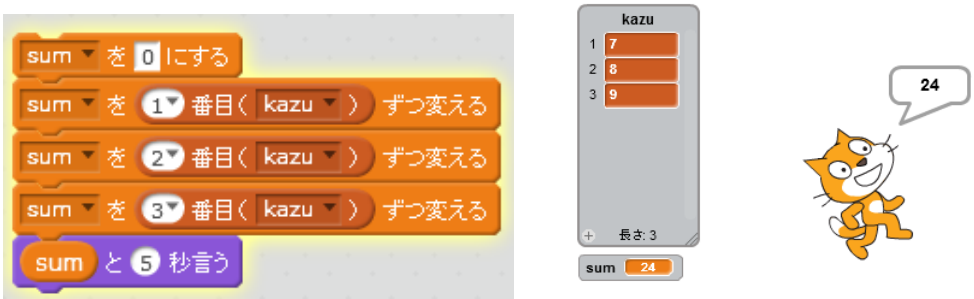
4.2.4　リストの利用（合計）

リストの要素の合計を求めてみます。まず、合計を格納する変数を作成します。

図 57: 使用する変数とリスト

まず、合計を 0 にしてから、3 つの要素を順に加えて、猫に話してもらいます。

図 58: プログラムと実行結果



しかしこれでは先ほどの最大値の場合と同じで、3 要素の場合にしか対応できていません。そこで、今度は、要素の数の依存しないような工夫をして、リストの値の合計を求めるプログラムを考えてみます。

要素の番号を数える変数も新たに作成します。

図 59: 要素の番号を数える変数

先ほどは、可算のためのブロックを 3 回置いていましたので、それを反復のブロックを用いて表してみると、下のようになります。

図 60: 反復を利用したプログラムと実行結果

4.3 基礎的なプログラム

4.3.1 探索

リストの要素に、探している値があるかどうか探索するプログラムを作ってみます。実際には探索するためのブロックが Scratch には用意されていますが、練習のため自分で作ってみます。

まず、リストとリストの何番目かを数える変数を作成します。

図 61: リストと変数の作成

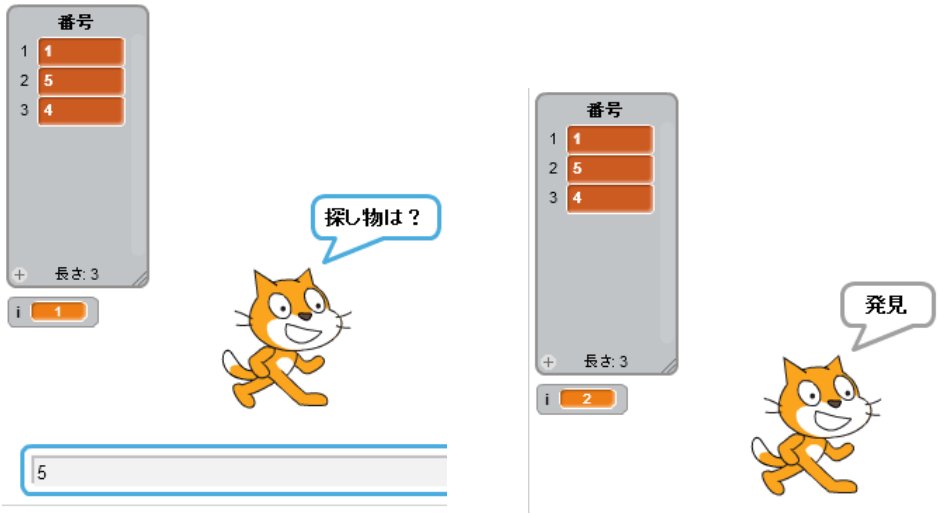
次の方針でプログラムを作成します。

(1) 探す値をキーボードから入力します。
(2) 何番目かを数える変数は最初 1 にして、以降 1 ずつ増やします。
(3) 比較して一致すれば、猫が「発見」と言って終了です。
(4) 何番目かを表す変数がリストの長さを超えてしまったら、猫が「なかった」と言って終了です。

