タミヤロボットスクール

ロボットプログラミングコース

講師用ガイド

Unit8.サブプロシージャ

「サブプロシージャを使いこなす」

Ver 1.1.0

Unitの学習目標

サブプロシージャの作り方と処理の流れを理解して、プログラム全体の構造を捉える視点を養う。

PRACTICE1の学習のめあて

サブプロシージャを扱う場合のコマンドと処理の流れを学び、そのメリットについて理解する。

ワークシート「今日の目標」例

① サブプロシージャを使ったプログラムを作る
②「ロボ暗記」を完成させて、プログラムを改造する

授業の流れ（1回/90分）

時間	学習内容	テキストタイトル
10	ワークシート Unit解説	サブプロシージャとは？
10	WARM UP1	GSBコマンドとRNTコマンドの使い方 （問題数：11問）
8	TRY IT1	
10	WARM UP2	サブプロシージャのメリット （問題数：11問）
8	TRY IT2	
4	チェックポイント	
30	ロボット制御 〈ゲーム＆改造〉	「ロボ暗記」
5	片付け	
5	ワークシート	

(90分)

8.サブプロシージャを使いこなす

▶ サブプロシージャとは？

MESSAGE

プログラムのあちこちで同じコマンドを使うことはとても多い。
そのコマンドが短いものなら気にしなくてもいいけど、もし

```
A=12345:?"ABCDEFGHIJKLMNOPQRSTUVWXYZ":B=23456
```

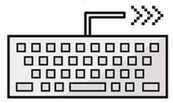
というような長いものだったらどうしよう。
あちこちで同じコマンドを打つのはとても大変だ。
そんなときは「サブプロシージャ」を使おう。
長いコマンドをあちこちで使うのがとても楽になるぞ。

```
たくさんのコマンドがつながっていて打つのがとても大変だけれど何回も実行したいコマンド:コマンド1:コマンド2:たくさんのコマンドがつながっていて打つのがとても大変だけれど何回も実行したいコマンド:コマンド3:たくさんのコマンドがつながっていて打つのがとても大変だけれど何回も実行したいコマンド:コマンド4:コマンド5:たくさんのコマンドがつながっていて打つのがとても大変だけれど何回も実行したいコマンド
```

長いコマンドを何度も打つのは大変で、
プログラムも読みにくい

```
★:コマンド1:コマンド2:★:コマンド3:★:コマンド4:コマンド5:★

★…たくさんのコマンドがつながっていて打つのがとても大変だけれど何回も実行したいコマンド
```



サブプロシージャを使うと、打つのが楽
になるし、プログラムも読みやすい

講師への解説

サブプロシージャとは、ある一連の処理を単位としてまとめたものです。
プログラムは複雑な処理が増えるほど長くなり、処理の流れも複雑になります。
そうなると、プログラム全体を読んで内容を理解することが難しくなってしまいます。
しかし複雑な処理が多くても、プログラムの一部分を「小さな単位＝サブプロシージャ」
という単位でまとめていくことで、読み易い・理解し易いものにすることができます。
（プログラムの読み易さ・理解し易さのことを可読性と言います）

サブプロシージャには、以下のようなメリットがあります。
1.同じ処理を何度も記述する手間が省ける
2.同じ処理が何度も登場するようなプログラムを書きやすくなる
3.プログラムが読み易くなる
4.プログラムを直しやすくなる

サブプロシージャと同じような役割を担うものに「関数」があります。
関数とサブプロシージャの違いは「それに対する入出力があるかないか」です。
ただしIchigoJamでは厳密な意味での関数を定義することができません。
また、サブプロシージャと関数の違いは僅かなものであるため
（プログラミング言語によってはふたつを区別しないものもあります）、
本Unitではサブプロシージャについてのみ学びます。

