マップ内コメント・ヒント　基本ステージ

Tutorial

Basiccal

Array1

Array2

Ifmaze

Output

Lobby

Lastmap

ほかに必要な情報

Tutorial

書法・プログラムはじめおわり

プログラミングをするにあたって知っておくべきことの一つとして、メソッドや関数と呼ばれるものがあります。メソッドとは何らかの演算機能を持った命令であると思ってください。このメソッド、命令は自分で作ることもできますし、他人が作ったものを使うこともできます。ここでは用意されたメソッドを使ってコーディングします。例えば、method1,method2というメソッドがあるとします。これらを使うにはコーディング画面に、

method1

method2

と記述します。基本的にプログラムは上に記述してあるものから順に実行されます。ここではmothod1,method2の順に実行されます。それでは、メソッドをマップ内から探して適切に記述してください。

Basiccal

|  |  |
| --- | --- |
| 正解コード |  |
|  | def calc(tubo, oke, hako, tubo\_covered) |
|  | # この下にソースを入力 |
|  | Integer a = tubo + oke |
|  | Integer b = tubo - hako |
|  | Integer c = tubo / hako |
|  | a = a + b + a \* b |
|  | a = a / c \* hako |
|  | a = a % oke |
|  | tubo\_covered.setResult(a) |
|  | end |

書法・変数

プログラミングで演算を記述するときには、ほとんどの場合変数を用います。

変数とは数値を入れておく箱のようなものです。ここでは壺や木箱がその変数です。

また、変数に数値を入れることを代入と言います。たとえば、変数tuboとokeの数値の和を変数aに代入する時には、a=tubo+okeと記述します。四則演算の記号は加算+、減算-、乗算\*、除算/、除算のあまり%を使います。また、変数aにaの2倍の値を代入する場合はa=a\*2と記述します。=の右側の演算が先に行われてから代入されることがポイントです。

Array1

書法・配列

配列を使えば1つの変数名でたくさんの値を扱うことができます。例えば配列変数名をarrayとしたときarray=[1,2,4,8,16]と記述すればこの場合、1,2,4,8,16の5個の値を扱うことができます。一列につながった箱を想像すると良いでしょう。これらの値を配列の要素と言います。これらの要素を個別に扱うときにはarray[0],array[1],,,array[4]、のように変数名を指定します。配列の要素のスタートが0番目扱いであることがポイントです。ここでは5個の要素を扱っているため0~4番で指定します。また、配列の要素array=[1,2,4,8,16]は左から順に0,1,2...番目が割り当てられます。ここではarray[0]には1,array[1]には2,array[4]には16が変数の値として割り当てられています。これらのことから要素を指定するとa=array[2]+array[0]といった演算を行うこともできます。また要素を指定して代入を行うことで配列の要素を置き換えることもできます。array[2]=0とすると配列に入っている値は順に、1,2,0,8,16となります。

Array2

書法・while

繰り返し処理について説明します。繰り返し処理とは、ある命令を指定の回数または、無限に実行することをいいます。ここでは予約語while文による繰り返し処理を使います。まず、while文の基本的な書法を示します。

while 繰り返し条件

文

end

この場合文が繰り返し条件に従って実行され続けます。whileとendで囲まれた文に繰り返しは適用されます。

具体例をあげます。

i=0

while i<10

method1

i+=1

end

この例ではmethod1とi+=1がi<10を満たしている間繰り返し実行されます。ここでi+=1という演算がありますが、これは代入演算子と呼ばれるものの一つであり、i=i+1と同じ実行結果が得られます。これにならって、a-=2,a\*=2などがa=a-2,a=a\*2と同じ実行結果となります。ここで、while文の繰り返し条件を見てみるとi<10となっています。またi+=1によってwhile内の文が実行されるたびにiの値が1ずつ増えています。これによって、iの値が0,1,2...8,9,10と更新されていき10以上になったときwhile文は実行されなくなりループを抜けることになります。while文のポイントはi+=1のような繰り返し条件を決めている変数の値を変更して、正常に実行が終了するように記述することです。上の例においてもi+=1を書き忘れればwhile i<10の条件は常に満たされることになりwhile内の文は実行され続け、無限ループに陥りプログラムが終了しなくなります。もう一つのポイントは繰り返し条件を満たしているかどうかをどのタイミングで判定しているのかについてです。プログラムは上から順に実行される原則がありますから、それに則るとwhileの中身の文が実行される前に繰り返し条件の判定が行われます。もしも、この段階で条件を満たしていなければwhile文の中身は一切実行されずにそれ以降の文が実行されます。さらにwhileの最後のendに到達した際には一度先頭のwhileにもどり、この段階で繰り返し条件を満たしているかどうかを判定します。