図 62: 探索プログラム

リストに 1、5、4 とあるとき、5 を探してみると、発見されます。

図 63: 実行結果



4.3.2 ユークリッドの互除法

ユークリッドの互除法で 2 つの数の最大公約数を求めるプログラムを作成してみます。a と b の最大公約数を求めるユークリッドの互除法は次のようなアルゴリズムです。ただし、a と b は自然数で、の少なくとも一方は 0 ではないとします。

(1) b が 0 ならば最大公約数は a です。
(2) b が 0 でないなら、a を b で割った余りを t として、a と b の代わりに、それぞれ、b と t を考えて (1) に戻ります。

だんだん数が小さくなるので、このアルゴリズムはいずれ停止することに注意しておきます。下のプログラムでは、(2) の手順が直接できないため、補助的に変数 t を利用しています。

図 64: ユークリッドの互除法

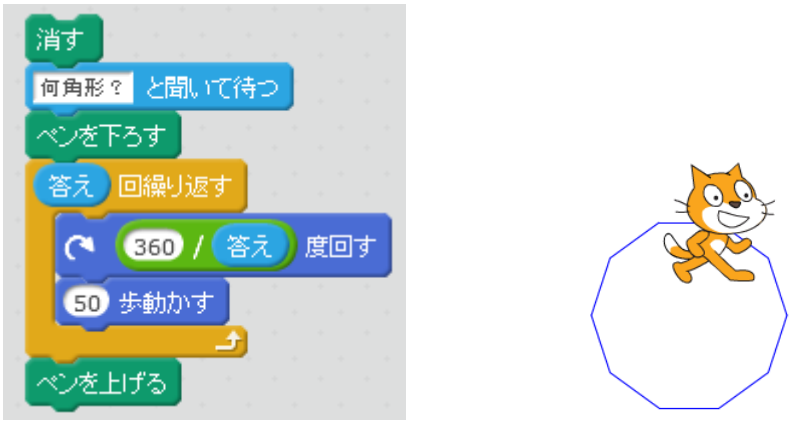
実行結果です。何を入力したのでしょう。

図 65: 実行結果

4.3.3 多角形

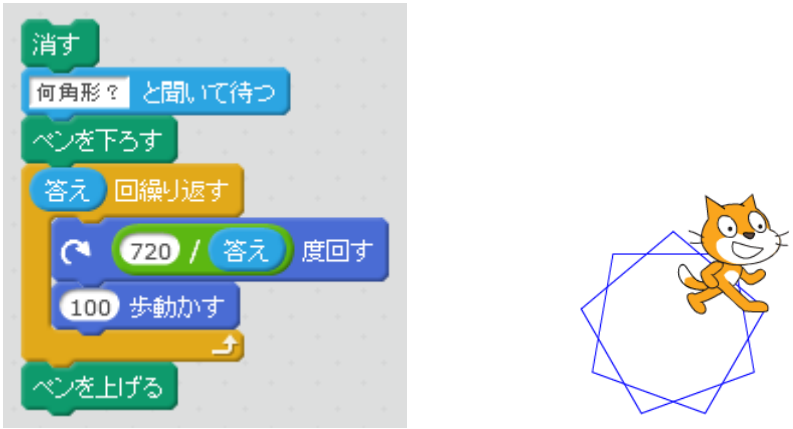
ペンを利用して図形を書いてみます。少しずつ向きを変えながら線を引くと正多角形が描けますが、正 n 角形を描くときは、どれだけの角度曲がればよいか考えると、下のようなプログラムになります。

図 66: 正多角形



向きを変える角度を少し変えるとどうなるでしょうか。

図 67: 正多角形のバリエーション



4.3.4 バブルソート

リストの要素を小さい順に整列するプログラムを作成してみます。どんな教科書でも最初に紹介されるという意味で一番有名なバブルソートというアルゴリズムで整列してみます。バブルソートとは、次のようなアルゴリズムです。