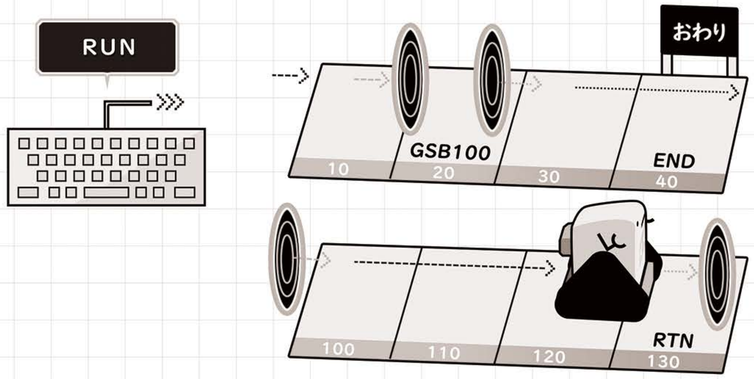
8.サブプロシージャを使いこなす

PRACTICE1

▶ GSBコマンドとRTNコマンドの使い方

MESSAGE

GSBというコマンドを使えば、好きな行番号にジャンプできる。
GSBコマンドでジャンプした後は、RTNコマンドで元の場所にもどれるぞ。
GSBコマンドとRTNコマンドを使うときは、プログラムをその場で
終了するENDコマンドも使おう。



▼ 好きな行番号にジャンプする方法

1

GSB 行番号 ↵

▼ GSBコマンドでジャンプした後、元の場所にもどる方法

RTN ↵

▼ プログラムを終了する方法

END ↵

指導の流れ

ここでは、GSBコマンドとRTNコマンドの使い方と実行される行番号の順序を理解し、サブプロシージャを作成する方法について学びます。
また、サブプロシージャを作成した場合におけるENDコマンドの役割についても学びます。

講師への解説

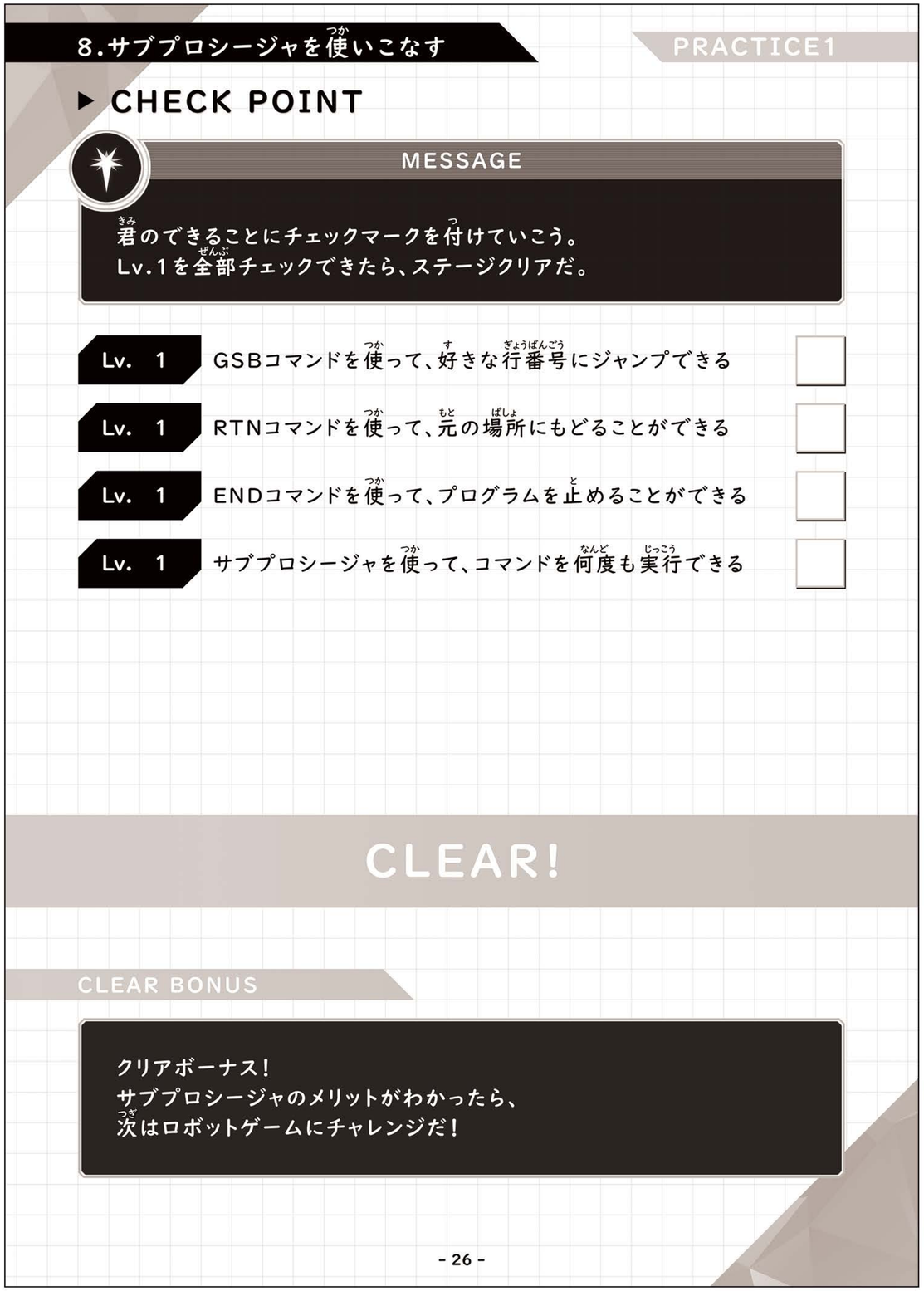
GSBコマンドは、次に実行される行番号を変更するコマンドです。
GSBコマンドが実行されると、処理は引数で指定された行番号に移り、以降のコマンドが実行されていきます。
「Unit6.くり返し処理」で学んだGOTOコマンドは、特定の処理をスキップしたりループさせたりするためのものでしたが、GSBコマンドは、サブプロシージャを作ることを目的としているため、挙動がやや異なります。
GSBコマンドで指定した行番号に処理が書かれていない場合、「Line error in ○○」というエラーメッセージが表示されます。
また、GSBコマンドは、「GOSUB」の省略形です。