また、配列を扱う際に繰り返し処理がとても便利です。特に規則的に配列の値を扱う際に用いられます。たとえば配列の値をすべて3倍にするプログラムを記述します。

array=[0,1,2,4,8]

i=0

while i<5

array[i]=array[i]\*3

i+=1

end

この実行過程を説明します。iの初期値は0です。またwhileの繰り返し条件はi<5ですから、while文内の命令が実行されます。while文内に入るとarrayの0番目の要素が3倍されて再びarrayの0番目の要素に代入されます。つぎにiが1加算され値が更新されます。繰り返し条件はi<5ですから、iの値が1の状態で先の命令が実行されます。配列の1番目の要素が指定されます。それらが繰り返されてiの値が5になったときには繰り返し条件を満たさなくなるのでwhile文から抜け出します。

Ifmaze

もしxxのときにooしたいという命令を書きたいならば予約語if文を使いましょう。基本的な書法を示します。

if 分岐条件文

実行文

end

分岐条件を満たしているときにif-endで囲まれた実行文が実行されます。

具体的な例をあげます。

i=3

if i<5

method1

end

if i>10

method2

end

のような７行のコードがあります。まずiの値は3と初期化されています。次にif文の条件分岐が入り、i<5を満たしているかどうかを判定します。この場合iは3ですから条件を満たすため、if-endで囲まれたmethod1が実行されます。次に別のif文が登場します。この分岐条件はi>10ですからmethod2は実行されずにプログラムは次へ進みます。このように、条件分岐によってあるメソッドや演算の実行を制御することができます。また以下のようにプログラムを記述することができます。

if i<3

method1

if k>5

method2

end

end

このようにif-endの中にif-endを記述することができます。この例ではi<3ならばmethod1を実行します。さらに同時にk>5であればmethod2が実行されます。これはmethod2を実行するにはi<3とk>5の2つの条件を同時に満たしておく必要があります。ですから、そもそもi<3を満たしていなければk>5であるかどうかの判定は行われることなく2つ目のendまでプログラムの実行は進みます。

Output

演算を実行したらその演算が正しいかどうか確かめることができます。その手段の一つとして標準出力があげられます。標準出力とは変数に入った値や文字等をpc画面に出力することです。これを演算後の変数に用いると変数に入っている値を人間が目で見て確認することができます。それでは標準出力putsの書法を示します。

puts 変数

や

puts 文字列

などです。

具体例をあげます。

a=1

puts a

puts "Hello"

このように書くとpc画面には以下のように出力されます。

1

Hello

上のように出力されます。ここで文字列の扱い方ですが、ある文字列を扱いたいときには必ず"　"で囲む必要があります。文字列が命令や、変数でないことを区別するために必要です。

Lobby

これからプログラミングを学びます。学ぶことはプログラミングの文法です。基本的な予約語を中心に学習します。予約語とはプログラミング言語にあらかじめ用意された、基本的な機能を持った命令です。また予約語を適切な文法にしたがって用いることでプログラムに機能を持たせることができます。

Lastmap

Congratulations!

プログラミング基礎ステージをすべてクリアできました。この基礎ステージで学んだことを駆使してアルゴリズムステージに挑んでみてはどうでしょうか。検討をお祈りします。また、プログラミング言語は何千種類もあるといわれていますが、ここで学んだ文法はどの言語にも当てはまるような基礎的なものですから、きっと別の言語を学ぶ時の助けになるでしょう。

ほかに必要な情報

2次元配列

nil

boolean

2次元配列

配列といえば箱が一列に並んだものを想像してもらっていましたが、その配列をさらに横に並べることができます。それを2次元配列といいます。箱がオセロや将棋のマス上にぎっちり詰まって置いてあるものを想像するといいでしょう。配列の扱い方はすでに知っているかと思いますが、配列を別の方法で作ることができます。書法を示します。

Array.new([要素数[, 値]])

値は省略して記述できます。以下に具体例を示します。

array1 = Array.new(3)  
このように記述することで大きさ3の配列を作成できます。また、この配列には初期値として空を意味するnilというものが入ります。

また2次元配列を作成するときには以下のように記述します。

a = Array.new(第一要素数).map{Array.new(第二要素数,初期値)}

ここでは基礎編ですから詳しい説明はしません。今はある意味呪文と思って覚えておく必要があります。3\*3の2次元配列を作成する具体例をあげます。

a = Array.new(3).map{Array.new(3,0)}  
a[0][0] = 5

p a

以下のように出力されます。

[[5, 0, 0], [0, 0, 0], [0, 0, 0]]  
アルゴリズムステージで2次元配列を頻繁に使うので覚えておきましょう。

nil

nilとは配列の初期化をしないと勝手に入る予約語です。これは変数等に値が入っていないことを示す予約語と覚えておきましょう。値がないのですから0とは異なることがポイントです。