(1) リストの先頭の方から順に隣接する要素を比較して、その 2 要素を小さい順に並べ換えます。
(2) これがリスト末尾まで終わると、最後の要素はリストの最大値になっています。
(3) リストの最後の要素以外の部分に同じ手順を施すと、それらのうち最大のものは、最後から 2 番目に格納されます。
(4) 以下同様にすると、注目する部分が短くなっていき、最後にはリスト全体が整列されます。

下のプログラムでは、点数という名前のリストを整列しています。また、変数 a は注目する部分の末尾の番号、変数 b は隣接する 2 項のうち若い方の番号、変数 x と y は隣接 2 要素を交換するときの補助的な変数です。

図 68: バブルソート

下図左のようなリストがあるときに、バブルソートを実行すると下図右のように整列されます。

図 69: バブルソートの実行結果

4.3.5 クイックソート

バブルソートは最初に紹介されるという意味でも有名ですが、遅いという意味でも有名です。最近のコンピュータは高速ですから遅いといっても早いのですが、Scratch の動作速度はそれほど早くないので、別のアルゴリズムとの差が体感できます。

ここではクイックソートという、最も高速な整列アルゴリズムの 1 つを用いてプログラムを作成してみます。クイックソートのアルゴリズムは以下の通りです。

(1) リストの末尾の要素をピボットと名付ける。
(2) リスト全体を、ピボット未満の値たち、ピボット、ピボット以上の値たちの順に並べ換えます。「ピボット未満の値たち」と「ピボット以上の値たち」の部分は整列されていなくて構いません。
(3) 「ピボット未満の値たち」と「ピボット以上の値たち」の部分をクイックソートします。

(3) の部分で部分リストをクイックソートする、つまり自分自身を呼び出しています。これを実現するためには、「定義」ブロックを利用して関数を作成します。このような関数を再帰関数と呼びます。

プログラムは以下のように作成できます。まず、変数 i と j は並べ換えの終わっていない範囲の番号を表す変数で、変数 p はピボットの値です。

図 70: 必要な変数

リストも用意します。

図 71: 整列するリスト

プログラムは下のようになります（「でなければ」の下が空いていますが、これはその下の 3 つのブロックが移動して入るべきです。撮影の時には間違いに気付きませんでした。すみません）。

