

第1章

Introduction

1.1 準備

1.1.1 この講義の目的

大学生活において、さらには社会人生活において、何かを的確に表現することは非常に重要となってくる。この講義では以下の3項目を主目的として進めていく。

1. 情報化社会で必要となる知識を身につけ、活用させる能力を修得すること。
レポート作成や卒業研究に必要となるツール (LaTeX) の利用方法を学習する。
2. 中学・高校の教員になって役立つ情報を得ること。
GeoGebra を利用した視覚的な中学・高校数学の学習し、実習を行う。
3. 2年次以降の講義で必要となる予備知識を身につけること。
プログラミング言語の導入として、簡単なプログラミングの仕組みについて Scratch の実習を行う。

1.1.2 ディレクトリ構造

コンピューターにおいて、ファイルの管理は、ディレクトリ (= フォルダ) の階層構造で行われている。Windows パソコンならエクスプローラーから確認ができる。

Windows 10 なら検索 (   ) からエクスプローラーと入力 (図 1) すると、表示される (図 2) エクスプローラーを起動する。



図 1



図 2

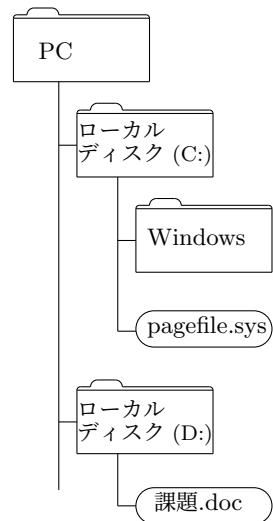
なお、応用数学科 計算機室のパソコンではツールバーの    にあるエクスプローラーを起動する。

エクスプローラーでディレクトリとファイル見ると、右の図のような階層構造をしている（一部のみ掲載）。

PC、デスクトップや、ローカルディスク (C:) というディレクトリがあり、それぞれのディレクトリには（サブ）ディレクトリやファイルが収納されている。

応用数学科計算機室のパソコンでは、PC の下層ディレクトリとして、

PC の下層ディレクトリ
ローカルディスク (C:)
qemu(¥¥172.16.2.2) (D:) …… (*)



あることに注意する。今後、

『ローカルディスク D』, 『D ドライブ』, 『ボリューム D』

と言ったりすることがあるが、いずれもここにある (*) のことをさしている。

課題 1.1.1. 応用数学科 計算機室のパソコンを使用している場合は、D ドライブの直下にサブフォルダ `files` を作成し、`files` のサブフォルダとして

`scratch`、`tex`、`geogebra`

の 3 つのフォルダを作成せよ。

サブフォルダの作成は、エクスプローラーの左側のウインドからサブフォルダを作成したいフォルダをマウスで右クリックし、新規作成からフォルダを選ぶ。

1.1.3 拡張子

この図にあるファイル `pagefile.sys` や `課題.doc` の . 以降をファイルの **拡張子** という。したがって、`課題.doc` の拡張子は `doc` である。

拡張子には“ファイルの種類を識別する”という重要な役割がある。拡張子の例として

`.sys` `.exe` `.com` `.bat` `.txt` `.html` `.doc` `.jpg` `.c` `.JAVA` `.mp4` `.wma` `.tex`

などなど沢山ある。

Windows パソコンを扱っていると拡張子を見る機会が減っているが、無くなったわけではなく、隠れているだけである。拡張子を確認したい場合は、エクスプローラーの設定を変更したり、コマンドプロンプトを利用して確認出来る。

1.1.4 ショートカット

パソコンでの作業でマウスを使う場面は多いが、文章の入力中にマウスに手を持って行くのは手間である。そこで、いくつか便利なショートカットキーを紹介しておく。

ただし、これらは Microsoft Windows の場合であり、Mac OS では異なる。

[Shift] + ↑ ↓ ← →	文字の“センタク”
[Ctrl] + ← →	“カタマリ”の前後まで移動 ([Shift] 併用可)
[Ctrl] + a	表示されているモノ全てを“センタク”
[Ctrl] + c	“センタク”したものを“キオク”する
[Ctrl] + s	編集中のファイルを上書き保存する
[Ctrl] + v	“キオク”したものを貼り付ける
[Ctrl] + x	“センタク”したものを削除して“キオク”する
[Ctrl] + z	アンドゥ(取り消し)をする
[Ctrl] + Tab	タブの切り替え
[Alt] + Tab	ウィンドウの切り替え
[Ctrl] + r	置換のウィンドウを開く
[Ctrl] + f	検索のウィンドウを開く
[Ctrl] + n	新規ウィンドウ、文章を開く
[Ctrl] + w	ウィンドウを閉じる ([Alt] + F4)
[Home]	カーソルキーのある行の先頭に移動する
[End]	カーソルキーのある行の終端に移動する

IME(日本語入力のとき) のショートカット (の一例)

[Ctrl] + u または [F6]	ひらがな	[Ctrl] + i または [F7]	全角カタカナ
[Ctrl] + o または [F8]	半角カタカナ	[Ctrl] + p または [F9]	全角英数
[Ctrl] + t または [F10]	半角英数		

♠ 補足 (挿入モード、上書きモード) -

ショートカットキーではないが、入力方法では“挿入”と“上書き”的2つのモードがある。これらの切替は **[Insert]** キーまたは **[Ins]** キーで行う。

ab|cd の状態 (|はカーソルキーの位置) で、“挿入”モードのとき **e** を入力すると **abe|cd** となり、“上書き”モードのとき **e** を入力すると **ab|e|d** となる。

例 1.1.1. 以下の (A) の状態からショートカットキーを用いて (B) の状態に編集する。ただし、カーソルキーの初期位置は 789 の 7 の前とする。

(A) 789 456 123

(B) 123 456 789

このときの入力は以下の通りとなる。

```

End
Shift + Ctrl + ←
Ctrl + x
Ctrl + ← ←
Ctrl + v
Shift + Ctrl + →
(Ctrl を離して) Shift + ←
Ctrl + x
Ctrl + → →
Ctrl + v

```

この例では文字数が少ないので再度入力した方が楽かもしれないが、コピー&ペーストやカット&ペーストが多い場合は非常に作業が楽になる。

課題 1.1.2. 以下の文章を TeXworks で作業し、課題 1.1.1 で作成した tex フォルダの中にファイル名 kadai1-1-2.txt で保存せよ。

```

abcdefg hijklm nopqrst uvwxyz ABCDEFG HIJKLM NOPQRST UVWXYZ

ABCDEFG uvwxyz ABCDEFG uvwxyz ABCDEFG uvwxyz ABCDEFG uvwxyz
HIJKLM nopqrst HIJKLM nopqrst HIJKLM nopqrst HIJKLM nopqrst
NOPQRST hijklm NOPQRST hijklm NOPQRST hijklm NOPQRST hijklm
UVWXYZ abcdefg UVWXYZ abcdefg UVWXYZ abcdefg UVWXYZ abcdefg

```

課題 1.1.3. 課題 1.1.2 で作成したテキストファイル kadai1-1-2.txt をメールに添付して mori@xmath.ous.ac.jp まで送る。なお、送信は sXXmYYYab@ous.jp のアドレスから送ること。

件名は学生番号と課題番号をこの順番で書き、

例) S23M999 課題 1.1.3.

本文は空欄でよい。

1.1.5 正規表現

正規表現を用いると検索、置換が容易に行える。詳しく説明することは避けるが、こういった方法があることを知っておけば、必要となったとき調べることが出来るであろう。

正規表現を用いて検索、置換を行う場合に特殊な文字として メタ文字というものがある。検索用と置換用では使用が異なるものが多いので、検索用の一部を紹介しておく。

検索用メタ文字（の一部）

.	任意の 1 文字
*	直前のパターンの 0 回以上繰り返し（最長一致）
+	直前のパターンの 1 回以上繰り返し（最長一致）
?	直前のパターンの 0～1 回繰り返し（最長一致）
*?	直前のパターンの 0 回以上繰り返し（最短一致）
+?	直前のパターンの 1 回以上繰り返し（最短一致）
??	直前のパターンの 0～1 回繰り返し（最短一致）
A 列 B 列	の左右の文字列のいずれか（A 列または B 列）
[文字列]	文字列のいずれか 1 文字（例：[abc] や [0-9]）
(文字列)	置換後へ継承。継承する場合は ¥1 などとする。（例 2-2）
¥	直後のメタ文字をエスケープする (直後の文字を正規表現のメタ文字として扱わないことを指定。 例えば、() や [] なら ¥(¥) や ¥[¥] のようにする)
¥n	改行文字
¥d	すべての半角数字
¥D	半角数字以外すべて
¥w	すべての半角英数字とアンダースコア
¥W	半角英数字とアンダースコア以外すべて
¥l	半角英小文字すべて
¥L	半角英小文字以外すべて（英大文字、数字、全角文字など）
¥u	半角英大文字すべて
¥U	半角英大文字以外すべて（英小文字、数字、全角文字など）
¥n	改行文字
¥s	空白（スペース）

♡ 注意! エディタによって正規表現を使える、使えないがあり、使える場合も形式が異なっていることがある。

例 1.1.2. (正規表現を用いた置換の例 1)

A regular expression, regex or regexp is, in theoretical computer science and formal language theory, a sequence of characters that define a search pattern. Usually this pattern is then used by string searching algorithms for “find” or “find and replace” operations on strings.

という文章に対して、正規表現の置換で $\$s..\s を $__$ に変換させると

A regular expression, regex $__$ regexp is, $__$ theoretical computer science and formal language theory, a sequence $__$ characters that define a search pattern. Usually this pattern $__$ then used $__$ string searching algorithms for “find” $__$ “find and replace” operations $__$ strings.

となる。これは空白と空白の間に 2 文字あるものを見つけ変換している。従って、*is*, が変換されていないことが解る。($_$ は半角スペースを意味する。)

例 1.1.3. (正規表現を用いた置換の例 2)

`one, two, three, four, five, six, seven, eight, nine, ten.`

に対して、 $\$s...([,.])$ を $__ \$1$ に変換すると

`___, ___, three, four, five, ___, seven, eight, nine, ___.`

となる。これは空白 $'\$s'$ に続く 3 文字があり、その後に , または . があるものを検索している。また、() で囲むことによって置換した時に継承している。すなわち $\$1$ は検索でヒットした文字が , なら , を継承し、 . なら . を継承して表示している。

例 1-3. 正規表現を用いた置換の例 3

$f(x,y)=6x+y-3, g(x,y)=x+y$ に対して、 $g(0,0)=-3, g(1,0)=3, g(0,1)=-2$

と入力した時点で、後半は g でなく f であることに気付いた。ここで、 $g\$((\$d), (\$d)\$)$ を $f(\$1,\$2)$ に置換すると

$f(x,y)=6x+y-3, g(x,y)=x+y$ に対して、 $f(0,0)=-3, f(1,0)=3, f(0,1)=-2$

となる。 $\$d$ は数字なので、 x や y は (すなわち $g(x,y)$ は) ヒットしない。また、() はメタ文字なので、検索するためには $\$$ が必要となる。継承するものが 2 つあるので $\$1$ と $\$2$ を使う。

数字が 2 桁以上のものを含む場合は $\$d+$ とする。したがって、上記の場合 $g(10,20)$ などはヒットしない。

1.2 ギリシャ文字、記号、文字コード

1.2.1 ギリシャ文字

ギリシャ文字は複数の書体があるが、ここでは、一般的な（数学で用いられる）文字について紹介しておく。

大文字	小文字	読み方	大文字	小文字	読み方
A	α	アルファ	N	ν	ニュー
B	β	ベータ	Ξ	ξ	クサイ, グザイ
Γ	γ	ガンマ	O	o	オミクロン
Δ	δ	デルタ	Π	π, ϖ	パイ
E	ϵ, ε	イプシロン	P	ρ, ϱ	ロー
Z	ζ	ゼータ, ツェータ	Σ	σ, ς	シグマ
H	η	イータ	T	τ	タウ
Θ	θ, ϑ	シータ	Υ	υ	ウプシロン
I	ι	イオタ	Φ	ϕ, φ	ファイ
K	κ	カッパ	X	χ	カイ
Λ	λ	ラムダ	Ψ	ψ	プサイ
M	μ	ミュー	Ω	ω	オメガ

1.2.2 記号

この講義で使われている記号について、読み方を紹介しておく。

記号	読み方	記号	読み方
$ $	パイプライン、縦棒	$*$	アスタリスク
$^$	ハット	$\{ \}$	波カッコ, 中カッコ
$/$	スラッシュ	\backslash	バックスラッシュ
$"$	ダブルクォーテーション	$.$	ドット、ピリオド
$'$	シングルクォーテーション	$,$	カンマ, コンマ
$:$	コロン	$;$	セミコロン
$-$	アンダーバー, アンダースコア	$@$	アットマーク
\cdot	ナカグロ, 中点	$!$	エクスクラメーションマーク
\sim	チルダ	$\#$	シャープ, いげた
$\$$	エンマーク	$\$$	ドルマーク

1.2.3 文字コード

コンピュータで文字を扱う場合、文字そのものを扱うのではなく、文字を表す“文字コード”を用いる。例えば、1Byte で表すことが出来るものとして、ASCII コードと呼ばれる文字コード(0 から 127 まで)があり、A が 65, B が 66, ..., Z が 90 で、a が 97, b が 98, ..., z が 122 となっている。

その他、! が 33, " が 34, # が 35 なども定義されている。

	0	1	2	3	4	5	6	7	8	9
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT
10	LF*	VT	FF*	CR	SO	SI	DLE	DC1	DC2	DC3
20	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS
30	RS	US	SP	!	"	#	\$	%	&	,
40	()	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[\]	^	_	'	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{		}	~	DEL		

1.2.4 バックスラッシュと円マークについて

文字コードの 92 にある\ (バックスラッシュ) は日本では ¥ が用いられるが、文字コードでは\ が表示される。逆に、メーラやエディタでは、¥ が\ と表示される場合がある。

Windows PC の日本語キーボードでは右 [Shift] キー周辺の [ろ] のキーに\ があるが、押すと¥となる。

♡ 注意! MacOS を使用している場合は注意が必要で、普通に入力した¥は正しく動作しない ([¥] は¥として機能しない) ことがある。

MacOS では [option] キーを押しながら [¥] キーを押して表示される\を使う。

なお、全ての MacOS と、エディタが上記の仕様となっているかは不明なので、最初に使う時に確認してみてもらいたい。

第2章

Scratch

2.1 Scratch とは

初心者が最初に正しい構文の書き方を覚えること無く結果を得られるプログラミング言語学習環境である。

2年生の講義では C 言語を用いてプログラミングを行う。プログラミングを行うとき、最初に必要なのは アルゴリズム（すなわち答えまでの手順）を考えることである^{*1}。実は、この作業が苦手な人が多いようなので、難解なプログラミングの構文やフローチャートなど意識せず、プログラミングが出来る Scratch を紹介する。

Scratch は、web 上で起動して使用する方法と、アプリをインストールして起動して使用する方法がある。Scratch のホームページ

<https://scratch.mit.edu/>

を表示すると



の画面が表示される。

^{*1} この点は、数学の証明問題と似ていると思う。

★ インターネットに接続された環境で使用する場合、このサイトの [作ってみよう] をクリックすると Scratch を使うことが可能になる。

★ インターネットに接続されていないパソコンや Android スマホで使用する場合は、アプリをインストールして行う。

上記ホームページの下段に、[リソース] の下に [ダウンロード] があるので、クリックする。



下記の画面が表示されたら適切なアプリをダウンロードし、インストールする。残念ながら、iPhone, iPad 用のアプリはない。

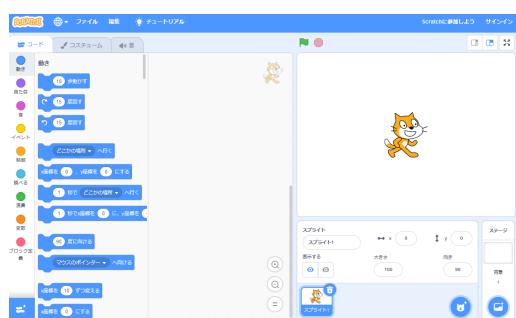


Scratch を起動すると右の画面が表示される。(チュートリアルが開く場合は、チュートリアルを閉じる)

左側はコードの一覧があり、この一覧のなかにはプログラムを作るために必要な命令(ブロック)がある。

真ん中には(ブロックを組み立てる)スクリプトエリアがあり、右側がスプライトが表示されたステージ(上段)とスプライト一覧(下段)がある。

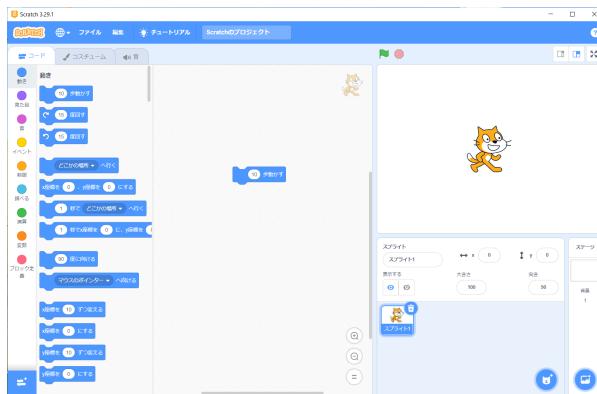
プログラムの作成は、コードにあるブロック一覧から命令や制御文をスクリプトエリアにドラッグし、繋げていく。



2.1.1 プログラム作成

実際に簡単なプログラムを作成して、動作の仕組みを理解する。

- (1) ブロックパネル内の【動き】から、 を選択し、スクリプトエリアにドラッグする。



- ドラッグした  をダブルクリックすると、実際にどのような動きか確認できる。

- (2) プログラムの開始動作を確実にするために、【イベント】から  を

スクリプトエリアにドラッグし、これら 2 つを  のように繋げる。

繋げたのち、右上の  の旗をクリックすると scratch cat が 10 歩進む。

- (3) これだけでは解りにくいので、今度は【制御】から  を選択し、スクリプトエリアにドラッグする。

そのとき、右図のように繋がるようにドラッグする。

繋げたら先ほどと同様に右上の  の旗をクリックする。今度は scratch cat が右端まで行き、止まる。

このとき、スクリプトエリア内のブロックが黄色枠で囲まれた状態になっている。これは、プログラムが動き続けていることを意味している。



のように繋げる。

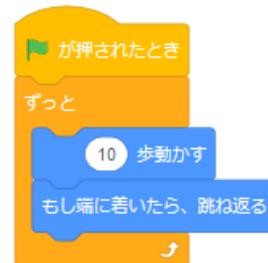


そこで、プログラムを停止するために  の●をクリックする。すると、黄色枠が無くなり停止したことを表している。

次に【動き】から  を選択し、右図のブロックの組になるようにドラッグする。

再び右上の  の旗をクリックすると、今度は右端まで行くと折り返して左端まで行き、また折り返して右端へと向かう。

ただし、右から左に移動するとき上下反転していて不自然なので、 を間に inserer。



♠ 補足 このときの折り返し方は、

[左右のみ]、[回転しない]、[自由に回転]