RTNコマンドは、既にGSBコマンドが実行されている場合にのみ実行できます。
GSBコマンドによって「GSBコマンドで指定した行~RTNコマンドが実行される行までの処理」が行われ、RTNコマンドによってGSBコマンドが実行された地点に戻る、というイメージです。
「GSBコマンドで指定した行~RTNコマンドが実行される行までの処理」を、サブプロシージャと呼びます。
サブプロシージャが作られることで、プログラムはメインの道と、サブの道に分割されます。
ふたつの道を混同しないように、サブの道の行番号はメインのものと比べて大きくすることが推奨されます。
プログラム中にRTNコマンドだけの場合や、GSBコマンドとRTNコマンドが正常に組み合わせられていない場合、「Not match in ××」というエラーメッセージが表示されます。
また、RTNコマンドは、「RETURN」の省略形です。

ENDコマンドは、プログラムを終了させるコマンドです。
プログラムは一番大きな行番号が実行されると終了しますが、ENDコマンドが実行されると行番号によらずプログラムは終了します。
サブの道を作った場合は、メインの道の最後でENDコマンドを実行しなくてはなりません。
ENDコマンドを実行しないと、メインの道の後で、サブの道の処理が実行されてしまうからです。（エラーが発生する原因となってしまいます。）

指導ポイント

1 ここでは、コマンドの紹介に留め、次ページの例のプログラムを用いて、GSB,RTN,ENDコマンドの役割を説明するとよいです。



指導の流れ

チェックポイント項目を読み上げ、各項目について簡単に解説します。
解説の時は、「各チェック項目の解説」を参考にして、WARMUPの例題を交えながら説明します。
生徒は、授業を振り返りながら項目にチェックマークを記入します。
基本的に、全ての項目にチェックマークが付けられるように誘導し、チェックできた生徒に手をあげてもらいます。
WARMUPとTRYITをやり遂げたという達成感を持ってもらうように心がけ、拍手などで盛り上げて、ゲームに移ります。

各チェック項目の解説

- **【Lv.1】GSBコマンドを使って、好きな行番号にジャンプできる** ➡ WARMUP1
  GSBコマンドをプログラムに記述し、指定した行番号へ処理を移すことができるかどうか。
- **【Lv.1】RTNコマンドを使って、元の場所にもどることができる** ➡ WARMUP1
  サブプロシージャの処理の最後にRTNコマンドを記述し、処理を元の場所に戻すことができるかどうか。
- **【Lv.1】ENDコマンドを使って、プログラムを止めることができる** ➡ WARMUP1
  プログラムの適切な行にENDコマンドを記述し、プログラムの処理を停止させることができるかどうか。
- **【Lv.1】サブプロシージャを使って、コマンドを何度も実行できる**➡ WARMUP2
  自由にサブプロシージャを何度でも呼び出すことができるかどうか。