図 72: クイックソート

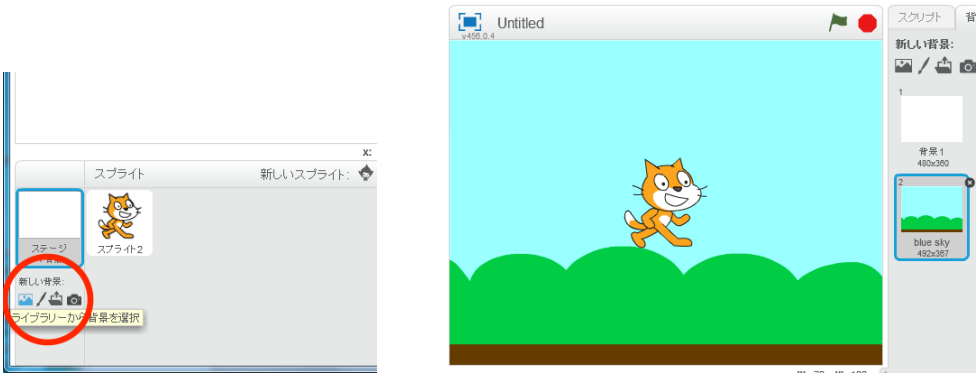
整列前と後は、下のようになります。

図 73: クイックソートの実行結果

4.4 その他の機能

背景は、下図左の丸で囲んだ部分で変更できます。用意されているものを利用したり、自分で描くこともできます。変更すると、途端に雰囲気が変わります。

図 74: 背景



スプライトは複数同時に用いることができます。下図左の丸で囲んだ部分で追加できます。用意されているものを利用したり、自分で描くこともできます。

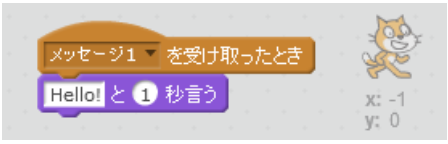
図 75: 複数のスプライト

スプライトが異なると、プログラムも別のものになります。オブジェクトがプログラムを持っている感覚です。このように賢いオブジェクトを作る方針のプログラミングをオブジェクト指向と呼びます。

プログラムが共通でないため、いろいろ不便に思えますが、Scratch ではオブジェクト間で通信することで不便さを補います。その仕組みは「メッセージ」と呼ばれるものを使って実現されています。「メッセージ」を送ったり受け取ったりするためのブロックは、「イベント」のブロックにあります。

もとからいた猫は、メッセージの受け手とし、メッセージを受けたら Hello!と言うことにします。

図 76: メッセージの受け手

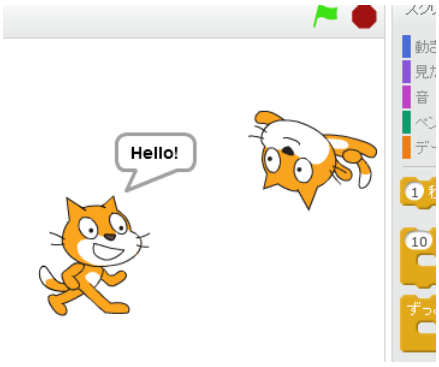


2 匹目の猫は、メッセージの送り手とし、動いている最中に領域端に着いたら、メッセージを送ることにします。

図 77: メッセージの送り手

絵ではわかりづらいですが、2 匹目の猫が端に当たって跳ね返ったときに、1 匹目の猫が Hello!と言っています。

図 78: 実行結果



5　おわりに

プログラミングに必要な能力としては、例えば以下のものがあげられます。

- 目的達成に必要なことがらを、一連の手順に整理できる（アルゴリズムの作成）
- 具体例で検証できる（動作検証、アルゴリズム作成の手掛り）
- 場合を網羅できる、例外に気付ける（正しい条件の記述）
- 困難を分割できる（関数定義、意味のあるまとまりを考える）

これらは、数学で要求される論理的な思考力の中でも重要なものばかりです。

- 目的達成に必要なことがらを、一連の手順に整理できる（解答や証明を文章に表現できる）
- 具体例で検証できる（検算、帰納的推論）
- 場合を網羅できる、例外に気付ける（隙のない論理）
- 困難を分割できる（長い解答や証明を表現・理解できる）

「プログラミング的思考」が導入されることとなり、賛否いろいろな意見があると思いますが、その趣旨に肯定的な立場からすると、「プログラミング的思考」は本当に重要で、算数・数学のみならず多くの教科で有用で、社会に出た後も有益であると、本心から思います。数学は役に立たないというよく耳にする意見に対して、算数・数学の教員であれば、そんなことはないという気持ちを本音として持っていると思いますが、それと同じです。ただ、実際にどのような授業をすれば、「プログラミング的思考」が身に付き、それを様々な場面で活用できるのかは、今はまだ非常に難しい問題です。本講習が少しでも参考になればさいわいです。

参考文献

学習指導要領などの他に、次の文献を参考にしました。

[BWF] Tim Bell, Ian H. Witten, Mike Fellows. 兼宗進 (監訳) コンピュータを使わない情報教育アンプラグドコンピュータサイエンス イーテキスト研究所, 東京, 2007