の3種類から選択できる。指定していない場合は、[自由に回転] になっている。

スプライトの向き(角度)を自由に変えるには  を用いる。

また、スプライトの移動する方向(角度)を変える

には、 と  の間に
 を入れる。

その後、90度の部分をクリックすると角度を変えることが出来る(右図)。

このようにScratchではブロックを繋ぎ合わせることでプログラムを作成することが出来る。

もちろんC言語やJavaのように複雑なプログラムは組めないが、アルゴリズムの学習には十分な命令(ブロック)があるので、各自で試してほしい。

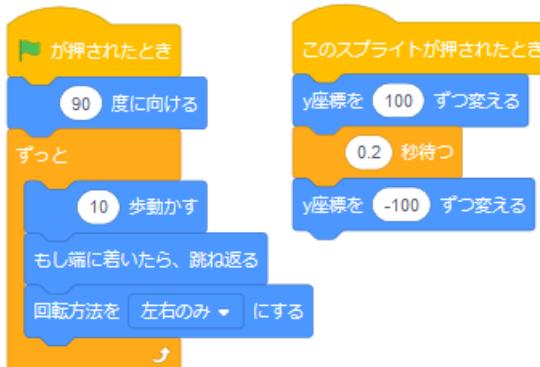


課題 2.1.1. 2つのブロック  と  をふくめたプログラムを作成せよ。

例 2.1.1. 下のプログラムは、ステージの Scratch cat を押すと（雑ではあるが）ジャンプするプログラムである。

このプログラムでは、別のブロックの組を作成し、ある条件で動作を実行するようになっている。

このように、複数のブロックの組を作成し、1つのスプライトを制御することができる。もちろん、複数のスプライトをステージに登場させることができる。



♠ **補足** ステージの座標は中心が $(0, 0)$ で、 x 軸が -240 から 240 、 y 軸が -180 から 180 のサイズになっている。ただし、スプライトはこの範囲を超えて移動する。

例 2.1.2. 途中で コスチューム を変えて移動するプログラムである。

歩数と待つ時間を空白にしているので、いろいろな数値を入力してどのように動きが変わるか確認してみる。

♠ **補足** スプライトを追加したい場合は、

画面右下にある をクリックする。

背景を変えたい場合は、右下の をクリックすると背景を選ぶことが出来る。

課題 2.1.2. 上記の例を参考に、スプライトが動くプログラムを作成せよ。

動きは自由に考えて良く、（詳しく説明していないが）背景を変えたり、スプライトを増やしても良い。



2.1.2 ファイルの保存、読み込み

作成したプログラムは、[ファイル] から [コンピューターに保存する] を押すと下の図の右図が出るので、[ファイルを保存する] を選択する (図 1)。



図 1

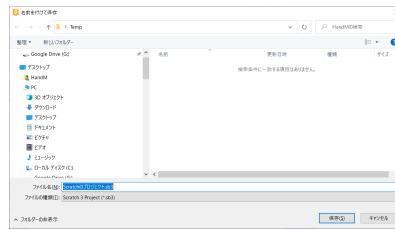


図 2

パソコンにインストールされたアプリを使っている場合は、保存先を聞かれる (図 2) ので、好みの場所に保存する。その際、ファイル名を変更してもよい。



図 3



図 4

web ブラウザ上で行っている場合、ブラウザによって処理が異なる。

ファイルの処理方法を聞かれた場合 (図 3) は、“ファイルを保存する (S)”を選ぶと、(通常) ダウンロードフォルダに保存される。

Firefox の場合、図 4 のように表示されファイルの保存が行われる。

また、スマホやタブレット (iPad 等 Windows OS 以外) でも同様に保存が出来る (場合もある)。もし上手くいかない場合は、[Scratch アカウント] を作成して保存する。

スマホの場合だけでなく、複数のパソコンで作業する場合は、[Scratch アカウント] の作成をする。まず、Scratch のホームページの [Scratch に参加しよう] をクリックして、質問に答えていく。

このとき、ユーザー名は本名や学生番号を使わないようにするように。

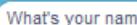
作成が完了したら、[サインイン] から、いつでもサインインしてファイルの保存、呼び出しが可能になる。

保存したプログラムを読み込む場合は、“コンピュータから読み込む”選び、保存した場所からファイルを選択する。

2.2 入出力と演算

2.2.1 入力と出力

キーボードから入力されたデータを利用するプログラムを考える。まずは、キーボードからデータを入力し、表示するプログラムを作つてみる。

キーボードからデータを入力するためには、ブロックパネル内の【調べる】にある  と聞いて待つ を使用する。このブロックを選択し、スクリプトエリアにドラッグする。

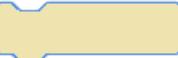
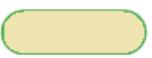
♠ 補足 ブロック内の“Wat's your name?”は環境によって、“あなたの名前は何ですか？”になっている。どちらの場合もスクリプトエリアにドラッグした後、変更できる。

入力されたデータの値は  に格納される。このブロックもスクリプトエリアにドラッグし、さらに、ブロックパネルの【見た目】から  もスクリプトエリアにドラッグする。

先にドロップした  を  の“こんにちは!”の上にドラッグし、



のようにプログラムする。このプログラムの実行結果の確認は、各自に委ねる。

♣ 補足 コードにあるブロックには、上下(もしくは一方のみ)に縫みと出っ張りのあるブロック  と、上下のどちらにも縫みも出っ張りも無いブロックがあり、さらに左右が丸まったもブロック  と、とがったブロック  に分けられる。

これらの種類は意味があり、はめ込むことが出来ない場合は利用できない(意味が通らない)ことを意味している。少し考えてみると解る(と思う)。

2.2.2 演算

四則演算

【演算】の中に、まず四則演算がある



これらは、数の四則演算と同じである。以下の例を見て、説明とする。

たとえば、右のようにブロックを組み、旗マークをクリックすると、Scratch cat が“3.50”と言ってくれる。ただし、有効桁数はあまり多くない。

今度は、前節の入力も含めたプログラミングとして、右のようにブロックを組んでみる。

“自然数を入力してください”と問われる所以、好きな自然数を入力すると3で割った値が表示される。

【演算】の中には、四則演算以外、(とがったブロックの形をした)制御の条件や、文字の演算をおこなうブロックもある。

また、簡単ではあるが三角関数なども扱える。

 を選び、“絶対値”的部分をプルダウンすると、平方根や sin, cos, tan の値を扱うことが出来る。

文字演算

Scratch では文字や文字列に対して演算を行なうことが出来る^{*2}。

【演算】にある文字の演算をおこなうブロックのうちから、ここでは両側が丸みのあるブロックとして



の3つを紹介する。

それぞれは、

^{*2} ここでいう演算は、文字や文字列に対して、ある文字、文字列または数値を返す関数と考えると解りやすい

- ・左側は文字列と文字列を併せて1つの文字列とする演算、
- ・真ん中は文字列の指定番目の文字を表す演算、
- ・右側は文字列の長さを表す演算

となっている。

使い方は、

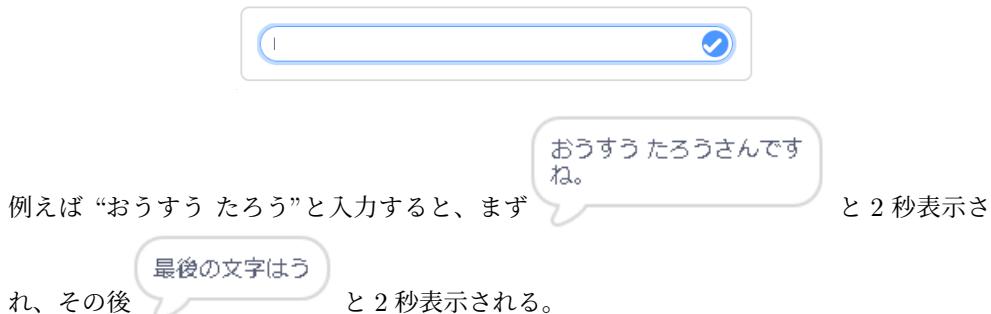


のように使う。

- ・2段目のブロックでキーボードからの入力を受け付け、入力されたデータが **答え** に格納される。
- ・3段目のブロックでは、**答え** と文字列 “さんですね。” を組み合わせて1つの文字列として、扱っている。
- ・4段目のブロックは3つとも使用していて、まず、**答え** の長さを表す演算で長さを求め、**答え** のその長さ番目の文字を表す演算で得られた文字を文字列 “最後の文字は” に併せて表示させている。

【実行の結果】

実行すると、“あなたの名前はなんですか？” と表示され、下のウインドが開く。



2.2.3 変数

上の話では1つの変数（**答え**）だけを扱っていたが、実際のプログラムを作るときは複数の変数が必要となることが多い。

ここでは複数の変数の準備（定義）を行い、それら変数を用いたプログラムを考える。

【変数】を選択すると、右の画面が表示される。

ここで、**変数を作る**を選ぶと、新しい変数名：と聞かれるので、変数名（例えば x ）を入力して **OK** を押す。

入力すると変数が作成されるので、後は **答え** と同様に扱えればよい。

これを繰り返すことによって、複数の変数を定義することが出来る。もちろん、これら定義した変数は演算を行うことが可能である。

ただし、

♡ 注意！ 演算で、数値の代入（ y の値を x に代入）

をするとき、 を用いてはいけない。

このブロックは比較のとき使うもので、変数への代入は【変数】にある



を用いて



としなければならない。

また、 x に $3y$ の値を代入するなら、演算の積を用いて



とする。

例 2.2.1. 2つの数を入力し、和と差を表示するプログラムを考える。考え方は、

- 1) 1つめの数値を聞き、入力された数値が **答え** に入る。
- 2) 2つめの数値を聞く前に、 **答え** にある数値を、変数に代入しておく。
(事前に変数を用意する。例えば、変数 **x** を作成する。)
- 3) 2つめの数値を聞き、 **答え** に入る。
- 4) **x** と **答え** の和と差を表示する。表示は **apple** と **banana** を使って、

2つの和は と **x** + **答え** **2つの差は** と **x** - **答え**

のようにする。プログラムとしては、もう1つ変数を用意した方が見た目は良い。



例 2.2.2. マウスのポインタの座標を表示し続けるプログラムを考える。

文字の和を2つ用いて、右のようにプログラムを作る。

しかし、この場合旗マークをクリックしたときのマウスのポインタがある場所の座標を表示して終わる。

表示させ続けるには、動作を継続させることを忘れないように。



2.2.4 亂数

変数は入力したものだけでは無く、生成させたものを使うことも出来る。乱数を生成させる場合は **1 から 10 までの乱数** を使う。初期では 1 から 10 までの自然数となっているが、-10 から 10 までとすることで負の数も生成する。

また、範囲を -10.0 から 10.0 とすることで、有理数の乱数を生成することができる。

課題 2.2.1. 2つの自然数を乱数で生成し、2秒ずつそれらの数を言い、その後、和と差を2秒言うプログラムを作成せよ。ただし、生成する数は 1 から 100 までとする。

2.3 制御

プログラムの流れの中で、条件によって分岐(制御)を行う代表的な2つを紹介する。

2.3.1 if 文に相当するブロック

まず、“if 文”と呼ばれる分岐方法で、

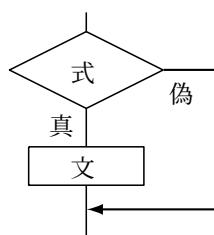


図 1

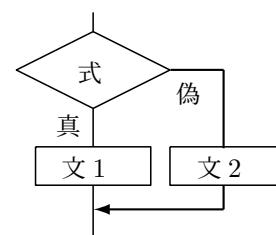


図 2

の2種類がある。フローチャートと呼ばれる図では



if 形式



if-else 形式

のように表される。

上のブロックの、条件 はフローチャートの“式”に当たる。

条件 を満たしているなら、その間ににあるブロック(フローチャートでは“文”)を実行することを意味している。

図 2 のブロックは、条件  を満たしているなら、1つ目の間にあるブロック（フローチャートでは“文 1”）を実行し、満たしていないなら、2つ目の間にあるブロック（フローチャートでは“文 2”）を実行する。

ここでいう“条件”とは [演算] にある



などである。これらの条件分岐は良く使われる所以、しっかり活用して欲しい。

実際、制御を使ったプログラムの例を見ながら考えてみる。

例 2.3.1. (分岐の例 1)

キーボードから入力されたデータが、50 より大きい数か否か判断し、50 より大きい数のときは、

“その数は 50 より大きいです”
と表示するプログラムである。

条件として、

“ が 50 より大きい”

が使われている。この条件を満たしていれば、“もし～なら……”ブロックの中にある“……”ブロックが実行される。



例 2.3.2. (分岐の例 2)

キーボードから入力されたデータが、50 より大きい数か否か判断し、50 より大きい数のときは、

“その数は 50 より大きいです”
と表示し、そうでないときは、

“その数は 50 以下です”
と表示するプログラムである。

条件として、“ が 50 より大きい”が
使われているのは、上記の例と同じである。

この条件を満たしていれば、“もし～なら…… でなければ……”ブロックの中にある“……”ブロックが実行され、条件を満たしていなければ、“……”ブロックが実行される。

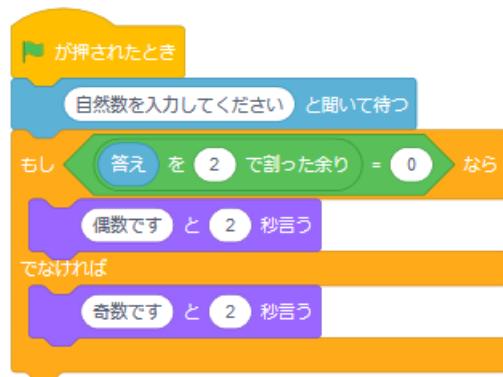
♡ 注意！ この例では、文字など数以外を入力した場合、正しく表示されない。(a, A, +, -などを入力してみる。)



例 2.3.3. (分岐の例 3)

キーボードから入力されたデータが、偶数か奇数か判断するプログラムを考える。

右のようなブロックの組になるが、3つ目のブロックの判定 (=0 となっている) 部分を =1 に変えると、“偶数ですと 2秒言う”ブロックと、“奇数ですと 2秒言う”ブロックを入れ替えても正しく動く。

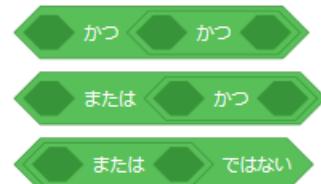


このようにプログラムを考えるとき、これが正解というものはないので、例と異なるプログラムを考えてみるのも楽しい。

条件を複数指定したい場合、例えば、右のブロックの例のような場合、組み合わせる順番に注意が必要な場合がある。

“かつ”と“かつ”を組み合わせる場合は、気にしなくとも良いが、“かつ”と“または”を組み合わせる場合、(A または B) かつ C と、A または (B かつ C) では異なる。

(偶数または 3 の倍数) かつ 5 の倍数 と 偶数または (3 の倍数かつ 5 の倍数) でみれば解る。前者は 4 や 6 は含まないが後者は含む。



2.3.2 for, while 文に相当するブロック

for 文、while 文は条件を満たしている間は、繰り返して使うことが出来る。Scratchにおいて、これらに相当するブロックは次の 2 つである。



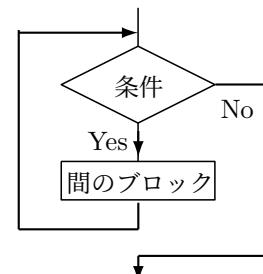
図 3



図 4

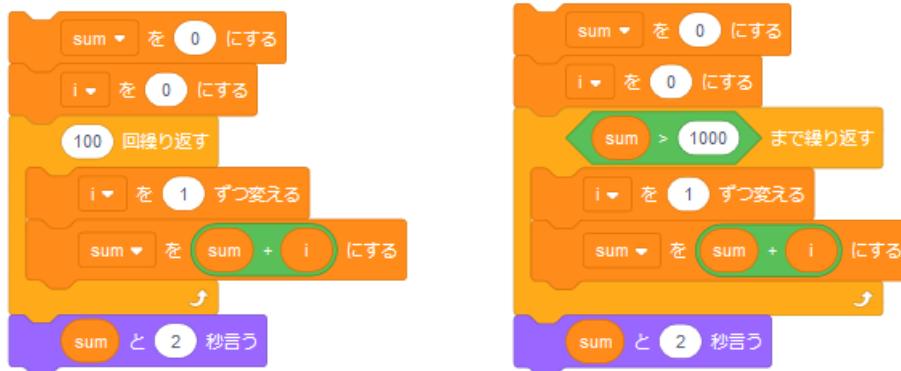
図 3 のブロックは、回数が 10 回を超えるまで間にあるブロックを繰り返し、図 4 のブロックは、条件 を満たしているなら、間にあるブロックを実行し続ける。

この 2 つのブロックの使い方の違いとして、図 3 のブロック (for 文に相当) は、条件の部分で回数が明確なときに使い、図 4 のブロック (while 文に相当) は、終了条件が明確なときに使うことが多い。



例 2.3.4. (総和の例)

1 から 100 までの和を求めるなら、左のブロック、1 から n までの和が 1000 を超えるまで求めるなら右のブロックを使う。



それぞれの場合、変数の扱いを理解していないと正しい結果を得られない。特に、**初期化** は大切で、忘れないようにする。

♠ 補足 初期化とは、変数の値を 0 や 1 などにしておくことであり、初期化していない変数は、別の値が残っていたり、ブロックを組み立て中に値が増えていったりしている。プログラムの途中で初期化をすることもある。

例 2.3.5. (初期化)

1 から 10 までの和と、1 から 10 までの積を求めるプログラムを作成する。

まず考えてみると、どちらも 1 から 10 と範囲は同じなので、まとめて計算する方法と、和と積は別々で計算する方法が思いつく。

後者は 例 2.3.4 の 1 から 100 までの和を参考に出来るので各自に委ねる。

まとめて行う場合は、変数をもう 1 つ追加する。(右にあるプログラムは変数 [sum] と [pro] を使用している。)

また、初期化は和は 0 で積は 1 となることに注意する。



例 2.3.6. (ユークリッドの互除法)

キーボードから入力された 2 つの数値の最大公約数を求めるプログラムを考える。

まずは、最大公約数の求め方の復習をする。

(ユークリッドの互除法) —————

大きい方を a 小さい方を b として以下の試行を繰り返す。

$$\begin{array}{rcl} a \div b & = & n_1 \text{ 余り } r_1 \\ b \div r_1 & = & n_2 \text{ 余り } r_2 \\ r_1 \div r_2 & = & n_3 \text{ 余り } r_3 \\ & \vdots & \\ r_{k-1} \div r_k & = & n_{k+1} \text{ 余り } 0 \end{array}$$

このとき、 r_k が最大公約数がとなる。

このアルゴリズムに従い、以下の点に注意し考える。

- ・入力された 2 つの自然数のうち大きい方を a 、小さい方を b とする。

そのため“もし～なら”ブロックを使って 1 つめに入力された数を a とし、2 つめに入力された数(答え)の大小を判定する。

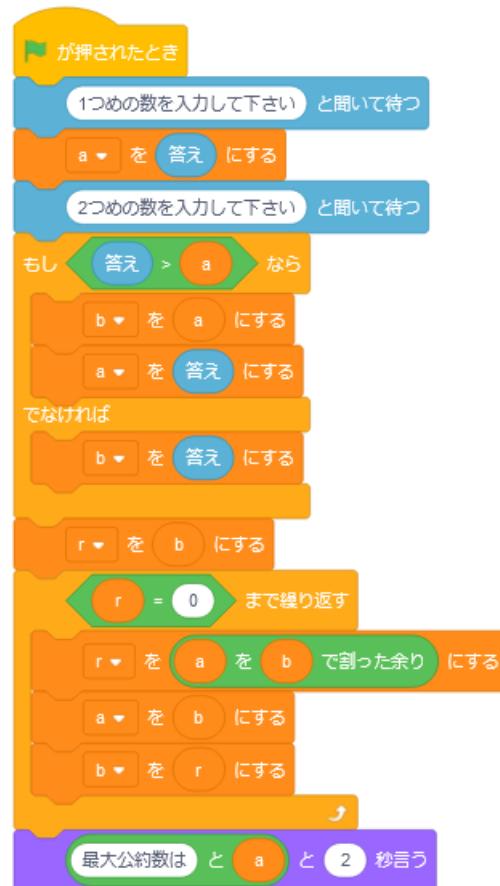
- ・1 回の演算の後、変数の入替えに注意する。

- ・終了判定は、余りが 0 となったときである。

余りの判定の段階では、余り(r)は 0 であり、変数 b の値も 0 になっている。

1 つ前の余りは、変数 a に代入されているので、答えは a となることに注意する。

以上を注意し、考えたものが右のプログラムとなる。

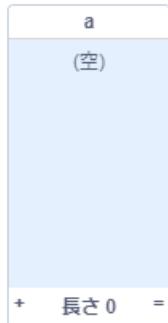


2.4 データの扱い、リストとソート

2.4.1 リスト

変数をたくさん使ったプログラムを考える場合は、リスト（配列）を使う。リストは変数を束ねたようなもので、*a, b* のようなリスト名に対して、*a* の 1 番目や、*a* の 2 番目のようにして使うことが出来る。