8.サブプロシージャを使いこなす

ロボット制御

▶ ロボ暗記

MESSAGE

これはボタンをおすとロボットが動き出すプログラムだ。
ロボットは、3回動くぞ。
ロボットの動きは、5パターンあるぞ。

NEW

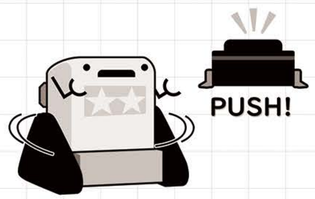
```
5 LED 1
10 IF BTN()==0 GOTO 10
20 SRND TICK()
30 A=RND(5):B=RND(5):C=RND(5)
40 I=A:GSB 300
50 I=B:GSB 300
60 I=C:GSB 300
70 END
300 IF I==0 P=33
310 IF I==1 P=1
320 IF I==2 P=32
330 IF I==3 P=2
340 IF I==4 P=16
350 OUT P:WAIT 60
360 OUT 0:WAIT 30
370 RTN
```

40～60行目、70行目、370行目にGSB，RTN，ENDが使われているぞ
300～370行目がサブプロシージャだ

RUN

プログラムを打ち込んだら、次のコマンドで0番にセーブしよう。

```
SAVE 0
```



RUNコマンドでプログラムを実行しよう。
エラーが出たら、まちがっているところを直してセーブだ。
エラーがないなら準備OK!
スイッチを切って、モニターとキーボードのコードを外そう。
IchigoJamのボタンをおしながらスイッチを入れれば、プログラムが実行されるぞ。

【ゲームの遊び方】
ボタンをおした後の動きをおぼえられるかどうか、チャレンジだ。

ロボットの動きとゲーム案

ロボットのランダムな動きを覚えるゲームです。
プログラムを実行してIchigoJamのボタンを押すと、ロボットは3回動きを見せます。
ロボットの動きは5種類の中からランダムで選ばれます。
初期のプログラムは、一度実行するとプログラムが終了してしまうため、早い段階で
改造ポイント4の改造を行うように指導すると良いでしょう。
（改造を呼びかける前にENDコマンドの役割を確認しましょう）
・ゲーム案
暗記した正解数を競う。
改造により動きの回数を増やし、何回まで暗記できたかを競う。

ゲームプログラムの解説

・プログラムの中のサブプロシージャ
このゲームプログラムの構成は、5～70行目がメインプログラム、300~370行目がサブプロシージャとなっています。
ここでのサブプロシージャは、ロボットを動かす重要な役割をもつサブプロシージャです。
サブプロシージャ内では、決定した動きのパターンに対して、出力となるOUTコマンドの引数を判断するような処理を行います。
このゲームプログラムでは、サブプロシージャのメリットを活かすための工夫が施されています。
その工夫とは、変数Iを用いて一つのサブプロシージャの処理を行うという手法です。
メインプログラムの30行目で、変数A~Cの3つの変数にランダムに決められた値を代入し、一旦3つの動きを決定します。
その後の40～60行目で、GSBコマンドを実行する前に、変数Iにそれぞれ変数A~Cの値を代入するようにしています。
これにより、3つの変数が別々の値をもっていても、同じサブプロシージャで対応することができます。

・変数の役目
A~C …… ロボットの動きを決める変数。
Aは動きの1回目、Bは2回目、Cは3回目。
I …… サブプロシージャで用いるための変数。
P …… OUTコマンドの引数として用いる変数。

・行ごとの解説
-行番号5
プログラムが実行されたことを確認するためにLEDを点灯します。
-行番号10
ボタンが押されない限り、この部分の処理がくり返されます。