平成２９年度
教員免許状更新講習修了試験問題

会 場 名	北海道教育大学釧路校会場
講 習 名	スクラッチ入門

注意事項

① 指示があるまで問題紙を開かないこと

② 問題紙は全部で３枚で、問１から問５まであります。

③ 解答は指定された解答用紙に記入してください。

④ 受講者ＩＤ、氏名は指定された欄に記入してください。

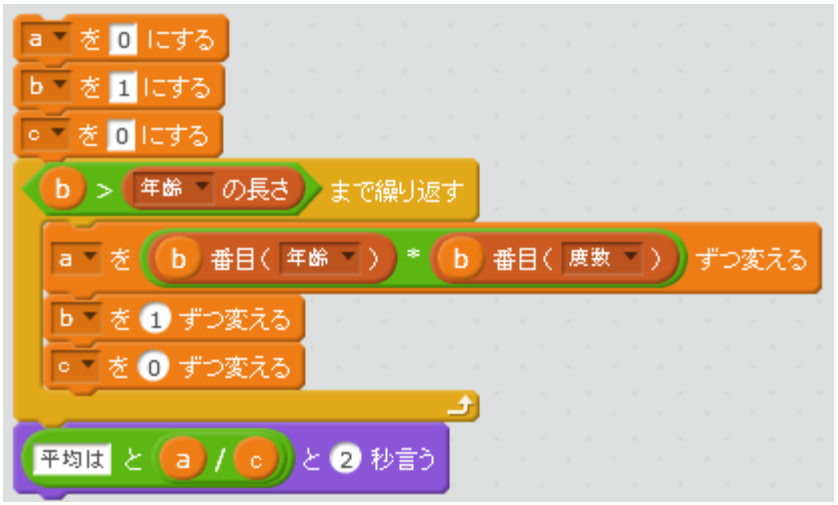
⑤ 本試験は、問題紙、解答用紙とも回収します。

問題 1　Scratch を学校教育で利用するときの利点・欠点を、Scratch の特徴を踏まえて答えよ（解答用紙の 1 ページの 1/3 から 1/2 の長さで書くこと）。

問題 2　本講習で紹介したプログラムについて、どういう点がプログラミング的思考につながると思えたか述べよ（解答用紙の 1 ページの 1/3 から 1/2 の長さで書くこと）。

問題 3　下のように、年齢のリストと、その年齢の人が何人いるかの度数のリストがあったときに、平均を求めるプログラムを作成するとします。

下のような変数とリストが用意されていて、プログラムも作成してあります。



しかし、正しく平均を求めるには、「c を 0 ずつ変える」の「0」が誤りです。これをどのように修正すればよいか、（ブロックではなく）言葉で答えよ。

問題 4　下のプログラムを実行すると、どんな図形が描かれるか答えよ。必要であれば、図をかいて説明してもよいが、図だけではなく説明も記すこと。

問題 5　ある自然数が与えられたとき、そのすべての約数をリストに格納するプログラムを作成したい。どのようなアルゴリズムが考えられるか 1 つ述べよ。

平成２９年度

教員免許状更新講習修了試験　　解答用紙 (その１)

講 習 名	スクラッチ入門
受講者ＩＤ	
氏　名	

得 点

〔解答欄〕

平成２９年度

教員免許状更新講習修了試験　　解答用紙 (その２)

講 習 名	スクラッチ入門
受講者ＩＤ	
氏　名	

〔解答欄〕

採点基準

問題 1　(20 点) 利点・欠点で 10 点。Scratch の特徴を踏まえていれば 10 点。

問題 2　(20 点) 本講習のプログラムに言及していれば 10 点。プログラミング的思考に言及していれば 10 点。

問題 3　(20 点) 言葉で説明できていればよい。説明不足であれば程度に応じて減点。

問題 4　(20 点) 言葉や図で説明できていればよい。説明がなく図だけであれば 10 点のみ。

問題 5　(20 点) アルゴリズムとして正しければ 20 点。曖昧さが残るようであれば 10 点のみ。