変数を作ったときと同じように【変数】を選択し、**リストを作る** を選ぶと、新しいリスト名 : と聞かれるので、リスト名（例えば *a*）を入力して **OK** を押す。作成されるとステージに右のようなリスト名が書かれたリストが作成される。

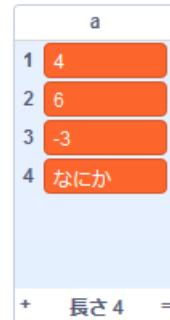


リストを扱うブロックが並んでいるのでこれまでと同様に見て貰えば解るかもしれないが、少し注意が必要なブロックもあるので、少し説明をしておく。

まず、**a ▾ の 1 番目** は、単なる変数と思うことが出来、**x** と同様に扱うこと出来る。リストにデータを追加していく場合は、**なにか を a ▾ に追加する** を使う。

また、既にあるデータを変更する場合は、**a ▾ の 1 番目を なにか で置き換える** を使って、値を代入することができる。

実際にデータを追加していくと、右のような状態になる。数値以外にも文字列も同時に扱うことができる。（1 つのリストで両方扱うことはあまり推奨はしない。）



ちなみに、これはリストの中に 3 つのデータ（4, 6, -3）が入っている状態で **なにか を a ▾ に追加する** が実行された状態である。このブロックはリストの最後に新たなものを追加するブロックである。

次に、**a ▾ の 1 番目を削除する** のブロックであるが、これはリストの(1) 番目のデータを削除し、(2) 番目以降を繰り下げるブロックである。右の例だと、1 番目が 6、2 番目が -3、3 番目が “なにか” になる。

注意が必要なのは、**a ▾ の 1 番目に なにか を挿入する** のブロックであり、これはリストの(1) 番目以降を繰り上げて、その後(1) 番目にデータを格納する。右の例だと、1 番目が “なにか”、2 番目が 4、3 番目が 6、4 番目が -3、5 番目が “なにか” になる。

また、リスト a の“長さ”が 10 のとき  は実行されない。(11) 番目に挿入はしてくれない。

もうひとつ注意が必要なものは  で、これはリスト a の中に“なにか”があるか探し、見つかればその格納番号を表してくれる。右上の例なら、格納番号 4 が値となる。また、見つからない場合は 0 が値となるが、複数見つかった場合は小さい格納番号のみが値となる。

その他  や  はそれぞれリストの内容を消すものと、リストの成分の数を扱っている。また、 は制御のときの条件として扱う。

  は、リストに格納されたデータの表示の ON/OFF を使う。

例 2.4.1. (乱数のリストへの格納)

1 から 10 までの乱数を 10 個生成し、それら乱数をリストに格納するプログラムを作成する。ここまで学習していると、右のようになることは容易に解る。



実際作成した後、旗マークを押すと、リストとして 10 個の乱数が格納されている。

ただ、注意が必要なのはこの後で、再度旗マークをクリックすると新たに 10 個の乱数が生成され、全部で 20 個になる。

新たに 10 個の乱数を生成したい場合は  を付けくわえればよい。

例 2.4.2. (約数のリストへの格納)

入力された数の約数をリストに追加するプログラムを考える。

入力された数()を 1 から順に割っていき、割り切れたとき(余りが 0 のとき)は約数なのでその数をリストに追加する。

これを  まで行う。

(1 から  までなので、 回行うことと同義である。)



例 2.4.3. (最大値、最小値)

例 2.4.1 で生成させた乱数のうち、最大値と最小値を求めるプログラムを考える。

右図は最大値 (max) と、最小値 (min) を求めるプログラムの例である。最大値と最小値を探す前に、変数 max と min をそれぞれ 1 と 10 にしていることに注意する。

表示を行う場合は、このプログラムの続きにブロックを追加する。

2.4.2 ソート

生成したり、与えられた複数のデータを、ある規則に従って並べ替えることを“ソート”といい、ソートの方法はデータの構造やデータ数によってさまざまな方法が存在する（興味のある人はネット検索で）。

いろんなソート方法を試すと良いが、どの方法でも注意しなければならないことがある。

♡ 注意！ リスト内の i 番目と j 番目を入れ替える場合、どちらかを別の変数に一度代入しておかなければならぬ。

例えば、list の 3 番目と 5 番目を入れ替えるには以下のようにしなければならぬ。



(list の 5 番目の値を tmp に退避させ、list の 5 番目に list の 3 番目の値を代入し、list の 3 番目に tmp の値を代入している。)

ここでは、アルゴリズムが簡単な、単純ソートを紹介する。

単純ソートは、リストの 1 番目 (比較元) の値と、2 番目以降 (比較対象に) ある各値を順に比較して、順序関係が条件 (大きい順、や小さい順など) の結果の逆になっているものがあれば、1 番目とそれらの値の位置を入れ替える。

リストの最後の値まで調べ終えたら、2 番目の値と、3 番目以降にある各値を比較する。これを $n - 1$ 番目と n 番目の比較まで行ったら終了となる。



例 2.4.4. リスト $\{3, 7, 2, 5, 1, 4\}$ において、数の小さい順に並べ替える手順を考える。

- | | |
|------------------------|-----------------------------|
| $\{3, 7, 2, 5, 1, 4\}$ | 1番目(比較元)と2番目(比較対象)を比較 |
| $\{3, 7, 2, 5, 1, 4\}$ | 1番目(比較元)と3番目(比較対象)を比較(入替える) |
| $\{2, 7, 3, 5, 1, 4\}$ | 新たな1番目(比較元)と4番目(比較対象)を比較 |
| $\{2, 7, 3, 5, 1, 4\}$ | 1番目(比較元)と5番目(比較対象)を比較(入替える) |
| $\{1, 7, 3, 5, 2, 4\}$ | 新たな1番目(比較元)と6番目(比較対象)を比較 |
| $\{1, 7, 3, 5, 2, 4\}$ | 1番目(比較元)との比較、入替え終了 |
| $\{1, 7, 3, 5, 2, 4\}$ | 2番目と3番目を比較(入替える) |
| $\{1, 3, 7, 5, 2, 4\}$ | 新たな2番目と4番目を比較 |
| $\{1, 3, 7, 5, 2, 4\}$ | 2番目と5番目を比較(入替える) |
| $\{1, 2, 7, 5, 3, 4\}$ | 新たな2番目と6番目を比較 |
| $\{1, 2, 7, 5, 3, 4\}$ | 2番目との比較、入替え終了 |

以下同様に繰り返し、ソートを完成させる。20個のデータの例は以下の通り。

最初の旗マークが押された
ときの後、リストの全てのデータを削除している。

1では、1から100までの乱数をリストに追加することを20回行っている。

2では、最初の比較元(i 番目)を1にしている。

比較元は19までなので、20になったら終わる(20になるまで続ける)ようにしている(3)。

そして、最初の比較対象(n 番目)は $i+1$ 番目なので、 n を $i+1$ にする(4)。

n は20番目まで比較するため、 n が20を超えるまで続けている。“ $n = 21$ まで繰り返す”でも良いが、何カ所かある

20を答えると、汎用性のあるプログラムになる(5)。

6では、 i 番目(比較元)の値が n 番目(比較対象)の値より大きい場合を見つけている。その場合、7の部分で入れ替えを行っている。

入れ替えを行った場合も行わない場合も、比較対象は次のデータに移る(8)。

比較対象が最後の値を比較した後、比較元を次のデータに移す(9)。



2.4.3 拡張機能

Scratch には最初に表示されているコード（ブロック）以外に、いくつかの拡張機能がある。それら機能は、画面左下の  から呼び出すことが出来る。



例えば、音楽を選択すると右図のブロックが追加される。これらも見るとどのような動作をするか解りやすくなっている。

その他、LEGO や micro:bit を扱えるブロックも存在する。Scratch に対応した LEGO ブロックがあると楽しさが増えるので、環境があれば試して欲しい。

ここでは、“ペン”的紹介を行う。ペンを選択すると、コード（ブロック）が追加される。これらのブロックも見てもらえば解りやすいが、2つの説明をしておく。

まず、“ペンを下ろす”と“ペンを上げる”は、Scratch ネコが移動中に、ペンを下ろして線を書き、ペンを上げると書くのを止める動作となっている。

“スタンプ”は、スプライトを張り付ける動作である。それぞれ確認するために、右のようなブロックの組（4組）を作つてみる。

スペースキーや上下の矢印キーはスマホやタブレットでは押すことが出来ないが、スマホやタブレットの場合は、それらブロックをタップすることで同様の動作となる。

例題として、正三角形や正五角形を書く。等間隔の点線を書くなど。



2.4.4 フローチャート

フローチャートとは、作業の流れを図で表したものである。

各ステップを様々な形の箱で表し、それらの間の流れを矢印（または実線）で繋ぐことでアルゴリズムやプロセスを表現する。

フローチャートを用いることで作成したいプログラムの大まかな流れを図示し、プログラムを組む際に役立てられる。また、他者へ説明する場合にもプログラムを見せるより視覚的で解りやすい。フローチャートはプログラミングのみならず、いろいろな分野で用いられている。

右のフローチャートは1から10までの和を求め、表示する流れを表したものである。

♡ 注意！ フローチャートの描き方にはいくつかの流儀があり、細かい点で異なっている。

例えば、startの直後の→のように、明らかに進む方向が解るものは実線で書く場合がある。

☆パート（様々な形の箱）

フローチャートに使われるパート（箱）のうち、よく目にするものを紹介しておく。

右の1は、手続きの始まりや終わりを意味する。

2は処理や命令

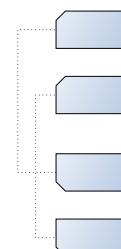
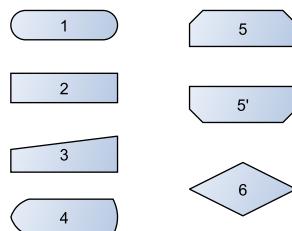
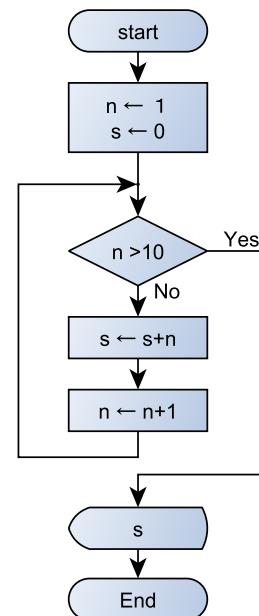
3は手動（主にキーボード）からの入力

4は画面への出力

5,5'はペアで使い、ループの開始と終了で、6は条件分岐である。

♡ 注意！ 5,5'を用いてループを考える場合、右のようなループは出来ないことに注意する。

また、6の条件分岐は成立つか否かの二択で用いることが多い。



2.5 課題

課題 2.5.1. 入力された数値が、4で割り切れる、4で割った余りが1、4で割った余りが2、4で割った余りが3か判別して2秒言うプログラムを作成せよ。

課題 2.5.2. 名字と名前をアルファベットで入力し、イニシャルを2秒表示するプログラムを作成せよ。例えば、Ridai と Taro と入力すると T.R と言う。

名前を入力してください

名前を入力してください

T.R

課題 2.5.3. 入力された2つの数値の最小公倍数を2秒言うプログラムを作成せよ。

課題 2.5.4. 数値を入力していく、0が入力されるまでに入力された数値の合計を2秒言うプログラムを作成せよ。

課題 2.5.5. 100個の乱数データを作成し、それらを順に0.5秒言うプログラムを右のように作成したが、正しく100個の乱数を表示してくれなかった。

正しく表示するようにプログラムを作り変えよ。

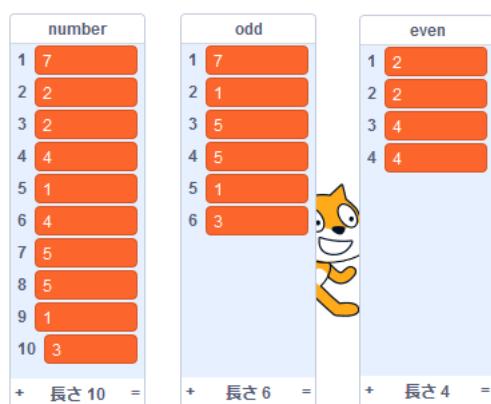


課題 2.5.6. 入力された数(≥ 2)までの素数をリストに追加するプログラムを作成せよ。

例えば12と入力すると、2,3,5,7,11がリストの上から順に登録される。

課題 2.5.7. 亂数を用いて10個の自然数(1から10まで)を生成し、偶数と奇数で分けて別のリストに格納するプログラムを作成せよ。

課題 2.5.8. 亂数を用いて10個の自然数(1から10まで)を生成し、偶数と奇数で分けて別のリストに格納し、それぞれのリストの中で数の小さい順に並べるプログラムを作成せよ。



右のリストのウインドウは課題 2.5.7 の結果である。

課題 2.5.9. 自然数を入力してくださいと聞き、**答え** に対して、**2の10乗は1024です**

2 の **答え** 乗を 2 秒言うプログラムを作成せよ。

♡ 注意! あまり大きな数にすると、 $1.245321654315e+19$ のような表示になる。これは、 $1.245321654315 \times 10^{19}$ を意味している。さらに大きい数にすると動作しなくなるので、ほどほどの数で試してみる。

課題 2.5.10. 自然数入力してくださいと 2 回聞き、それぞれを a, b とし、つるとかめの頭の数が合計 a で、足の数が合計 b 本の時、つるとかめの頭数をそれぞれ言うプログラムを作成せよ。ただし、存在しない場合はそのことを言うようにすること。

課題 2.5.11. “自然数を入力してください”と聞き、入力された値 **答え** に対して、 n を 1 から **答え** まで動くとする。 n を除く n の約数を全て足し合わせたとき、 n となる数を 2 秒ずつ言うプログラムを作成せよ。(上限は各自の好みで)

例 条件を満たす最小の数は 6 である。6 の約数は 1, 2, 3, 6 で、自身を除く約数の和は $1 + 2 + 3 = 6$ となる。

課題 2.5.12. 2 から 10000 までの数を 1 つ入力し(自然数が入力されることは仮定し)、その数が素数なら“素数”、合成数なら“合成数”と言うプログラムを作成せよ。

課題 2.5.13. 亂数を用いて、0 から 100 までの整数を 50 個生成し、それらの数の平均、最大値、最小値を言うプログラムを作成せよ。ただし、表示は一度に 3 つをまとめて言っても、平均、最大値、最小値をそれぞれ数秒ずつ言っても構わない。

平均は48.5, 最大値は92, 最小値は8

課題 2.5.14. (懸賞金 1 億 2 千万円) 自然数を入力してくださいと聞き、**答え** を n とし、以下のコラッツの予想で 1 になるまでの過程を 1 秒ずつ言うプログラムを作成せよ。

例えば、3 が入力されたとすれば

$$3 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

の順で 1 になるので、10, 5, 16, 8, 4, 2, 1 をそれぞれ 1 秒言う。

コラッツの予想

1. 自然数 n を考える。
2. n が偶数ならば n を 2 で割る。
 n が奇数なら 3 倍して 1 加える。
3. 2. の結果(を新たな n とし、 n) が 1 なら終わり、(n が) 2 以上なら再度 2. を行う。

以上の方法でどんな自然数 n も必ず 1 になる。(という予想)

第3章

LATEX

3.1 はじめに

3.1.1 TEX (テフ, テック), LATEX (ラテフ, ラテック, etc) とは

TEX とは、数式の処理に優れる組版ソフトウェアであり、その TEX を使ってもっと簡単に論文やレポートを作成出来るように開発されたのが LATEX や LATEX 2_E である。

HTML 文章と同じように、命令を用いてページを作成していく。沢山の命令があるが、基本的なものさえ覚えておけば、後は必要なときに調べればよい。

例 3.1.1. (LATEX ソースファイル)

```
%documentclass[dvipdfmx,a4j,12pt]{jarticle}
% ここがプリアンブル
\begin{document}
\textbf{第1回 集合の基礎、数学で使う記号}
\begin{flushright}
学生番号 \underline{\hspace{40mm}} 氏名 \underline{\hspace{50mm}}
\end{flushright}
\textbf{問題 1-1.} 以下の集合$A,B,C$に対して、(1) から (4) に答えよ。
(略)
\end{document}
```

TEX では細かい設定を

プリアンブル (`\documentclass` と `\begin{document}` の間)

に指定出来る。

例 3.1.2. (例 3.1.1 を変換して表示した例.)

第1回 集合の基礎、数学で使う記号

学生番号_____ 氏名_____

問題 1-1. 以下の集合 A, B, C に対して、(1) から (4) に答えよ。

(略)

最初はプリアンブルの指定は気にせずに、慣れるまでは

`\begin{document}` と `\end{document}` の間

の部分のみを編集していく。

3.1.2 実際に使用してみる

TeX は話を聞くより実際に使用した方が習得出来る。(細かいことは後述。)

TeX では作成したソースファイルを DVI ファイルや PDF ファイルに変換して、初めて文章、数式や表などが表示される。この講義(実習)では、TeXworks というランチャー兼エディタを使用して TeX のファイル作成, PDF の表示を行う。

デスクトップ上の TeXworks を起動すると、黒い画面(コマンドプロンプト)が表示されるが、少し待つと起動する。何も入力されていない新規のファイルが表示されたら、

例 3.1.3. (L^AT_EX ソースファイル)

```
\documentclass[a4j]{jarticle}
\begin{document}
吾輩は猫である。名前は真田奈衣。

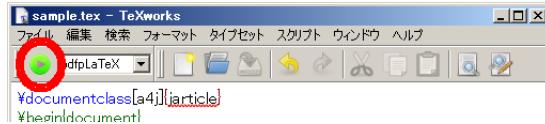
どこで生れたかとんと見当がつかぬ。
何でも薄暗いじめじめした所で
ニヤーニヤー泣いていた事だけは記憶している。
吾輩はここで始めて人間というものを見た。
\end{document}
```

と入力する。入力が終わったら、[ファイル] から [名前を付けて保存] を選び、適切なフォルダを選択する。フォルダを選んだら、ファイル名を入力する。

その際、ファイル名(フォルダ名)は 日本語, 空白入り, 記号は避けた方が良いので^{*1}、`sample.tex` や `HandM01.tex` などとする。

^{*1} 使用可能なものと不可能なものがある。なので使わない方がよい

[保存] 後、左上のアイコン（下図の丸で囲っているアイコン）をクリックする。



アイコンが に変わり、少し待ち、エラーが無ければ元に戻り、右側に pdf ファイルが表示される。

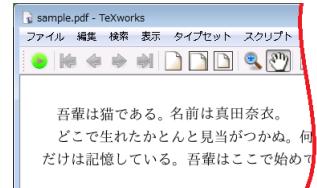
アイコンが のままならエラーメッセージが“ログの表示”に表示されている。

エラーの場合は、 をクリックして変換を中止し、エラーメッセージを元に修正し、再度アイコンをクリックする。

よく見かけるミスは、全角と半角の入力ミスと \$, {, } の数が対応していないミス。

例えば、\$ で囲むべき $y=x^2+3x$ が $y=x^2+3x$ や $y=x^2+3x \$$ となっていた場合、

```
! Missing $ inserted.
<inserted text>
                $
1.13
```



のようなエラーメッセージが“ログの表示”に表示される。

3.1.3 `\begin{document}` と `\end{document}` の間

★ 文字入力

(単純に入力した) 日本語や English はそのまま表示される。ただし、機種依存文字は使えない。例えば、全角入力した I は表示されない。I は半角大文字の I を使う。

太文字や italic type は命令が必要で、太文字なら `\textbf{太文字}`、イタリック体なら `\textit{italic type}` と入力する。

★ 記号入力

そのままでは入力できない記号もある。例えば、\$ や & などは命令文に使われるため、表示には一手間かかる。

★ 下線の入力

下線は `\underline{下線}` と入力する。さらに、`\underline{\underline{下線}}` と入力すれば、下線となる。

★ 改行

改行も一工夫必要で、例えば L^AT_EX ソースファイル内に

```
岡山県岡山市北区理大町1－1  
岡山理科大学
```

と書いても

岡山県岡山市北区理大町1－1 岡山理科大学

と表示される。改行したい場合は、空行を入れる。したがって、

```
岡山県岡山市北区理大町1－1  
岡山理科大学
```

とすると改行され

```
岡山県岡山市北区理大町1－1  
岡山理科大学
```

となる。ただし、1つ以上の改行を入れても空行は出来ない。そのため、空行を広げたい場合は `\vspace{3cm}` を使う。その際、`\vspace` の前に空行を挟むこと。

```
岡山県岡山市北区理大町1－1  
  
\vspace{3cm}  
岡山理科大学
```

★ 空白

空白を入れるには半角スペースを入れれば1つ空白が出来る。ただし、1つ以上入れても1つの空白しか入らない。例えば、半角スペースを複数入力して

```
学生番号(      ), 氏名(      )
```

と入力しても、変換結果は

学生番号(), 氏名()

と表示される。

1つの空白以上の空白が必要な場合は `\hspace{2cm}` を使う。例えば、`\hspace{2cm}` は 2cm の空白()をあけることを意味する。

また、改行後に空白が入る(この行の先頭のように)ことを避けたい場合は `\noindent` を使う。(この場合、`\noindent` の前に空行を挟むこと。)

例 3.1.4. (3 行前の文章の改行後に空白を入れない例)

の空白 (`\hspace{2cm}`) をあけることを意味する。

`\noindent`

また、改行後に空白が入る (この行の先頭のように) ことを避けたい場合は

とすると、

の空白 () をあけることを意味する。

また、改行後に空白が入る (この行の先頭のように) ことを避けたい場合は

のように表示される。

♡ 注意!