次ページに続きます

▶ ゴールを目指せ！DX

MESSAGE

これは、ボタンをおすとロボットが動くプログラムだ。
ゴールまでたどりつけるように、プログラムを完成させよう。

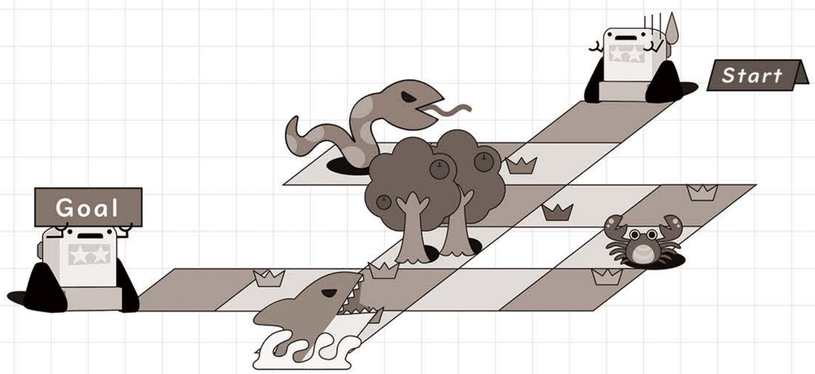
プログラムの行番号を大きくして、プログラムをどんどんふやしていこう。
「1マス前に進むサブプロシージャ」、「90度右に回転するサブプロシージャ」、といったように、
よく使うロボットの動きのサブプロシージャを作ろう。

NEW

```
5 LED 1
10 IF BTN()==0 GOTO 10
20 GSB 500
30 GSB 600
 :
400 END
500 OUT 33:WAIT 60:OUT 0
510 RTN
600 OUT 17:WAIT 60:OUT 0
610 RTN
 :
```

RUN

500行目からがサブプロシージャだぞ
プログラムを終了するENDコマンドは、わすれずに書こう



ロボットの動きとゲーム案

本ゲームのロボットの動きは、基本的にはUnit2.待機命令「ゴールを目指せ」と
ほとんど同じになります。
ただし、今回のゲームの目的は、サブプロシージャを自分で作り、サブプロシージャを駆使して、どんなコースでも対応できるようにすることです。
ゴールをすることがゲームのクリアではありますが、サブプロシージャをどれだけ使いこなせるかが、重要なポイントです。
また、これまで学んできたプログラミングの知識と技術を活かしてプログラムを組み立てることが望ましいです。
変数やIFコマンドによる分岐処理、GOTOコマンドによるくり返し処理など、
幅広く応用してプログラムを組み立てるよう、指導してください。

- 準備物
・コースのための道具
Unit2.待機命令の「ゴールを目指せ」で使用したものと同じでもよいです。
また、コースの難易度を上げるために、障害物となるものも用意するとよいでしょう。
・コースの準備
初級、中級、上級、とレベル分けをして作成しましょう。
初級のコースは、サブプロシージャが上手く機能しているかどうかを確かめるための
コースとして考えて作りましょう。

ゲームプログラムの解説

・行ごとの解説

-行番号5
プログラムが実行されたことを確認するためにLEDを点灯します。

-行番号10
ボタンが押されない限り、この部分の処理がくり返されます。

-行番号20
GSBコマンドで、行番号500へ処理を移します。

-行番号30
GSBコマンドで、行番号600へ処理を移します。

-行番号31～399
サブプロシージャを用いて、ロボットを動かすメインプログラムを組み立てましょう。
基本的には、作成したサブプロシージャを呼び出すGSBコマンドを、
コースをクリアできるように順番を考えて記述すればよいです。
距離センサーを用いて壁や障害物を避ける、ロボットのアームを振る、音を鳴らす、
LEDを点灯する、など、様々なアクションを行うように、自由にプログラムを組み
立ててもよいでしょう。

次ページに続きます

指導の流れ

2回目の終わりには、Unitが修了したことを伝えてください。
また、各PRACTICEでの学習内容を、「CLEAR STAGE」を読み合わせながら簡単に振り返りましょう。

Unit8の修了の時点で、全てのUnitの学習が修了します。
「UNIT COMPLETE」の文章を読み上げ、全てのUnitを学び、プログラミングの基礎を習得できた生徒たちを拍手などで褒めましょう。
生徒一人一人に、本部から送られた修了証を渡してください。
その後、コース2年目もプログラミングを頑張るよう気持ちを高めて、1年目を結びます。

変更履歴

- **2019/02/18　-Ver1.0.0**
  初版
- **2022/02/16　-Ver1.1.0**
  生徒用テキスト（Ver.2202）の変更を反映