数式は数式入力モードで入力する必要があり、`$$` で囲むか数式立てを使用する。以下の☆は数式入力モードで入力が必要。

☆ 数式！

数式の変数などはそのまま `x` と入力するとフォントが文中のものになるので、数式を表す命令 (`\x`) を使うことによって、`x` と表示される。ちなみに、 x^2 は、`\x^2` と入力し、 $\frac{1}{2}$ は `\frac{1}{2}` と入力し、 π や θ は、`\pi`, `\theta` と入力する。

また数式を中央に表示する場合は `$$` で囲む (数式立てまたは別行立て数式と言う)。

例えば、 $y = x^2 + 3$ を数式立てにする場合、`$$y=x^2+3$$` とすると、

$$y = x^2 + 3$$

となる。

なお、文字と数式の扱いには注意し、`\$x+\$y$` でなく、`\$x+y$` とするように。

また、数式として繋げて書く場合、例えば $2x - \frac{1}{2}y^2$ を、`2\$x-\frac{1}{2}\$y^2` のように書くのではなく、全体を`\$`と`\$`で囲み、`\$2x-\frac{1}{2}y^2$` とすること。

☆ 分数、連分数

分数は `\frac` を用いる。例えば、 $\frac{1}{2}$ は `\frac{1}{2}` とかく。このとき、分子と分母の順番に 注意すること。連分数は、分子や分母の { } 内にまた分数を入れればよい。

例えば、 $\frac{1}{1+\frac{1}{1+\frac{1}{1+\dots}}}$ は `\frac{1}{1+\frac{1}{1+\frac{1}{1+\dots}}}` となる。

☆ 上付き、下付き

上付きは`\hat{x}`で下付きは`\bar{x}`である。例えば、`x^n` や `x_0` は、`x^{[n]}`や `x_{[0]}` である。

また、 $\{a_i\}_{i=1}^{10}$ は `\{a_i\}_{i=1}^{10}` である。

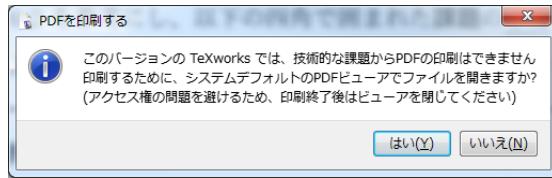
上付き、下付きは、`x_{[0]}^{[2]}`など { } 内が 1 文字なら { } を省略 (`x_{[0]}^{[2]}`) できる。

3.1.4 印刷

出来た PDF ファイルを印刷する場合は、PDF ファイルが表示されている（右側の）ウインドウの [ファイル] から、[PDF を印刷する] を選ぶ。

そのときに、右図のようなメッセージが出るので [OK] を押す。

その後、PDF ビューアー（Adobe Acrobat Reader）が開かれるので、[ファイル] から [印刷] を選択し印刷を行う。



課題 3.1.1. 例 3.1.1 を参考にし、以下の四角で囲まれた課題の中身が表示されるような LATEX ファイルを作成せよ。なお、連分数にある … は、`\cdots` を使うこと。

また、改行位置については気にしなくてよい。

課題

第2回 数式、図形の表現 I

学生番号 _____ 氏名 _____ 得点 _____

問題 1-1. 整式 $P(x) = ax^3 + bx^2 - 8x - 7$ が $x + 1$ で割り切れ、 $x - 2$ で割ったときの余りが -3 となるように、定数 a, b の値を定めよ。

問題 1-2. 次の等式を証明せよ。

$$\frac{1}{1-x} + \frac{1}{1-y} = 1 + \frac{1-xy}{(1-x)(1-y)}$$

問題 1-3. 円周率 π の連分数展開は以下であることを示せ。

$$\pi = 3 + \cfrac{1}{7 + \cfrac{1}{15 + \dots}}$$

♠ 補足 別ファイルで作成した LATEX のソースファイルを読み込むことも出来る。たとえば、`sin.tex` を別ファイルで作成し、読み込むには

```
\input{filename.tex}
```

とする。命令文の行数の多い図や表など もしくは、何度も使うような命令文は、別のファイルに保存しておくと便利である。

3.2 TeX コマンド

3.2.1 TeX のコマンドと注意事項

★ 章, 節

文章を書くとき、章の見出し、節の見出し、本文 … のようなレイアウトで書く。TeX では章や節を `\section{***}` や `\subsection{***}` で記述することによって 左上のよう、上下のスペース、フォントサイズの調整と番号付けを行ってくれる。これは、章や節の追加や削除を行ったとき、自分で番号の付け替えを行わなくて良いことを意味する。

★ 注釈

% 以降の記述は注釈として解釈され、本文に影響を与えない。

★ 地の文と命令文

TeX の命令文は % で始まることが多い。たとえば、`LATeX` と表示するには、`\LaTeX` と入力している。このとき、`\LaTeX` は命令文で、それ以外は地の文である。これを、

“このとき、`\LaTeX` は命令文で、”

と続けて書くと、% 以降どこまでが命令文で、どこからが地の文か解らなくなる。(特に英文ならなおさら。) そこで、命令文の終わりにはスペースを 1 つ以上入れるか、または命令文を { } で囲まなければならない。

♡ 注意! 後ろに空白を必要としない命令もある。例えば、`\$` や `\&` など。これらは空白を入れると \$ 30 のようにスペースが出来て不格好になる。

♠ 補足 数字やスペースは全角でなく半角を使う (方が望ましい)。

“問題 1.” と “問題 1.” では、どちらが好みか。

3.2.2 パッケージとプリアンブル

今後必要となる以下のパッケージをプリアンブルに指定する。

```
\usepackage{amsmath}
\usepackage{amssymb}
\usepackage{arydshln}
\usepackage{ascmac}
```

プリアンブルは、`\documentclass` と `\begin{document}` の間 (例 3.1.1 参照)。

これらパッケージを読み込んでおくことによって、使えるコマンドが増える。これら以外にも多数のパッケージが存在するが、同時に使えないパッケージの組み合わせなどがあるので注意が必要である。

3.2.3 T_EX の命令と種類

単独で命令となるもの

例えば、先の $\$LaTeX$ や数式モード内の命令 x^2 などは単独で命令となっている。

引数を伴い命令となるもの

分数を表す $\$frac{ }{ }$ や下線を引く $\$underline{ }$ などは { } 内にある引数に対して命令を行う。たとえば、ここに下線を引く。は、 $\$underline{ここ}$ に下線を引く。

$\$begin$ と $\$end$ で命令となるもの

例 3.1.1 にあった右寄せの $\$begin{flushright}$ ~ $\$end{flushright}$ や、中央寄せの $\$begin{center}$ ~ $\$end{center}$ などがそうである。

☆ 数式 II

$\$$ で囲む以外に数式立てする方法として eqnarray 環境 (equation 環境) を使用する方法がある。eqnarray 環境は複数行の場合に使用し、equation 環境は單一行の場合を使う。どちらも式番号を付けられる。

例 3.2.1. (eqnarray 環境の例)

```
 $\$begin{eqnarray}$ 
y&=&x^2 - 4x + 3 $\$%$     % =の前後に & を入れて、以降の=を揃える。
&=&(x-3)(x-1)
\$end{eqnarray}
```

とすると、

$$y = x^2 - 4x + 3 \quad (3.1)$$

$$= (x - 3)(x - 1) \quad (3.2)$$

となる。各式の右側に付く式番号は自動的に通し番号になる。式番号をつけたくない場合は $\$begin{ }$, $\$end{ }$ の eqnarray の最後に*を付けて eqnarray*とする。

```
 $\$begin{eqnarray*}$ 
y&=&x^2 - 4x + 3 $\$%$ 
&=&(x-2)^2-1
\$end{eqnarray*}
```

$$\begin{aligned} y &= x^2 - 4x + 3 \\ &= (x - 2)^2 - 1 \end{aligned}$$

♡ 注意!

以下は数式モードか、数式立てもしくは数式 II で上げた命令内で入力すること。

☆ カッコ

カッコのうち、(), [] はそのまま表示される。{ } は `\{ \}` と入力する。 $(\frac{\sqrt{2}}{2}, 1)$ のようなとき、() の高さを調節するにはそれぞれの()の前に `\left.` または `\right.` を付けて `\left(\right)` と入力すると、 $\left(\frac{\sqrt{2}}{2}, 1\right)$ のように調整してくれる。

☆ 三角関数、指数関数、対数関数、n乗根

`sin`, `cos`, `tan` は `\sin`, `\cos`, `\tan` と書き、`exp` も `\exp` と書く。このとき `\` を忘れる `sin` や `exp` となる。`ex` と書く場合は、`e` の書体を `e` とするため、`\mathrm{e}^x` と書く。

平方根は `\sqrt` を使う。例えば、 $\sqrt{x^2 + y^2}$ は `\sqrt{x^2 + y^2}` と書き、 n 乗根は `\sqrt[n]{\text{数式}}` を使い、 $\sqrt[3]{-8}$ は `\sqrt[3]{-8}` と書く。

☆ 総和、総乗

総和と総乗はそれぞれ `\sum_{条件}{式}`, `\prod_{条件}{式}` を使う。

たとえば、 $c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$ は `c_{ij}=\sum_{k=1}^n a_{ik} b_{kj}` と書く。同じ命令文でも \$\$ で囲めば文中 (数式モード) と異なり、以下のようない表示となる。

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

☆ 極限、微分、積分

極限は `\lim_{x \rightarrow 0} f(x)` と書くと、 $\lim_{x \rightarrow 0} f(x)$ と表示される。微分は `f'(x)` と入力して `f'(x)` と表示したり、分数を用いて `\frac{\mathrm{d}y}{\mathrm{d}x}` と表示させたりする。

積分は `\int_{下限}^{上限} f(x) dx` を使う。 $\int_0^1 f(x) dx$ ならば、`\int_0^1 f(x) dx` と入力する。これらは、数式立てにすると

$$\lim_{x \rightarrow 0} f(x), \frac{dy}{dx}, \int_0^1 f(x) dx$$

と表示される。ちなみに、 ∞ は `\infty` と書く。

☆ ベクトル、行列

ベクトルは `\vec{a}` を使い、`\vec{a}` と書くと \vec{a} と表示される。ただし、文字が多いときは \vec{AB} では寂しいので、`\overrightarrow{AB}` と書いて \overrightarrow{AB} と表示させる。

行列の表示にはいくつかの方法があるが、汎用性の高い array 環境を使う^{*2}。使用方法は以下の例を参考するとよい。

array 環境使用例

出力	コマンド	コメント
$\begin{array}{ c c } \hline a & b \\ \hline c & d \\ \hline \end{array}$	$\left \begin{array}{cc} a & b \\ c & d \end{array} \right $ <pre>\left \begin{array}{cc} a & b \\ c & d \end{array} \right </pre>	1行目の{cc}の2つの cは中央(<i>center</i>)で 2列の配列を意味する。 ¥¥は改行、行の終わり
$\left(\begin{array}{cc c} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \end{array} \right)$	$\left(\begin{array}{cc c} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \end{array} \right)$ <pre>\left(\begin{array}{cc c} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \end{array} \right)</pre>	{cc:c}は2列目と 3列目の間に点線を 入れることを表す。
$\begin{array}{ c c } \hline & a & b \\ \hline c & ac & bc \\ \hline de & ade & bde \\ \hline \end{array}$	$\left \begin{array}{cc} & a & b \\ c & ac & bc \\ de & ade & bde \end{array} \right $ <pre>\left \begin{array}{cc} & a & b \\ c & ac & bc \\ de & ade & bde \end{array} \right </pre>	{ c r1 }は2列目と 3列目の間以外に縦線 を入れることを表し、 1列目は中央、2列目は 右、3列目は左に揃えて いる。¥hlineは横線を 入れる。
$\left[\begin{array}{ccc} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{array} \right]$	$\left[\begin{array}{ccc} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{array} \right]$ <pre>\left[\begin{array}{ccc} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{array} \right]</pre>	…, :, …の出力は ¥cdots, ¥vdots, ¥ddotsを使う。

◇ 余談 行列や行列式を扱う場合は pmatrix や vmatrix を使うことが多い。汎用性はない。

★ 枠, 表

表の作成には array 環境以外に table 環境がある。こちらの使用は数式環境ではない。

```
\begin{table}[htb]
\begin{tabular}{|l|c|c||r|} \hline
単位 & 評価 & 点数 & GPA \\ \hline \hline
認定 & S & 100~90 & 4 \\
& A & 89~80 & 3 \\
& B & 79~70 & 2 \\
& C & 69~60 & 1 \\
不認定 & D & 59~0 & 0 \\
& E & 未資格 & 0 \\ \hline \hline
\end{tabular}
\end{table}
```

単位	評価	点数	GPA
認定	S	100~90	4
	A	89~80	3
	B	79~70	2
	C	69~60	1
不認定	D	59~0	0
	E	未資格	0

ここではこの table 環境について紹介のみに留め、詳しく触れないことにする。

^{*2} この array 環境も数式モード、数式立てで使う。

☆ その他 (ギリシャ文字)

ギリシャ文字は、数式モード、数式立てで使用すること。

大文字	コマンド	小文字	コマンド	大文字	コマンド	小文字	コマンド
A	A	α	$\$alpha$	N	N	ν	$\$nu$
B	B	β	$\$beta$	Ξ	$\$Xi$	ζ	$\$xi$
Γ	$\$Gamma$	γ	$\$gamma$	O	O	\circ	\circ
Δ	$\$Delta$	δ	$\$delta$	Π	$\$Pi$	π	$\$pi$
E	E	ϵ	$\$epsilon$	P	P	ρ	$\$rho$
Z	Z	ζ	$\$zeta$	Σ	$\$Sigma$	σ	$\$sigma$
H	H	η	$\$eta$	T	T	τ	$\$tau$
Θ	$\$Theta$	θ	$\$theta$	Υ	$\$Upsilon$	v	$\$upsilon$
I	I	ι	$\$iota$	Φ	$\$Phi$	ϕ	$\$phi$
K	K	κ	$\$kappa$	X	X	χ	$\$chi$
Λ	$\$Lambda$	λ	$\$lambda$	Ψ	$\$Psi$	ψ	$\$psi$
M	M	μ	$\$mu$	Ω	$\$Omega$	ω	$\$omega$

一部のギリシャ文字の小文字には変体文字がある。

小文字	変体文字	コマンド	小文字	変体文字	コマンド
ϵ	ε	$\$varepsilon$	ρ	ϱ	$\$varrho$
θ	ϑ	$\$vartheta$	σ	ς	$\$varsigma$
π	ϖ	$\$varpi$	ϕ	φ	$\$varphi$

☆ その他 (記号 1)

以下の記号は、数式モード、数式立てで使用すること。

出力	コマンド	出力	コマンド	出力	コマンド
\leq	$\$le$	\because	$\$because$	\leqq	$\$leqq$
\geq	$\$geqq$	\geq	$\$ge$	\therefore	$\$therefore$
\pm	$\$pm$	\times	$\$times$	\in	$\$in$
\ni	$\$ni$	\supseteq	$\$supseteq$	\subseteq	$\$subseteq$
\cap	$\$cap$	\cup	$\$cup$	\subset	$\$subset$
\supset	$\$supset$	\uparrow	$\$uparrow$	\downarrow	$\$downarrow$
\rightarrow	$\$rightarrow$	\leftarrow	$\$leftarrow$	$\uparrow\uparrow$	$\$Uparrow$
\Downarrow	$\$Downarrow$	\Rightarrow	$\$Rightarrow$	\Leftarrow	$\$Leftarrow$
\rightarrowtail	$\$to$	\rightarrowtail	$\$mapsto$	\neq	$\$ne$
\Leftrightarrow	$\$Leftrightarrow$	\circ	$\$circ$	\bullet	$\$bullet$
\oplus	$\$oplus$	\otimes	$\$otimes$	\emptyset	$\$emptyset$
\bot	$\$bot$	\angle	$\$angle$	\triangle	$\$triangle$

★ その他(記号 2)

以下の記号は通常の入力で使用するものである(数式モード内で使用可否を含む)。

出力	コマンド	出力	コマンド	出力	コマンド	出力	コマンド
#	\#	\$	\\$	%	\%	&	\&
-	_	{	\{	}	\}	\\$	\\$
\text{£}	\pounds	\oe	\oe	\emptyset	\o	\emptyset	\o
TeX	\TeX	L ^A T _E X	\LaTeX	L ^A T _E X 2 _ε	\LaTeXe	(C)	\o

★ フォントサイズ

フォントのサイズは { }で囲み、\tiny, \scriptsize, \footnotesize, \small, \normalsize(標準), \large, \Large, \LARGE, \huge, \Huge で指定する。例えば、{\Large OUS}とすればOUSとなる。それぞれのサイズは以下のようになる。

ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC, ABC

★ 囲み

複数行の文字や数式などを囲む場合には、以下のコマンドを使う。このとき、[l] は“イチ”ではなく“エル”で、left を意味している。

タイトルを付ける場合

```
\begin{itembox}[l]{タイトルを付ける場合}
内容
\end{itembox}
```

または

```
\begin{screen}
内容
\end{screen}
```

のように、タイトルを付ける場合と付けない場合で分かれる。また、screen を shadebox もしくは boxnote に変更すると枠の形状が変わる。

shadebox にした場合



boxnote にした場合

3.3 図形と画像

3.3.1 TeX 図形

TikZ

TeXで図を描くには picture 環境や METAPOST などが使われてきた。この講義ではより高度なグラフを描くことができる TikZ(ていくす)を用いる。いろいろな例を見ながら説明する。ちなみに、TikZ のマニュアルは

<http://ftp.yz.yamagata-u.ac.jp/pub/CTAN/graphics/pgf/base/doc/>

に、`pgfmanual.pdf`というファイルで置いてあるが、1200 ページを超える量 (9.7MB) があり、目的のコマンドを探すのは難しいかもしれない。そこで、

<https://www.ctan.org/pkg/visualtikz>

にある `The document itself` を見るのが良いかもしれない。こちらは、コマンドと結果が表示されている。

簡単な図形を書く場合

簡単な直線や円を書く場合は、座標や半径が解れば、それらをもとに書くことができる。

例 3.3.1. (TikZ の例 1)

```
$documentclass[dvipdfmx,a4j]{jarticle}
$usepackage{tikz}
$begin{document}
$\tikz\draw (0,0)--(0,2)--(1,0.5)--(2,2)--(2,0);
$\tikz\draw (0,0) circle (1);
$end{document}
```

この実行結果は 右図が描かれる。

TikZ を用いて図を作成する場合、まずプリアンプルに

`\usepackage{tikz}`

を追加する (上の例の 2 行目)。そして、図を描きたい場所で コマンドを入力する。

この例ではまず、`\tikz\draw` のコマンドで、座標 $(0,0), (0,2), (1,0), (2,2), (2,0)$ を結ぶ線を描いている。次に、`\tikz\draw` のコマンドで、 $(0,0)$ を中心とする半径 1 の円を描いている。

$(0,0)$ の位置が 2 つの図で異なるのは、それぞれの図に異なる図形領域が確保されるからである。重ねて描きたい場合は、複数の図を 1 つの領域に描く場合を利用する。



例 3.3.2. (TikZ の例 2)

```
$documentclass[dvipdfmx,a4j]{jarticle}
$usepackage{tikz}
$begin{document}
$tikz$draw[line width=2pt,rounded corners=12pt,->]
(0,0)--(1,1)--(2,0)--(3,1)--(4,0);
$end{document}
```

この実行結果は 右図が描かれる。

(4 行目) 線の幅 (line width) を 2pt、角を 12pt
分丸くし、線の終わりを矢印 -> にしている。

線の両端 (一方) を矢印にする方法は、-> 以外に、<-，<->，->> などがある。



☆ 複数の図を 1 つの領域に描く場合

\$draw コマンドなどを \$begin{tikzpicture}\$ と \$end{tikzpicture}\$ で囲む。その際に、例で挙げていた

\$tikz\$draw

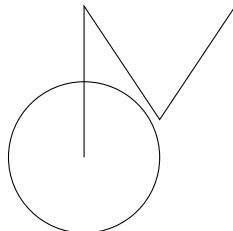
の

\$tikz

は省く。例 3.3.1 をまとめて描くと次の例となる。

例 3.3.3. (TikZ の例 3)

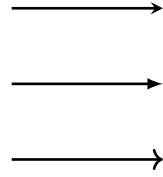
```
$documentclass[dvipdfmx,a4j]{jarticle}
$usepackage{tikz}
$begin{document}
$begin{tikzpicture}
$draw (0,0)--(0,2)--(1,0.5)--(2,2)--(2,0);
$draw (0,0) circle (1);
$end{tikzpicture}
$end{document}
```



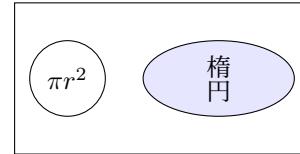
この場合、2 つの図の座標 (0,0) が一致していることが解る。

例 3.3.4. 以下は TikZ 部分のみ (左) と、それを含んだ実行結果 (右) を示している。

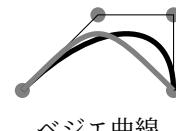
```
\$begin{tikzpicture}[x=20mm,y=20mm,line width=1pt]
\$draw[->] (0,1)--(1,1);
\$draw[-latex] (0,1.5)--(1,1.5);
\$draw[-stealth] (0,2)--(1,2);
\$end{tikzpicture}
```



```
\$begin{tikzpicture}
\$draw(0,0) circle(0.5) node {\$\pi r^2\$};
\$fill[blue!10](2,0) ellipse(1 and 0.5);
\$draw(2,0) ellipse(1 and 0.5);
\$draw(2,0) node {\$\text{Hbox\{state 楕円\}}\$};
\$draw(-0.7,-1) rectangle(3.1,1);
\$end{tikzpicture}
```



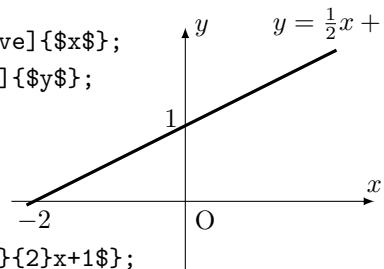
```
\$begin{tikzpicture}[x=1mm,y=1mm]
\$fill[gray] (0,0) circle (1)
(10,10) circle (1)
(20,10) circle (1)
(20,0) circle (1);
\$draw(0,0)--(10,10)--(20,10)--(20,0);
\$draw[line width=2pt] (0,0)..controls (10,10) and (20,10)..(20,0);
\$draw[line width=2pt,gray] (0,0)..controls (10,10)..(20,0);
\$draw(10,-5) node {\$\text{Hbox\{ベジエ曲線\}}\$};
\$end{tikzpicture}
```



```
\$begin{tikzpicture}[x=1mm,y=1mm,line width=2pt]
\$draw (2,2) circle (2);
\$draw (2,4)--(6,4)--(6,0) .. controls (6,3) and (7,4) ..
(12,4)--(9,0)--(12,0);
\$end{tikzpicture}
```

ORZ

```
\$begin{tikzpicture}
\$draw[-latex] (-2.3,0)--(2.5,0)node[above]{\$x\$};
\$draw[-latex] (0,-1)--(0,2.3)node[right]{\$y\$};
\$draw (0,0)node[below right]{\$O\$};
\$draw (-2,0)node[below]{\$-2\$};
\$draw (-0.4,1.1)node[right]{\$1\$};
\$draw[very thick,domain=-2.1:2]
plot(\$x,\$x/2+1)node[right]{\$y=\frac{1}{2}x+1\$};
\$end{tikzpicture}
```



☆ 繰り返し

同じような線を沢山描きたい場合は、`\foreach` が便利である。

```
\foreach #変数 in {a,b,...,c} object
```

これは、変数を `a` から `c` まで、増分 `b-a` で変化させ、`object` を実行させるコマンドとなる。

例 3.3.5. (`foreach` の使用例)

```
\begin{tikzpicture}[x=1mm,y=1mm]
\draw (0,0)--(100,0);
\foreach \x in {0,...,100} \draw(\x,0)--(\x,3);
\foreach \x in {0,5,...,100} \draw(\x,0)--(\x,5);
\foreach \x in {0,10,...,100} \draw(\x,0)--(\x,7);
\end{tikzpicture}
```

この実行結果は 下図が描かれる。



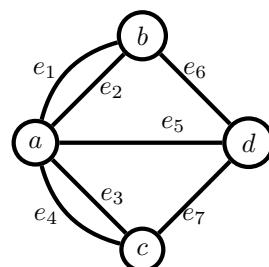
3 行目は変数 `x` を 0 から 100 まで変化させ、 $(x, 0)$ から $(x, 3)$ まで直線を描いている。

4 行目は変数 `x` を 0 から 100 まで増分 5 で変化させ、 $(x, 0)$ から $(x, 5)$ まで直線を描いている。

5 行目は変数 `x` を 0 から 100 まで増分 10 で変化させ、 $(x, 0)$ から $(x, 7)$ まで直線を描いている。

例 3.3.6. 同様に、次のような図も描ける。

2		7						6
	9		4			8		
		5	3	6	1			
		9				4		1
5	2		8	1			3	7
1			4	8	7	6		
9	5	2		4	8	7		3
3	6		7			1	2	
8			6	2	3	5	9	4



3.3.2 画像ファイルの貼り付け



文中に画像を張り付けたい場合(例えば のように)、

`\includegraphics[オプション]{ファイル名}`

を使う。

[オプション] には

`bb` : 画像サイズの情報 , `width` : 表示幅 , `height` : 表示高さ

`scale` : 拡大縮小率 , `angle` : 回転角 , `clip` : はみ出しを切り取る

などがある。オプションを複数使う場合は、, で区切る。

`bb` の指定方法は、画面左下の座標、右上の座標で指定する。例えば、画像サイズが 640×480 の画像なら、

`[bb=0 0 640 480]`

とする。この講義では必ず指定すること。

表示幅、表示高さは縦横の比率を変えて拡大や縮小したい場合や表示サイズを指定したい場合 など に使用する。例えば、画像サイズが 640×480 の画像を縦方向だけ半分にしたい場合は、

`[bb=0 0 640 480, width=640pt,height=240pt]`

とする。縦横の比を保って拡大(縮小)したい場合は `scale` を使う。例えば 30% にしたい場合は、

`[bb=0 0 640 480, scale=0.3]`

とする。

画像を回転させたい場合は `angle` (と `origin`) を使う。`angle=n` で回転角 n を指定し(反時計方向回りを +)、回転の中心は `origin=oz` で指定する。`oz` は図の 中心 [c], 左上 [tl], 右上 [tr], 左下 [bl], 右下 [br] のいずれかの ([] 内の) 値を入れる。[デフォルトは bl]

ファイル名の場所には、ファイル形式が EPS, BMP, JPEG などの画像のファイル名を書く。



`\includegraphics[bb=0 0 228 236, angle=30,origin=c]{tanQ.bmp}`



また、レポートや論文などで画像の位置を自動的に調節して表示させる場合は、画像を貼り付けたい 辺りに

```
\begin{figure}[位置指定]
\includegraphics[オプション]{ファイル名}
\end{figure}
```

と入力する。“辺り”と言ったのは、画像の貼り付け場所を [位置指定] で以下の 4 つから指定するためである。複数書く場合、書く順は関係なく、優先順位は h,t,b,p の順である。

h : 記述した場所に表を出力 , t : ページの上端に表を出力
b : ページの下端に表を出力 , p : 表専用のページを用意して出力

例 3.3.7. たとえば、画像サイズが 640x480 の画像を 30% に縮小し、そのページの上端に張り付ける場合のコマンドが以下となる。

```
\begin{figure}[t]
\includegraphics[bb=0 0 640 480,scale=0.3]{ous.bmp}
\end{figure}
```

実際にはこの場所にこの (↑) コマンドを書いているが、表示されるのはこのページの上端になっている。

また貼り付けるための画像として、TikZ にはいろいろな素材が用意されている。例えば....。



```
\begin{tikzpicture}[gray]
\node[name=a,shape=bob,minimum size=1.5cm] {};
\node[ellipse callout, draw, callout absolute pointer={(a.mouth)}]
at (3,.5) {我々も TikZ だよ。};
\node[name=b,alice,minimum size=1.5cm,mirrored] at (8,0) {};
\node[ellipse callout, draw, callout absolute pointer={(b.mouth)}]
at (5.8,.-0.3) {すごいね!};
\end{tikzpicture}
```

3.4 プrezentーション

3.4.1 Beamer とは

TEXでプレゼンテーションを行う場合、以前は slides や seminar が使われていた。

最近では、powerdot や Beamer などのクラスがよく使われている。ここでは、Beamer を使用した方法の説明を行う。まずは例を述べ、その後個別に説明を行う。

例 3.4.1. (Beamer の使用例)

説明のために、行頭に行番号を付けているが、実際は入力しない。

```

1 \$documentclass[dvipdfmx,cjk]{beamer}
2 \$usettheme{Antibes}
3 \$title{Beamer によるプレゼンテーション}
4 \$author{応用 数学}
5 \$institute[岡山理科大学]
6
7 \$begin{document}
8 \$begin{frame}{}}
9 \$titlepage
10 \$end{frame}
11 \$begin{frame}{まずは}
12 最初のページ。
13 \$end{frame}
14 \$begin{frame}[t]{その次}
15 2 ページ目
16 \$end{frame}
17 \$end{document}
```

まず、1 行目はいつもの呪文。

2 行目はメイン・テーマの指定。省略可能で、省略すると default となる。

プリアンブルには メイン・テーマの他にカラー、フォント、インナー、アウターのテーマも指定できる。

3,4,5 行目は発表タイトル、著者名、所属等である。

6 行目 (`\$begin{document}` の前) に空行が無い場合、エラーとなる。

8~10 行がタイトル、11~13 行が 1 ページ目、14~16 行が 2 ページ目となる。

11 行目や 14 行目の `{frame}` の後の `{ }` 内は各ページのタイトルを示す。また、14 行目の `{frame}` の後の `[t]` は、フレーム内のどの位置 [top, center, bottom] (上詰め、中央、下詰め) に表示するかを指定している。省略すると `[c]` となる。



3.4.2 Beamer のコマンド

★ frame

Beamer では、1つの画面を frame と呼び、`\begin{frame}` と `\end{frame}` の区切りで 1 つの frame を作る。(基本的に) この frame 内を 1 ページとして表示される。frame 環境の中では、通常の L^AT_EX のコマンドを使用する。例えば、以下のような frame を考える。

```
\begin{frame}[options]{はじめに}
```

ここには、通常の L^AT_EX のコマンドを使用することができる。

.....

```
\end{frame}
```

この options にはさまざまなオプションが用意されているが、t,c,b を良く使う。

★ ブロック

L^AT_EX で普通に使用していた囲みも使えるが、Beamer ではブロックが使える。

基本的なブロック囲みは `\begin{block}` と `\end{block}` で囲む。

例 3.4.2. (block の例)

```
\begin{block}{タイトル}  
基本的なブロック  
\end{block}
```

タイトル
基本的なブロック

また、`\begin{block}`, `\end{block}` の block を `exampleblock`, `alertblock` に変えると以下のようになる。

◎ `\begin{exampleblock}`{ }～`\end{exampleblock}`

exampleblock
例えば、例につかうブロック

◎ `\begin{alertblock}`{ }～`\end{alertblock}`

alertblock
例えば、注意につかうブロック

★ 一時停止, オーバーレイ

1つのフレーム内で、表示を一時停止する場合に `\$pause` を使う。スペースキーを押すことによって、`\$pause` 以下を表示する。

(一時停止の例) —

```
\$begin{frame}{一時停止}  
まず、この部分は表示される。  
\$pause  
この部分は、スペースキーが押されるまで表示されない。  
\$end{frame}
```

`\$pause` よりも表示の順番を細かく設定する方法として、`itemize` を使用する。次の例にしめすように、表示させたいフレーム番号を書く形で指定する。

例 3.4.3. (オーバーレイの例. 説明のため行番号付き)

```
1 \$begin{itemize}  
2 \$item<1-> その 1  
3 \$item<2-3> その 2  
4 \$item<3> その 3  
5 \$item<4> 2,3 を消して、その 4  
6 \$item<1,4> その 1 とその 4 のとき表示  
7 \$end{itemize}
```

1つのフレームの中に、`\$begin{itemize}` と `\$end{itemize}` を入れると、このフレームの中にサブフレームが出来るイメージになる。

例 3.4.3 では 4 枚のサブフレームが出来ていて、

2 行目の `\$item<1->` に続く “その 1” は 1 フレーム目以降、表示される。

3 行目の `\$item<2-3>` に続く文字は 2 フレーム目と 3 フレーム目に表示される。

4 行目の `\$item<3>` に続く文字は 3 フレーム目のみ表示される。

5 行目の `\$item<4>` に続く文字は 4 フレーム目のみ表示される。

6 行目の `\$item<1,4>` に続く文字は 1 フレーム目と 4 フレーム目に表示される。

★ 色付け

LATEX と同様に文字に色付けをすることが出来るが、Beamer では標準で red, blue, green, cyan, magenta, yellow, black, darkgray, gray, lightgray, orange, violet, purple, brown などがある。

使用方法は `\$color{上記の色} 色付けしたい文字` であり、例えば

```
\$color{red}赤色の文字
```

とする。

3.4.3 二段組

ページを分割し、二段組にすることが出来る。試験問題などで使われることがある。

二段組をする場合は、プリアンブルに `\usepackage{multicol}` を書き、二段組をしたい場所に `\begin{multicols}{2}`³ と `\end{multicols}` を入れる。その中に、まず左側の内容を入力し、左側が終わったら `\columnbreak` を書き、右側に移る。

変換後の pdf ファイルでは、真ん中に線が引かれ左右に分けて書かれる。

例 3.4.4. (二段組みの例)

以下の問いに答えよ。

```
\begin{multicols}{2}
{\bf 問題 1}. 64MiB の容量を....  

....  

\columnbreak  

{\bf 問題 2}. 次の数式を....  

....  

\end{multicols}
```

表現とメディアの数理 定期試験

学生番号_____ 氏名_____

以下の問いに答えよ。

問題 1. 64MiB の容量をもつ USB メモリに、日本語が何文字保存できるか答えよ。
ただし、メディアフォーマット等に必要な領域は 0 とする。

問題 2. 次の数式を表示させる L^AT_EX コマンドを答えよ。

$$(1) \sum_{i=1}^{10} i = 55$$

$$(2) \int a^x dx = \frac{a^x}{\log a} + C$$

この命令文の前に次のコマンドによって、左右の間隔や区切り線の太さを指定できる。

`\setlength{\columnsep}{10.0pt}` で左右コラムの間隔の指定

`\setlength{\columnseprule}{0.4pt}` で区切り線の太さの指定

♠ 補足 なお、1 ページ全体や 1 つのソース全体を二段組にする場合は、`\twocolumn` ~ `\onecolumn` を用いたり、`twocolumn` クラスを使う。

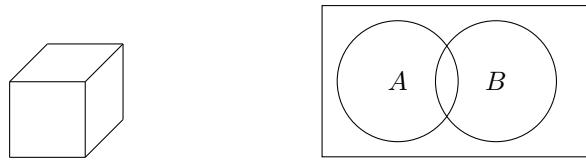
³ この 2 を 3 になると三段組みになる。

3.4.4 L^AT_EX まとめ課題

課題 3.4.1. A4 用紙に次項にある課題の枠囲みの中が outputされるように TeX ソースファイルを作成せよ。ただし、最後の S00M000 氏名 は各自の学生番号、氏名に変えること。

また、文章の改行の位置が異なるかもしれないが、そのままで構わない。

課題 3.4.2. 以下の図が出力となるように TeX ソースファイルを作成せよ。ただし、横に並べる必要はなく、大きさや線の太さもそれぞれ異なって構わない。



課題 3.4.3. 好きな図を考え、その図が出力となるように課題 3.4.2 の TeX ソースファイルの続きを継ぎに作成せよ。その際、作成した 3 つの図が 1 枚の A4 用紙にまとまるようにすること。

課題 3.4.4. (1) 例 3.4.1 を参考に、プレゼンテーション用の L^AT_EX ソースファイルを作成せよ。

1 ページ目はタイトルがくること。

2 ページ目には大学生活や卒業研究、教職についてなど何かコメントを上詰めで書くこと。

3 ページ目には itemize を用いて、1,2,3,4,5,6,7 に対して、1 回目は 1 のみ、2 回目は偶数のみ、3 回目は奇数のみ、4 回目は素数のみ、5 回目は全てが表示されるページを作ること。

4 ページ目にはブロックを用いて何か定義か定理を挿入すること。

(2) (1) で作成したものに対して、メイン・テーマ(例 3.4.1 の 2 行目)をネットで調べ、Antibes 以外のものに変更し、\\$author{応用 数学}の“応用 数学”は各自の学生番号、氏名に変更せよ。

課題

課題プリント

三角関数の加法定理

$$\sin(\alpha + \beta) = \sin \alpha \cos \beta + \cos \alpha \sin \beta \quad (1)$$

$$\sin(\alpha - \beta) = \sin \alpha \cos \beta - \cos \alpha \sin \beta \quad (2)$$

$$\cos(\alpha + \beta) = \cos \alpha \cos \beta - \sin \alpha \sin \beta \quad (3)$$

$$\cos(\alpha - \beta) = \cos \alpha \cos \beta + \sin \alpha \sin \beta \quad (4)$$

定義 1.1 f を開区間 I 上の関数とする。 f が $x_0 \in I$ で 微分可能であるとは

$$\lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

が有限値として存在するときをいう。

定理 1.1

$\{f_n\}_{n=1}^{\infty}$ を $[a, b]$ 上の関数列で、 $[a, b]$ 上の関数に一様収束しており、各 f_n は有界可積分関数であるとする。このとき、極限関数も有界可積分であって

$$\lim_{n \rightarrow \infty} \int_a^b f_n(x) dx = \int_a^b \lim_{n \rightarrow \infty} f_n(x) dx$$

が成り立つ。とくにベキ級数 $\sum_{n=0}^{\infty} a_n(x - x_0)^n$ に対しては、その収束区間ににおいて

$$\int \sum_{n=0}^{\infty} a_n(x - x_0)^n dx = \sum_{n=0}^{\infty} \frac{a_n}{n+1} (x - x_0)^{n+1} + C$$

が成立する。

定義 1.2 (正則変換、正則行列)

1 次変換 f

$$f : \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

において、 $ad - bc \neq 0$ のとき、 f は逆変換を持ち、その逆変換 f^{-1} は

$$f^{-1} : \begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}$$

となる。このように表現行列に逆変換が存在する 1 次変換 f を 正則変換といい、正則変換の表現行列を 正則行列といいう。

以上の作成を LATEX 2 ε で行いました。S00M000 氏名

第4章

Geogebra

GeoGebra とは、GeoGebra = Geometry (幾何) + Algebra (代数) を意味し、対話的なグラフィックスで直感的に操作が出来、代数、幾何、解析を統合的に扱える数学ソフトウェアである。

4.1 GeoGebra の起動と入力

4.1.1 インストール版と web 版

GeoGebra を起動するには、web 上で行う方法と、アプリをインストールして起動する方法がある。インストールされた環境では、 で起動させられる。

GeoGebra のホームページ <https://www.geogebra.org/> を表示すると



の画面 (左: パソコン、右: iPhone) が表示される。このまま「アプリの開始」をクリックすると GeoGebra を使うことが可能になる。ただし、作成したファイルを保存する場合に [GeoGebra アカウント] が必要になる (作成方法は後述)。

自分のパソコンや iPhone に作成したファイルを保存させたい場合は、アプリのインストールを行うことを推奨する。

パソコンの場合は、左画面の [アプリのダウンロード] から、[全機能統合版・クラシック 6] を選択してインストールを行う。

iPhone、Android スマホではストアからインストールを行う。

[GeoGebra アカウント] の作成をする場合は、GeoGebra のホームページの [アプリ開始] から、開いたウィンドウの右上の [ログイン] を押すと、



上左図の画面が表示され、[アカウントを作成] を選ぶと、上右図が表示される。

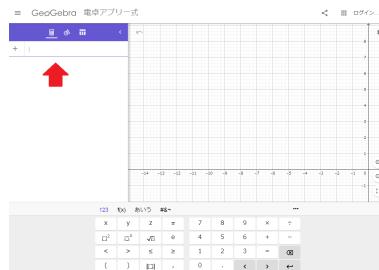
メールアドレスとユーザー名を入力し、パスワードを作成する。後、14歳以上であることのチェックをする。

このとき、ユーザー名は面倒で、既に使われているユーザー名は使えない。ドットやスペースは使えないが、数字は使える。

4.1.2 GeoGebra の入力方法

GeoGebra に関数、グラフ、図形等を入力する方法は大きく分けて 2通りある。1つは、キーボードから式をテキストで入力する方法、もう1つは、作画ツールを用いてマウスやタッチで入力する方法である。

これ以降、パソコンで [アプリ開始] から始めた場合を例に紹介する。大きく異なる場合にのみ、個別の注釈を入れる。



上図の画面、左側にカーソルの点滅した入力フィールド (赤矢印) がある。ここに $y=2x$ と入力すると、グラフが右側の作画領域に表示される。その他、座標、方程式、コマンドや関数を直接入力することが出来る。

★ 数式入力

和、差の $+$, $-$ はそのまま入力すれば良いが、 x^2 は x^2 と入力し、以下 x^3 は x^3 と入力する。また、掛け算 $a \times b$ は $a*b$ で、割算 $\frac{a}{b}$ は a/b と入力する。

下付は $_$ を使い、 $\log_3 x$ は、 $\log_3 x$ のように入力する。

実際の入力では少し注意が必要で、例えば $y = x^2 + 3x + 1$ を入力するとき、 $y=x^2+3x+1$ と入力すると、 $y = x^{2+3x+1}$ となってしまう。この場合、 $y=x^2\rightarrow+3x+1$ と入力する。

(\rightarrow はキーボードの→キー)

同様に、 $\frac{1}{2}x$ はキーボードから $1/2\rightarrow x$ と入力し、 $\log_3 x$ は、 $\log_3\rightarrow x$ と入力する。

♠ 注意! 数式を入力するとき、上付き、下付き、分数など一区切り終えたら \rightarrow を押す。

♠ 注意! 数と変数の掛け算の場合、*は省略可能である ($2a$ など)。しかし、 ab は ab といい変数か、 $a \times b$ か解らないため、省略できない。

♠ 注意! コマンドの大文字、小文字は区別されないが、変数は区別されるので注意。

例) $A=(1,1)$ は点 $A(1,1)$ となり、 $a=(1,1)$ はベクトル $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ となる。

★ テキスト入力の補助 1

テキスト入力の補助として、画面下の入力キーボードがある。



このキーボードを用いて、数式や関数を入力することが出来る。例えば、 $y = x^2 + 3x + 1$ を入力する場合は、



と入力します。

また、 $y = \sqrt{x+1}$ は、テキスト入力の場合 $y=sqrt(x+1)$ と入力が必要であるが、画面下の入力キーボードでは



と入力することで出来る。

さらに、 $y = |x + 1|$ は、テキスト入力の場合 $y=\text{abs}(x+1)$ と入力が必要であるが、画面下の入力キーボードでは



と入力することで出来る。

♠ 注意! $y = |x + 1| + 1$ を画面下キーボードで入力する場合は



としなければ、ならない。

このキーボードには、いろいろな関数の入力の補助や論理演算子などの入力も可能となっている。



高校までの数学を学習していると表示を見ると想像がつくと思うので詳細はここでは省略する。論理演算については、使用するときに説明を行う。

入力練習 以下の関数を入力してみる。

$$(i) \ y = \sin x + \cos x \quad (ii) \ y = \frac{\sin x}{x}$$

$$(iii) \ y = \frac{x^2 + 1}{x - 1} \quad (iv) \ y = \frac{1}{2} \log_{10} x$$

$$(v) \ \frac{x^2}{4} - \frac{y^2}{9} = 1 \quad (vi) \ (x^2 + y^2 - 1)^3 = x^2 y^3$$

例 4.1.1. 3次関数 $f(x) = x^3 - 3x^2 - x + 1$ に対して $f(x) = 0$ の解、 $y = f(x)$ の極値と変曲点を求める。

入力フィールドに以下を入力する (# 以降はコメントである。入力しない)。

```
f(x)=x^3-3x^2-x+1
r=root(f)                                # 根
s=extremum(f)                            # 極値
t=inflectionpoint(f)                      # 変曲点
```

3次関数のグラフと、 x 軸との交点、極大、極小の点、変曲点が表示される。

この `root(f)` は $\sqrt{f(x)}$ ではなく、 $f(x) = 0$ を満たす x の値(根)という意味である。

★ テキスト入力の補助 2

コマンドを入力中に、入力候補がいくつか表示される補完機能がある。長いコマンドの場合、途中まで入力し、この補完機能を使うとよい。

例えば `s=extremum(f)` を入力している途中で、候補が表示される。この中から目的のものを選択すれば、そのコマンドの続きを打つ必要がなくなる。`()` なども補完されるので、自分で入力するとミスになることがある。少し慣れが必要となる。

♠ 補足 いくつかの関数で、 x 軸との交点、極大、極小の点、変曲点を求めたいと思った場合、続けて同じように入力するのではなく、新たに

$$f(x) = x^3 - 3x^2$$

と入力する。3次関数のグラフが変わり、それに伴って各点も変化する。

このように、他のオブジェクトに従属するオブジェクトは、主のオブジェクトを変更すると自動的に変更される。

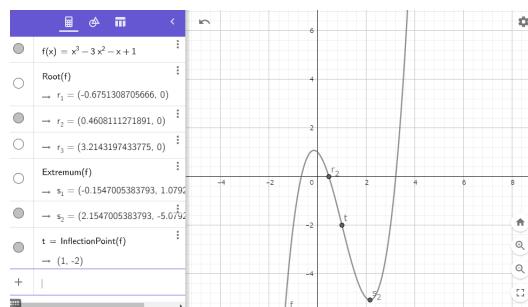
♠ 補足 マウスのホイールを使うとズームイン、ズームアウトが可能である。また、ドラッグすることで座標軸をスライドさせられる。(ただし、後述の作画ツールを使用している場合、“移動”にしなければ、スライドできない)

♠ 補足 グラフが沢山表示されて見辛くなったら、入力した履歴から表示させたくないオブジェクトの○印をクリックすると表示の ON/OFF が切り替えられる。



左が ON、右が OFF.

例 非表示とする例



上図の場合、根の点 r_1, r_3 と、極値の s_2 の点を非表示としている。

例題. 以下の不等式を解け。

$$|x - 1| + |x + 3| \leq 5$$

絶対値が2つもあり、嫌がる問題かもしれません。GeoGebraを用いると目で見て答えを導くことが出来る。

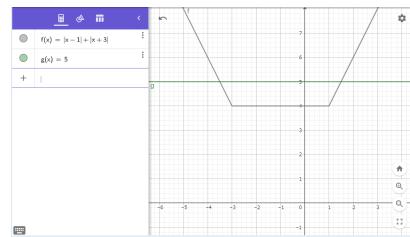
新たなファイルを作成し、入力フィールドに

$$\begin{aligned} f(x) &= \text{abs}(x-1) + \text{abs}(x+3) \\ g(x) &= 5 \end{aligned}$$

と入力する。

ここで、 $y = f(x)$ のグラフ(不等式の左辺)が、 $y = g(x)$ のグラフ(不等式の右辺)より下となる場所を探す。

実際グラフを見ると、答が -3.5 から 1.5 の範囲だと解る(ズームインも使う)。



$$\text{答え } -3.5 \leq x \leq 1.5$$

♡ point もちろん、目視で見ているため正確な答えとは言えないが、自分の答えの確認や、習い始めた生徒に教えるときのイメージとして使うことは出来る。

課題 4.1.1. GeoGebraを用いて、次の2つのグラフの交点を観察し、その値を答えよ。

$$(1) y = \sqrt{2x+2}, y = -x+3$$

$$(2) y = \left| |x| - 2 \right|, y = \sqrt{|x^2 - 4x| + 4}$$

$$(3) y = 3 \sin x, y = \frac{x^2}{2}$$

課題 4.1.2. 次の2次関数のグラフを GeoGebra を用いて描き、最大値または最小値の観察した値を答えよ。またそのときの x の値も観察し、答えよ。

$$(1) y = 2x^2 + 8x + 7$$

$$(2) y = -x^2 + 2x + 1$$

課題 4.1.3. 次の関数のグラフを GeoGebra を用いて描き、最大値と最小値の観察した値を答えよ。またそのときの x の値も観察し、答えよ。

$$(1) y = -2x^2 - 4x + 1 \quad (-2 \leq x \leq 1)$$

$$(2) y = x^2 - 8x + 12 \quad (5 \leq x \leq 6)$$

$$(3) y = x \cdot \sin x \quad (0 \leq x \leq 6)$$

4.2 GeoGebra 作画ツール群を用いた入力の方法

4.2.1 作画ツール

作画ツールは、マウスやタッチ操作によって図形を簡単に描くことが出来る。それらツール群は、入力フィールドの上にある  の真ん中のマーク  を押すとでてくる。

右にそれらツール群の簡易版を載せている。この一番下にある、[もっと他のツール] を押すと^{*1}、さらに多くのツール群の表示となる。1つ1つ説明する必要はなく、アイコンを見てもらえば内容はが解る。

例 4.2.1. (三角形の作図)

簡易版作画ツールが表示されている場合は一番下の [もっと他のツール] を押して、スクロールさせると、中ほどに



がある。この左端の [多角形] を選択し、作画領域内で異なる場所を 3ヶ所クリックし、3点 A, B, C を作成する。その後、三角形を完成させるため最初の場所(点 A)をクリックする。点 A と異なる点をクリックし、点 D が作画されたら、一度多角形を完成させ、その後削除する。



から [削除] を選び多角形をクリックすると削除される。ただし、点は残るので、点もクリックして削除していく。三角形や辺の色や辺の太さは、図を右クリックして設定から変えられる。

また、正三角形(正多角形)を書きたい場合は、別のツールがあるので、そちらを使用する。

^{*1} スマホにアプリをインストールした場合は、別アプリで GeoGebra 幾何が必要となる

課題 4.2.1. 三角形を 1 つ描き、その三角形の 5 心の作図をせよ。

ユークリッド空間内の 三角形の 5 心

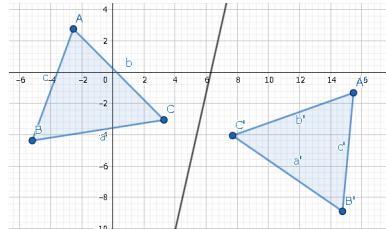
- ・外心 三角形の 3 辺の垂直二等分線は 1 点で交わる。
- ・重心 三角形の辺の中点と向かい側の頂点を結ぶ線分 (3 つ) は 1 点で交わる。
- ・垂心 三角形の 3 つの頂点から対辺に下ろした垂線は 1 点で交わる。
- ・内心 三角形の 3 つの内角の二等分線の交点は 1 点で交わる。
- ・傍心 三角形の 1 つの内角と他の 2 つの外角の二等分線 (3 つ) は 1 点で交わる。

例 3.2.2. (直線に関する折り返し)

ツール群の [基本ツール] にある、[点] を選び、相異なる 2 点 D, E を作図する。次に、ツール群の [線] にある、[2 点を通る直線] を選び、先ほど作成した点 D, E をクリックする。

今度は、ツール群の [移動・変換] にある、[直線に関する鏡映] を選択し、三角形をクリックした後に直線をクリックする。上手く行けば、直線に関する折り返された三角形が表示される。この後、ツール群の [基本ツール] から [移動] を選択し、点 A,B,C,D,E を移動させてみる。

点対称移動の場合は、ツール群の [移動・変換] にある [角度を指定して点の回りに] を選び、三角形をクリックした後に点 D をクリックし、出てきたウィンドウの角度を好みの値にして [OK] を押す。



4.2.2 真偽値演算子

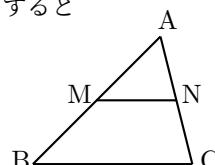
入力キーボードの中に見慣れない演算子がある。そこで、次の定理の確認を例に、紹介する。

中点連結定理

$\triangle ABC$ の 2 辺 AB, AC の中点をそれぞれ M, N とすると

$$MN \parallel BC, \quad MN = \frac{1}{2}BC$$

が成り立つ。



今回は座標を気にしないので、作画領域を右クリックし、右図のようなメニューを開く。そして、軸の表示のチェックを外し、グリッドを表示からグリッドなしを選択。

スマホの場合は、右上の歯車ボタンから選ぶ。

<input checked="" type="checkbox"/>	軸を表示	✓
<input type="checkbox"/>	グリッドを表示	✗
<input checked="" type="checkbox"/>	グリッドに寄せる	✓
<input type="checkbox"/>	すべての残像の消去	✗
<input type="checkbox"/>	フィットするようにズーム	✗
<input type="checkbox"/>	設定	✗

まず、先ほどと同じように三角形を作図する。三角形が出来たら、辺 AB, AC の中点を作画するためにツール群の [作図] にある、[中点または中心] をえらぶ。その後、辺 AB をクリックすると中点 D が作画される。同様に辺 AC をクリックすると、中点 E が作図される。

中点 D, E が作図されたら、作画ツール群の [線] から [2 点を結ぶ線分] を選び、点 D と E をクリックして線分 DE を作図する。

次は、入力フィールドを使うので、ツール群の上にある  をクリックして入力フィールドを表示させる。

このとき、作画ツールで描いた図の代数的値が表示されている。

画面下の入力キーボードが表示されていない場合は、画面の下辺りに  があるので、このアイコンをクリックする。

入力キーボードが表示されたら、上段にある  を選択する。

右図のように、辺 BC が a 、辺 DE が f と表示されていると仮定して（異なる場合は読み替えて）、入力フィールドに

$a \parallel f$

と入力し、エンターキーを押す。このとき、画面表示は、 \parallel が $\|$ に変わるが、正しい。

♡ point この  は、平行を意味する真偽値演算子であり、上で入力したものは、

“ a と f が平行ですか？”

という意味である。入力すると、下に “ $\rightarrow \text{true}$ ” と表示されたはずである。これは線分 a と f が平行なとき true と表示され、平行でないとき false と表示される。

次に、入力フィールドに

$a \stackrel{?}{=} 2*f$

と入力し、エンターキーを押す。この  は、等しいか否かの真偽値演算子であり、左辺と右辺が等しければ true 等しくなければ false と表示される。

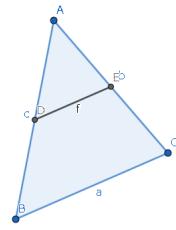
どちらも true と表示されたら、作画ツール群に移り、[基本ツール] の [移動] を選び、再び入力フィールドを表示させる。そして、点 A, B, C の 1 点をドラッグすると三角形の形が変わるが、平行と等しい真偽値演算子の値は常に true を表示している。

以上で、どのような三角形でも中点連結定理が成り立つことが確認できた。

課題 4.2.2. 三角形を描き、チェバの定理、メネラウスの定理が成り立つことを確認せよ。

（ヒント）辺 AB 上に点 P 、辺 BC 上に点 Q をとる。次に、辺 CP と辺 AQ の交点を K とするとき、直線 BK と辺 BC の交点を R とする。

線分の長さを公式の形（分数の形）で表し、1 に等しいか確かめる。



4.2.3 定規とコンパスによる作図

定規とコンパスによる作図とは

定規とコンパスだけを有限回使って図形を描くことを指す。(Wikipedia より)

作画ツール群にはさまざまなツールがあるが、定規とコンパスによる作図問題で使えないものがある。ツール群の中には、[中点または中心] や、[垂線]、[3 点を通る円] などがあるが、すべてが定規とコンパスだけを用いて、1 作業で行えるとは限らない。

1 作業で行うこととは、

[点]、[2 つのオブジェクトの交点]、[2 点を結ぶ線分, 直線, 半直線]

などである。

また、[多角形] は可能であるが、[正多角形] は不可能である。作画ツール群 [円] に関しては、[中心と円周上の 1 点で決まる]、[コンパス]、[中心と半径で決まる円] が該当する。

例 3.2.3. 正五角形を定規とコンパスで作図する方法で作図せよ。

まず、2 点 A,B を作図し、線分 AB を 1 辺 (長さ a) とする正五角形を作図する。

円の書き方を工夫し、円が多くなり見難くなったら、不要になったオブジェの表示を OFF にして見やすくする。

- (1) 線分 AB の垂直二等分線 l を引き、AB との交点を C とする。
- (2) 点 C を中心に半径 AB の円^{*2}を描き、直線 l との交点の一方を D とする。^{*3}
- (3) 2 点 A,D を通る直線 m を引き、点 D を中心に AB の半分の長さ ($a/2$) の円を描き、直線 m との 2 つの交点に対し、ADE の順となるように交点 E をとる。
- (4) 点 A を中心に半径 AE の円を描き、直線 l との交点を F とする。
- (5) 点 F が正五角形の一番上の頂点であるが、点 F, 点 A, 点 B を中心に半径 AB の円 $O(F)$, $O(A)$, $O(B)$ を描き、円 $O(F)$, $O(A)$ の交点を G とし、円 $O(F)$, $O(B)$ の交点を H とする。

注) G,H はそれぞれ 2 点あるが、五角形 ABHFG が正五角形になるように選ぶ。

以上の作図で五角形 ABHFG は正五角形となる。ちなみに、点 A または B を移動させても、正五角形は保たれている。

課題 4.2.3. 定規とコンパスで作図する方法で、三角形の 5 心を作図せよ。

^{*2} 線分 AB を作成し、長さが a だとする。“中心と半径で決まる円”を選び、中心を選んだ後 半径を a と入力。

^{*3} これ以降の点はすべて直線 AB に対し D 側に点を取るとする。

4.3 スライダー

4.3.1 軌跡

GeoGebra では軌跡を表すことが出来る。そこで次の例題を考える。

例題. (軌跡) —

点 A(4,4) に対して、点 P が円 $x^2 + y^2 = 4$ の周上を動くとき、次の点の軌跡を求めよ。(東京書籍 数学 II)

- (1) 線分 AP の中点 M
- (2) 線分 AP を 1:2 に内分する点 Q

ここでは、問題文の点や円を正しく入力するために入力フィールドを使う。

まず、入力フィールドに

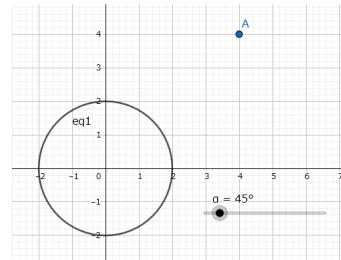
$A=(4,4)$

$x^2+y^2=4$

と入力する。A は必ず、大文字にすること。

次に、作画ツール群の [基本ツール] から [スライダー] を選択する。選択後、作画領域内をクリックするとスライダーの設定画面が表示される。

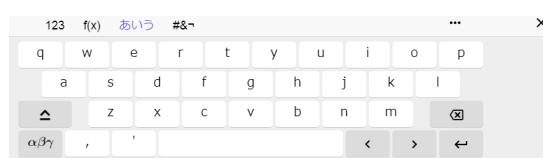
ここで、“数値”に入っているチェックを“角度”に変えて“OK”を押す。



次に、入力フィールドに戻り、

$$P=(2\cos(\alpha), 2\sin(\alpha))$$

と入力する。この α は、画面下の入力キーボード



から、上段にある“あいう”を選び、左下の $\alpha\beta\gamma$ を選択する。そして、 α を入力する。

続けて

```
segment(A,P) # 線分
M=midpoint(A,P) # 中点
Q=2*A/3+P/3 # AP を 1:2 に内分する点
```

と入力する(分数表記の入力には注意)。

入力が終わったら、入力フィールドを表示する画面にもどり、



の右端、再生ボタンを押してみる。アニメーションされるが、この動きだけでは、観察しづらいので、入力フィールドの上に表示された、 $M = \text{Midpoint}(A, P)$ の右端の

$M = \text{Midpoint}(A, P)$:

をクリックして、[設定]から[基本]の、残像を表示にチェックを入れる。これにより、残像が表示されるので、観察しやすくなる。(Qも同様)

課題 4.3.1. 3次関数の係数の変化による、関数のグラフの変化を観察するために、スライダーを3つ(a, b, cを)作成する。

次に、入力バーに

$$f(x)=x^3+a*x^2+b*x+c$$

と入力する。

入力後、スライダーの値を変化させてみて、グラフの形が変わることを確認する。

その後、入力フィールドに

```
s=root[f]
t=extremum[f]
u=inflectionpoint[f]
```

を追加する。(例 4.1.1. 参照)

場合によっては、値が存在せず、未定義と表示されることもある。

以上の準備を踏まえて、以下の(1)～(3)を予想せよ。

(1) 3次関数 $y = x^3 + ax^2 + bx + c$ の極値が存在するのは、係数(a, b, c)がどのような場合か。

(2) 3次関数 $y = x^3 + ax^2 + bx + c$ の変曲点がy軸上に存在するのは、係数(a, b, c)がどのような場合か。

(3) 3次関数 $y = x^3 + ax^2 + bx + c$ の極大値、極小値が存在し、極大値となる点が第2象限、極小値となる点が第4象限に存在するのは係数(a, b, c)がどのような場合か。

4.3.2 ある区間での最大値最小値

例題. (最大値、最小値) —

a を実数とする。関数 $f(x) = x^3 - 3x + 1$ の区間 $a \leq x \leq a + 1$ のおける最大値と最小値を求めるよ。

まず、関数を入力フィールドに書く。

$y=x^3-3*x+1$

次に、先ほどと同様にスライダーを作成する。今回は、“数値”的まで作成する。スライダーが出来たら、入力フィールドに戻り、

$a \leq x \leq a + 1$

と入力する。このとき、 \leq は入力キーボードを用いて入力する。

ここまで出来たら、アニメーションをさせてみる。色の変わっている領域内の最大値と最小値を観察することが出来る。

こちらも観察は少ししづらいので、次のコマンドを入力する。

`Max(f,a,a+1)`

`Min(f,a,a+1)`

これらは、それぞれ、関数 $f(x)$ の区間 $a \leq x \leq a + 1$ での最大値、最小値をとる点を表すコマンドである。

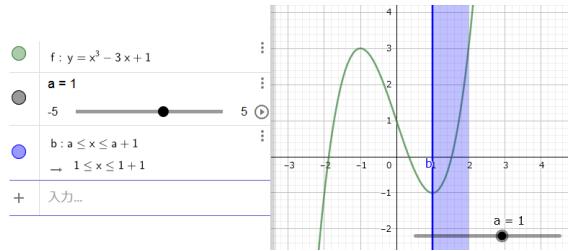
これにより、区間内での最大値と最小値をとる関数上の点が表示される (最大値と最小値は、 y 成分を見ればよい)。

♡ 注意! `Max`, `Min` にはバグがある。

`f(x)=x^3-3*x+1`

`Max(f,1,2)`

と入力すると、 $(1.6180339887499, -1.2360679774998)$ のような値が表示される。これは、この区間で最大値となる点が 2箇所あり、処理が正しく行われないと予想される。この他にもバグがありそうなので、このようなソフトを使う場合は全てを信じるのではなく、知識をもって使う必要がある。



4.3.3 関数の接線

関数 $f(x) = x^3 - 3x + 1$ のグラフ、そのグラフ上の点における接線を表示する。

まず、スライダーを作成し、次に

```
f(x)=x^3-3*x+1
A=(a,f(a))                                # グラフ f 上の点
t=tangent(A,f)                            # 点 A における f の接線
s=slope(t)                                 # t の傾き
```

と入力フィールドに入力する。

ここで、スライダーをアニメーションさせると、関数 $f(x) = x^3 - 3x + 1$ の各点における接線の傾き（の値 s ）が表示される。各点での傾きを見るため、

```
B=(x(A),s)                                # x(A) は点 A の x 座標の値
```

と入力フィールドに入力し、先ほどと同様にスライダーをアニメーションさせ、点 A を動かしてみる。

B の動きを見るには、B に残像をつければ解りやすい。この B の軌跡は、もちろん……。

課題 4.3.2. 次の軌跡を表示させ、それぞれの方程式、座標を目視で考察し解答せよ。

(1) 点 A(-2, 0), B(2, 0) と円 $x^2 + y^2 = 4$ がある。点 C がこの円上を動くとき、3 点 A, B, C が三角形を成したときの重心の軌跡を表示し、軌跡の方程式のおおよそを答えよ。

(2) 放物線 $y = x^2 + 2(a-2)x + a$ の頂点の軌跡を表示し、その軌跡の描く曲線と x 軸、 y 軸との交点を予想し答えよ。

課題 4.3.3. 次のパラメーター表示（媒介変数表示）された関数のグラフを、点の軌跡を用いて表示させよ。

(1) $x = \sin \theta + 2, y = \cos \theta - 1$. ただし、 θ はパラメーター。

(2) $x = \theta - \sin \theta, y = 1 - \cos \theta$. ただし、 θ はパラメーター。

(3) $x = \frac{2(1-t^2)}{1+t^2}, y = \frac{2t}{1+t^2}$. ただし、 t はパラメーター。

(4) $x = \sin(a\theta), y = \sin(b\theta)$. ただし、 θ はパラメーター、 a, b は任意の自然数。

発展課題. 点 P(2, 1) から曲線 C : $y = x^3 + 2x^2 - 2x - 1$ に引ける接線の本数を観察せよ。

また、点 P を P(2, l) とした場合、曲線 C に引ける接線の本数を観察せよ。

同様に、点 P を P(k , l) とした場合も観察せよ。

4.4 空間図形

4.4.1 点の作図

空間図形を扱う場合は、GeoGebra を起動したあと、画面の右上にあるメニュー（図 1 の赤○）から空間図形を選択する。

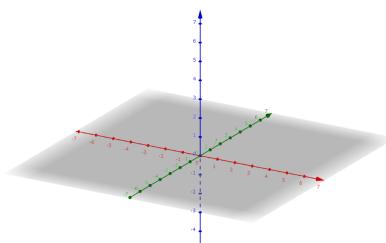
スマホの場合は、新たにアプリをインストールする必要がある。GeoGebra を起動後メニュー（図 2 の赤□）から空間図形を選択してインストールを行う。

それぞれの場合、起動させると下図のような画面が表示される。



図 1

図 2



空間図形では、作画ツール群に空間内の図形の作画ツールや、それら図形に対する変換のツールが追加されている。

まずは、点の作図をするため、[基本ツール] から [点] をえらび、入力してみる。



図 3

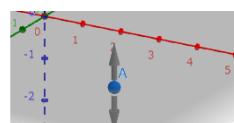


図 4

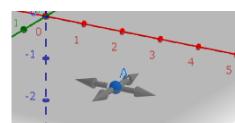


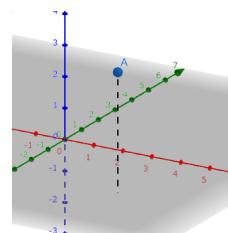
図 5

平面上に十字（図 3）が表示されるので、クリックすると点が作図される。

その後、点をクリックする毎に z 軸方向への移動（図 4）か、 $x-y$ 平面の移動（図 5）か変わる。

そして、ドラッグすることによって、点を移動させることが出来る。例えば、 z 軸方向への移動を行った場合が右図である。

このとき、入力フィールドを表示させておくと、座標をその都度表示してくれる。もちろん入力フィールドから座標を指定して入力することも出来る。



4.4.2 直線と平面

空間内に直線を描くために、少し復習をする。空間内に直線を描く場合、直線を一意的に決めなければならない。そのため、直線のベクトル方程式

$$(x, y, z) = (x_0, y_0, z_0) + t(a, b, c) \quad (t \in \mathbb{R})$$

を思い出す。GeoGebra で扱うため、 (a, b, c) はベクトル、 (x_0, y_0, z_0) は点として入力する。

例 4.4.1. 点 $(0, 2, 1)$ を通り、方向ベクトルが $(2, 1, -3)$ の直線の作図と直線のベクトル方程式を考える。

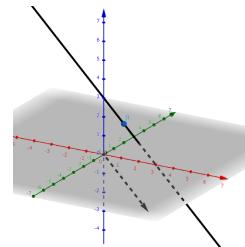
まず、入力フィールドに

$$\mathbf{a} = (2, 1, -3)$$

$$\mathbf{B} = (0, 2, 1)$$

と入力する。

次に、作画ツール群の [直線と多角形] から、[平行線] を選び、作画領域内に描かれたベクトルと点をクリックすると、直線が表示される。また、代数ウィンドウで確認すると、



●	$f : \text{Line}(\mathbf{B}, \mathbf{a})$ $\rightarrow X = (0, 2, 1) + \lambda (2, 1, -3)$	\vdots
--	---	----------

と直線のベクトル方程式が表示されている。 $(\lambda : パラメーター)$

♣ 補足 残念ながら、空間図形の場合、直線の方程式を表示する方法は不明。誰か知っていたら教えてください。

直線の方程式を入力することは可能なので、入力フィールドに $\frac{x}{2} = y - 2 = \frac{z - 1}{-3}$ と入力すると正しく直線を表示してくれる。ただし、代数ウィンドウの表示は

●	$g : (\frac{x}{2} + 0z = y - 2, \frac{x}{2} = \frac{z - 1}{-3})$ $\rightarrow X = (0.33, 2.17, 0.5) + \lambda (-0.33, -0.17, 0.5)$	\vdots
--	---	----------

と表示される。

例題. 以下の直線を作図せよ。

(2) 点 $(-1, 2, 1)$ を通り、方向ベクトルが $(0, -1, 3)$ の直線

(3) 2 点 $(-1, 0, 2), (3, 0, -2)$ を通る直線

(4) 2 点 $(-2, 3, 1), (-2, -2, 1)$ を通る直線

(5) 原点を通り平行移動すると直線 $\frac{x}{2} = y - 2 = \frac{z - 1}{-3}$ と一致する直線

次は、平面の作図を考える。平面の方程式が解っていれば、入力フィールドに入力すれば作図される。例えば、

$$3*x - 2*y + 5*z = 1$$

と入力してみる。作画領域内に平面が作図されたら、作画ツール群の [基本ツール] から [移動] を選び、作画領域内をドラッグするといろいろな角度から見ることが出来る。

例 4.4.2. 以下の平面の作図と平面の方程式を考える。

(1) 点 $(1, -2, 0)$ を通り、法線ベクトルが $(2, -1, 2)$ の平面

直線の場合と同様に、まず、入力フィールドに

$$\mathbf{a} = (2, -1, 2)$$

$$\mathbf{B} = (1, -2, 0)$$

と入力する。点とベクトルが作図されたら、作画ツール群の [平面] から、[直交平面] を選び、点とベクトルをクリックすると平面が作図される。

正しく作図されているか確かめるために、代数ウィンドウを確認すると、 $2x - y + 2z = 4$ が表示されている。

例題. 以下の直線を作図せよ。 (線型代数学と演習 I 例 3.4.1(2), (3))

(2) 3 点 $(-1, 1, 1)$, $(2, -2, 3)$, $(2, 0, -2)$ を通る平面

(3) 直線 $\frac{x}{2} = y - 2 = \frac{z - 1}{-3}$ と直交し、原点を通る平面

4.4.3 スライダーと線型変換

空間図形ではスライダーがツール群に無いが、使うことはできる。入力フィールドに

$$\mathbf{a} = \text{slider}(0, 1)$$

と入力すると、0 と 1 の間を動くスライダー \mathbf{a} が作成できる。

これを用いて、点 A, B を結ぶ線分を残像で表示してみる。上記の方法で、スライダー \mathbf{a} を作成し、

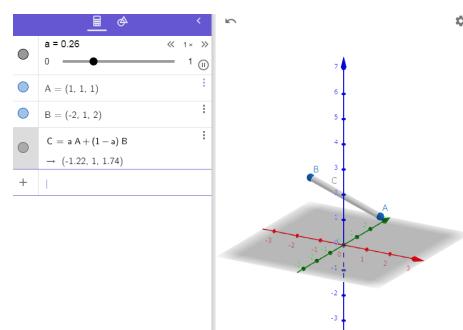
$$\mathbf{A} = (1, 1, 1)$$

$$\mathbf{B} = (-2, 1, 2)$$

$$\mathbf{C} = \mathbf{a} * \mathbf{A} + (1 - \mathbf{a}) * \mathbf{B}$$

と入力する。

点 C の残像表示を on にして、スライダー \mathbf{a} をアニメーションさせると、残像によって線分 AB を表示させることができる。



スライダーの“角度”を使う例として、次の問題を考える。

問題. 半径2で原点を中心とする球と、平面 $x + y - z = 0$ の交わりを表すベクトル方程式を答えよ。

入力フィールドに

$$x^2+y^2+z^2=4$$

$$x+y-z=0$$

`IntersectionPaths(eq2, eq1)` ← [基本ツール] の [2曲面の交線] でも可

と入力する。すると、 $X = \dots$ の形でベクトル方程式が表示される。

入力フィールドに

$$a=slider(0,2*pi,0.01*pi,1,100,true)$$

と入力し、先ほどの $X = \dots$ の部分を見ながら

$$A=(-1.41*\cos(a)-0.82*\sin(a), 1.41*\cos(a)-0.82*\sin(a), -1.63*\sin(a))$$

と入力する。スライダー a を動かすと、球面と平面の交わりを動いていることが解る。

また、行列 $M = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 2 & 0 \\ 2 & 0 & 0 \end{pmatrix}$ は、次のように入力する。

$$M = \{\{0, 0, 2\}, \{0, 2, 0\}, \{2, 0, 0\}\}$$

行列が表示されたら、

$$M*A$$

と入力することによって、 M を表現行列とする線型変換によって、点 A を移した先の点が表示される。

課題 4.4.1. 3点 $A(1, 1, 1), B(-2, 2, 1), C(1, -3, 2)$ を作図し、“残像を用いて”三角形 ABC を作図せよ。また、作図した三角形 ABC を表現行列

$$M = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

をもつ線型変換で移した先の図を“残像を用いて”作図せよ。

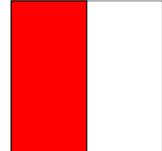
課題 4.4.2. 半径2で原点を中心とする球と、平面 $2x - y - 3z = 0$ の交わりを表す図を、“残像を用いて”作図せよ。また、作図して出来た図形を課題 4.4.1 の線型変換で移した先の図を“残像を用いて”作図せよ。

4.5 面積

4.5.1 面積と確率

2次元平面に話を戻して、面積を考える。

例えば、面積比が 1 : 1 で、赤と白に塗り分けられた紙がある。無作為にこの紙に針を落としたとき、必ず赤の部分か白の部分に刺さるとする。



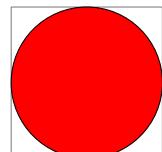
10回この試行を行ったとき、赤に刺さる回数は何回は、4,5,6回ぐらいと予想される。100回この試行を行った場合は、45回から55回の間と思われる。(正確には、境界線上を考えなければならないが、ここでは無視する。)

では、逆に面積比が $a : b$ で、赤と白に塗り分けられた紙に対して、同様の試行を行ったとき、1000回行ったら赤に 697 回刺さった。このとき、 $a : b$ はおそらく 7 : 3 ではないかと予想できるであろう。

このように面積と確率の関係を用いて円周率の近似値を計算することができる。

モンテカルロ法

右図のように半径 1 の円と、それに外接する正方形(面積は 4)を考える。この正方形の中に、ランダムに N 個の点を作図し、円の中に入った点の数を数える。



GeoGebra では、以下を入力バーに入力する。まず、解りやすくするため半径 1、原点中心の円を描くき、作画する点の個数を変えるため、スライダーを用いて N を定める。

```
x^2+y^2=1
N=slider(10,1000)
```

つぎに、乱数 (random) を用いて、点を打つ。

```
L=sequence((random(-N,N)/N,random(-N,N)/N),i,1,N)
```

このコマンドの中の `random(-N,N)/N` は、 $-N$ から N までの乱数を生成し、それを N で割っている。すなわち、-1.0 から 1.0 の数を生成している。

`sequence(F,i,1,N)` は、 F を 1 から N まで作成し、列としたものである。変数として i を使うこともできる。例えば、

```
sequence(2*i,i,1,10)
```

とすると 2 から 20 までの偶数列となる。

作成した点列のなかで、円に入っている点を数え上げる。表示コマンドとしては

```
Sum((if(length(element(L,n))<=1,1,0)),(n),(1),(N))
```

であるが、Sum(まで入力すると $\sum_{=}$ ()と表示が変わる。その後、nを入力し、右移動、1を入力し、右移動、Nを入力して右移動する。下のように入力が出来たらエンターキーを押す。

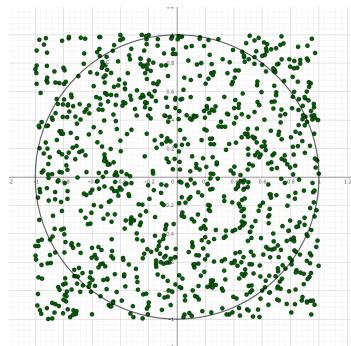
$$\sum_{n=1}^N (\text{if}(\text{length}(\text{element}(L, n)) \leq 1, 1, 0))$$

この element(L, n) は、Lの中の n 番目の成分という意味である。また、if(G<=1, 1, 0) は G が 1 以下なら、1、そうでないならば 0 と言う意味である。最後に

$$p=4*a/N$$

とすれば、円周率の近似値が表示される。分数で表示されている場合は、 を押せば小数表示してくれる。

$$\begin{aligned} p &= 4 \cdot \frac{a}{N} \\ &\approx 3.124 \end{aligned}$$



4.5.2 多角形の面積

多角形の面積を求める場合は、area を用いる。紹介のために次の例題を考える。

例題. (面積の最大値)

2次関数 $f(x) = x^2 + 3x - 3$ に対して、2次曲線 $y = f(x)$ と、曲線上の点 $A(-3, -3)$, $B(2, 7)$, $C(k, f(k))$ を考える。ただし、 $-3 \leq k \leq 2$ とする。

三角形 ABC の面積が最大となる k の値と、そのときの面積を求めよ。

まずは、入力フィールドに以下を入力する。ただし今回は、観測することにする。

```
f(x)=x^2+3*x-3
A=(-3,-3)
B=(2,7)
k=slider(-3,2)
C=(k,f(k))
p=polygon(A,B,C)
s=area(p)
```

作図が出来たらスライダー k を動かしてみる。この場合はアニメーションより手動で動かした方が解りやすい。三角形を解りやすくするために polygon を用いて三角形を作図しているが、area(p) は、area(A,B,C) としてもよい。

4.5.3 区分求積法

区分求積法については既に知っていると思うのでここでは述べない。ただ、知ってはいるが、実際に計算して求めたことは少ないと思われる所以、GeoGebra を用いて計算をしてみる。

さっそく以下を入力フィールドに入力して行く。コマンドが長く、sum もあり、入力が大変かもしれない。

```
f(x)=x^3+2*x^2-3
n=slider(2,100,1)
Sequence(Polygon((j/n,f(j/n)),(j/n,0),
                  ((j+1)/n,0),((j+1)/n,f(j/n))),j,0,n-1)
Sequence(Polygon((j/n,f((j+1)/n)),(j/n,0),
                  ((j+1)/n,0),((j+1)/n,f((j+1)/n))),j,0,n-1)
a=Sum(Area((j/n,f(j/n)),(j/n,0),
            ((j+1)/n,0),((j+1)/n,f(j/n))),j,0,n-1)
b=Sum(Area((j/n,f((j+1)/n)),(j/n,0),
            ((j+1)/n,0),((j+1)/n,f((j+1)/n))),j,0,n-1)
```

実際の値を求めるには、

```
abs(integral(f(x),0,1))
```

と入力する。

ここで関数 $f(x)$ をいろいろと変えて観察してみるのもよい。

積分が出てきたので、微分、積分のコマンドを紹介しておく。

4.5.4 微分、積分

GeoGebra は関数の微分、積分も表すことが出来る。入力フィールドに

```
derivative((x-1)^3) # 微分
```

と入力すると、 $(x - 1)^3$ の微分が求まる。 $\sin x$ や $\log x$ の微分

```
derivative(sin(x))
derivative(log(x))
```

も関数のグラフとして表示させることが出来る。積分も微分と同様に

```
integral((x-1)^3)
```

と入力すると、 $(x - 1)^3$ の積分が求まる。 $\int \sin(x)dx$, $\int \log x dx$ も同様に次のように入力する。

```
integral(sin(x))
integral(log(x))
```

積分の場合、区間を入力すると定積分を求めることができる。例えば $\int_0^{\pi} \sin(x)dx$ は

```
integral(sin(x),0,pi)
```

と入力する。通常の GeoGebra では 2.0000000000791 と表示されるが、CAS を使うと 2 と正しく表示される。

♡ 注意! $y = \frac{1}{x}$ の区間 $[0, 1]$ での定積分

```
integral(1/x,0,1)
```

は ? と表示される。

課題 4.5.1. モンテカルロ法を用いて、楕円 $\frac{x^2}{25} + \frac{y^2}{16} = 1$ の面積の近似値を求める手順を *GeoGebra* で作成せよ。

課題 4.5.2. ある関数を決めて、その微分、不定積分、定積分を求めよ。ただし、関数や積分区間などは自由に決めて良い。

4.6 数式の扱い

4.6.1 CAS

CAS とは、Computer Algebra System の頭文字を合わせたもので、数式処理を意味している。

前回少し紹介したが、[アプリの開始] から始めた直ぐの状態（以下 通常という）では、 $\int_0^\pi \sin(x)dx$ の値が正しく表示されなかった。しかし、数式処理（CAS）を使うと、正しく表示される。

♠ 捷足 $\int_0^\pi 2 \sin(x)dx$ はどちらも 4 と表示される。

数式処理（CAS）は作図より、式の計算に特化している。通常と CAS との違いを見るなどを含めて、幾つかのコマンドを紹介しておく。コマンドの一覧と、簡単な説明は、入力キーボード右上の



（図の赤丸の部分）をクリックすると表示される。数学関数は見慣れたものが並んでいる。また、コマンドは英単語を使っていることが多いので、意味合いは直感的に（もしくは、英和辞典で調べて）理解することができる。ただ、使い方は少し難しいものが多い。

(1) 平方完成

例. CompleteSquare(x^2+3*x+2)

(2) n 個の点を通る $n - 1$ 次関数

例. Polynomial((0,1),(2,-1),(3,1),(5,-2),(6,3))

(3) テーラー展開

例. TaylorPolynomial(ln(x),0,5) (注)ln(x) は、 $\log_e(x)$ のこと。

(4) 近似分数（CAS ではエラーになる）

例. FractionText(pi)

(5) 最大公約数、最小公倍数

例. GCD(12345,67890)

例. LCM(98765,43210)

例をあげるときりが無いので、後は各自で楽しんでください。

4.6.2 マクローリン展開

スライダー(パラメーター)を使うことによって、関数の幕級数の近似値を目視で求めてみる。ここでは、マクローリン展開を考えることにする。すなわち、

$$\begin{aligned} f(x) &= f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}x^2 + \frac{f^{(3)}(0)}{3!}x^3 + \dots \\ &\doteq a_0 + a_1x + a_2x^2 + a_3x^3 + \dots \end{aligned}$$

の $a_0, a_1, a_2, a_3, \dots$ を考える。

例題。関数 $f(x) = \frac{1}{1-x/2}$ のマクローリン展開の x^3 の項までを GeoGebra を用いて推測せよ。

まず、入力フィールドに

$$f(x) = 1/(1-0.5*x)$$

と入力する。 a_0 は明らかに 1 なので、

$$g(x) = 1$$

とする($g(x)$ が求めたい近似式)。次に、

$$t(x) = f(x) - g(x)$$

と入力し、プロパティからこの関数の色を

(例えば 青に) 変る。次に

$$a = \text{slider}(-3, 3, 0.01)$$

$$h(x) = a*x$$

とする。 $(h(x))$ の色を例えば黒に変えておく。)

a をアニメーションさせることにより、青色($t(x)$)のグラフと黒色($h(x)$)のグラフを観察すると a の値が 0.5 の辺りで一番近似されていることが解る(のではないか?)。

このことから、 x の係数は 0.5 と予想されるので、 $g(x)$ を

$$g(x) = 1 + 0.5*x$$

に変更する。

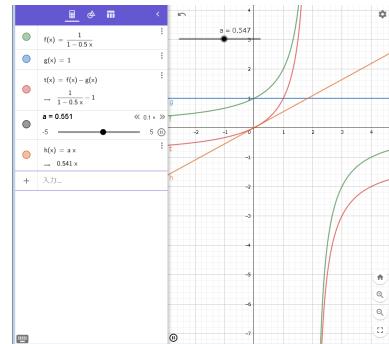
今度は 2 次の項 x^2 の係数を観察するために、 $h(x)$ を

$$h(x) = a*x^2$$

に変更する。

先ほどと同様に、青色($t(x)$)のグラフと黒色($h(x)$)のグラフを観察すると(見づらくなってくるが)、だいたい 0.25 だと解る。このことから、 $g(x)$ を $g(x) = 1 + 0.5*x + 0.25*x^2$ に変更し、 $h(x)$ を $h(x) = a*x^3$ に変更して再び観察する。

これらによって、3 次までの近似が $g(x) = 1 + 0.5x + 0.25x^2 + 0.125x^3$ だと推測できる。もちろん推測なので、別の答えが出るのも良い。



4.6.3 GeoGebraまとめ課題

課題 4.6.1. 三角形を作成し、その三角形に対して、垂心、重心、傍心を定規とコンパスを用いる方法で作図せよ。

課題 4.6.2. スライダーおよび点の残像を用いて、サイクロイド、内サイクロイド、外サイクロイドの曲線を描け。ただし、定数 $a, b > 0$ は好みで設定してかまわない。

$$\text{サイクロイド} \quad \begin{cases} x = a(\theta - \sin \theta) \\ y = a(1 - \cos \theta) \end{cases}$$

$$\text{内サイクロイド} \quad \begin{cases} x = (a - b) \cos \theta + b \cos \left(\frac{a-b}{b} \theta \right) \\ y = (a - b) \sin \theta - b \sin \left(\frac{a-b}{b} \theta \right) \end{cases}$$

$$\text{外サイクロイド} \quad \begin{cases} x = (a + b) \cos \theta - b \cos \left(\frac{a+b}{b} \theta \right) \\ y = (a + b) \sin \theta - b \sin \left(\frac{a+b}{b} \theta \right) \end{cases}$$

♠ 補足 内サイクロイド、外サイクロイドは、特定の定数のときアステロイド、カージオイドという名前がついている。

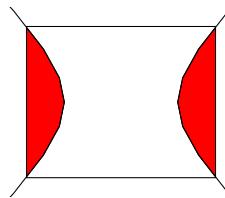
課題 4.6.3. 平面上に異なる 2 点 A, B を入力し、線分 AB を $2 : 1$ に内分する点を定規とコンパスで作図する方法で作図せよ。

課題 4.6.4. 空間図形をつかって、正二十面体を作図せよ。ただし、定規とコンパスの作図方法にこだわらなくてよい。

課題 4.6.5. 空間図形をつかって、正十二面体を作図せよ。ただし、定規とコンパスの作図方法にこだわらなくてよい。

課題 4.6.6. 右図は双曲線 $x^2 - y^2 = 9$ のグラフと、4点 $(5, 4), (-5, 4), (-5, -4), (5, -4)$ を頂点とする長方形である。

赤色で塗られた部分の面積の近似値を求めよ。



課題 4.6.7. $y = \sinh x, y = \cosh x$ のマクローリン展開の係数の近似を GeoGebra で目視し 3 次近似まで推測せよ。

課題 4.6.8. $y = \log(x + 2)$ のマクローリン展開の係数の近似を GeoGebra で目視し 3 次近似まで推測せよ。また、その近似を用いて $\log 2, \log 3$ の値を推測せよ。(注意が必要)

<おまけ>

以下の 2 次曲面を描け。ただし、定数 $a, b, c, p > 0$ は好みで設定してかまわない。

$$\text{楕円面 } \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

$$\text{一葉双曲面 } \frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$$

$$\text{二葉双曲面 } \frac{x^2}{a^2} - \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$$

$$\text{楕円錐 } \frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 0$$

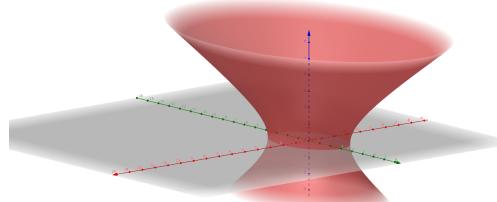
$$\text{楕円放物面 } \frac{x^2}{a^2} + \frac{y^2}{b^2} - 2z = 0$$

$$\text{双曲放物面 } \frac{x^2}{a^2} - \frac{y^2}{b^2} - 2z = 1$$

$$\text{楕円柱 } \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0$$

$$\text{双曲柱 } \frac{x^2}{a^2} - \frac{y^2}{b^2} - 1 = 0$$

$$\text{放物柱 } \frac{x^2}{a^2} - 2yp = 0$$



第5章

おまけ

5.1 L^AT_EX 環境の構築

5.1.1 MS Windows へ L^AT_EX のインストール

T_EX は沢山の種類があるが、ここでは、TeXLive のインストール方法を紹介する。

♡ 注意! この方法はインターネットに接続してインストール作業を数時間行うため、ネットの環境には注意すること。

まず、TeXWiki TeX Live/Windows サイト

<https://texwiki.texjp.org/?TeX%20Live%2FWindows>

を開くか、検索サイトで “tex wiki tex live windows” を調べて以下のサイト 1 を開く。

The screenshot shows the TexWiki TeX Live/Windows page. At the top, there's a navigation bar with links like 'トップ' (Top), '検索' (Search), '差分' (Difference), 'バックアップ' (Backup), '添付' (Attachment), 'リロード' (Reload), '新規' (New), and '一覧' (List). Below the navigation, there's a breadcrumb trail: 'TeX Live > Windows'. The main content area has a heading 'Installing TeX Live over the Internet'. Below it, there's a note: 'ここでは全てのファイルが入っている ISO イメージをダウンロードする方法と、ネットワークインストーラーによる install-tl-windows.exe / install-tl.zip を用いてインストールする方法を説明します' (This page explains how to download the entire ISO image and how to install using a network installer via install-tl-windows.exe or install-tl.zip). A red oval highlights the 'install-tl-windows.exe' link. At the bottom of the page, there's another section titled 'ネットワークインストーラーの場合' (When using a network installer) with a note about right-clicking and running as administrator.

サイト 1

Installing TeX Live over the Internet

TeX Live 2022 was released on April 3.

For typical needs, we recommend starting the TeX Live installation by downloading (these links go to mirrors): [install-tl-windows.exe](#) for Windows (~20mb), or [install-tl-.tar.gz](#) (~5mb) for everything else. There is also a zip archive [install-tl.zip](#) (~25mb) which is the same as the .exe. Although the .zip archive works fine on all platforms, the .tar.gz is much smaller, since it omits installation support programs needed only on Windows. The archives are otherwise identical.

The above links use the [generic.mirror.ctan.org.uk](#) which autoredirects to a CTAN mirror that should be reasonably nearby and reasonably up to date. However, perfect synchronization is not possible; if you have troubles following the links, your best bet is to replace the [mirror.ctan.org](#) in the above urls with a specific host from the [list of CTAN mirrors](#).

サイト 2

このサイト 1 の [Installing TeX Live over the Internet](#) (上図 丸囲み) をクリックすると、サイト 2 に移る。次に、このサイト 2 の [install-tl-windows.exe](#) (上図 丸囲み) をダウロードする。

セキュリティソフトが入っている場合は、警告等が表示されるかもしれないで、適時対応する。ダウンロードが終了したら、install-tl-windows を実行する。

ブラウザから直接実行出来ない場合は、ダウンロードフォルダを探してみる。

エクスプローラーを開き、

デスクトップ → ユーザー名 → ダウンロード

の順に辿る。

install-tl-windows を実行すると、右図の警告が表示される（場合がある）。この場合、詳細情報 をクリックすると [実行] が現れるので、実行をクリックする。

インストーラーが起動すると、まずシングルユーザーでの使用であることの注意が表示され、インストールするのか、解凍のみを行うのか聞かれる（図 1）。

もし、全てのユーザーで使用したい場合は、[Cancel] を押して管理者権限で実行する。

[Next >] を押すと、確認画面（図 2）が表示されるので、問題無ければ [Install] を押す。

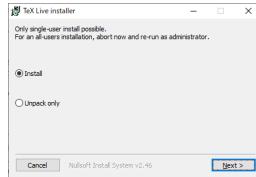


図 1

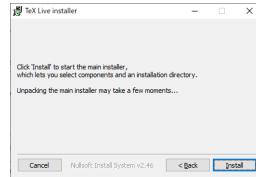


図 2



実行しない

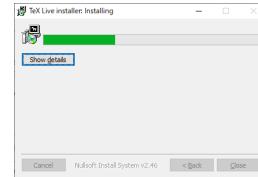


図 3

圧縮ファイルの展開が始まり（図 3）、展開が終わると図 4 の画面が表示される。そのまま少し待つと、図 5 が表示される。



図 4

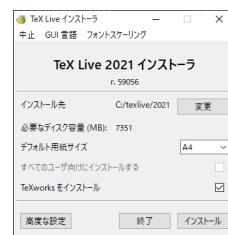


図 5

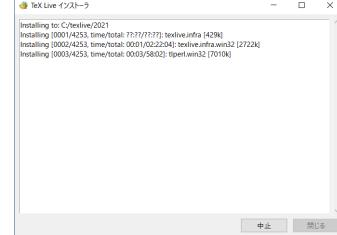


図 6

♡ 注意! ユーザ名が日本語の場合、図 6 の途中で止まることがある。その場合、図 5 にある [高度な設定] を選び、右図の画面を表示させる。

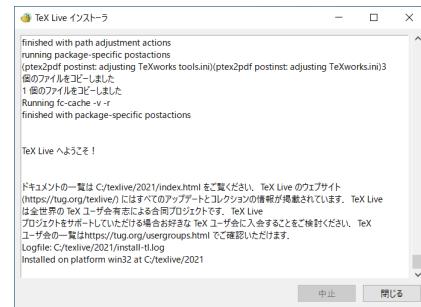
この左の欄の赤で囲まれた部分 “ポータブル設定: いいえ” の右の [切替] を押して、“ポータブル設定: い”にして [インストール] を押す。



図 5において、変更がある場合は変更を行い、[インストール] を押すと図 6 の画面が表示され、インストールが始まる。

(環境によって異なるが) 2~3 時間程度して、右図の画面 (“TeX Live へようこそ！”) が表示されると無事終了となる。

デスクトップにショートカット  を作りたい場合は、エクスプローラで



```
C:\texlive\2023\bin\win32
```

にある Texworks を右クリックし、ショートカットの作成を選ぶと
texworks.exe - ショートカット

が出来る。

これをデスクトップに移動し、名前を “TeX Works” に変える。

5.1.2 タイプセットに pLaTeX(ptex2pdf)が無い場合

タイプセット (図 7) に pLaTeX(ptex2pdf) が無い場合は、以下の方法で追加する。

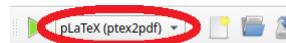


図 7

[編集] の [設定] から [タイプセット] を選び、タイプセットの方法の右下の  を押す。

開いたウインドウ内 (図 8) に、

名前 : pLaTeX (ptex2pdf)
 プログラム : ptex2pdf
 引数 : (上から順に)
 -1 (← 小文字のエル)
 -ot
 -kanji=utf8 \$synctexoption
 \$fullname

と入力して  を押す。

また、[デフォルト:] を今作成した
pLaTeX (ptex2pdf) に変えておく。

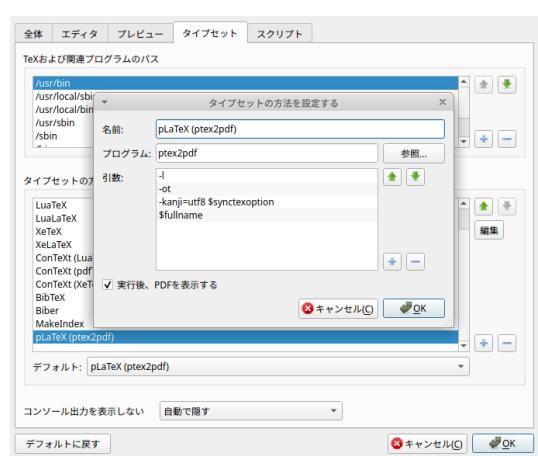


図 8

5.1.3 MacOSへのTeX Liveのインストール

Googleで“tex live インストール mac”と検索し、“TeX Live/Mac - TeX Wiki”的ページを開く。



このページ中ほどの **TeX Live** 公式のインストーラの使用に従って進めていく。

TeX Live 公式のインストーラの使用

TeX Live のインストールガイド

- <http://www.tug.org/texlive/quickinstall.html>
- <http://www.tug.org/texlive/doc/texlive-en/texlive-en.html#installation>
- <http://www.tug.org/texlive/doc/texlive-ja/texlive-ja.pdf#>

を見ると **MacTeX** が推奨されていますが、MacTeX に付属する TeX Live には universal-darwin バイナリのみが収録されています。一方、TeX Live 公式のインストーラを使えば、universal-darwin / x86_64-darwinlegacy いずれの環境でも TeX Live をインストールすることができます。

まず、[ミラーサイト](#)から `install-tl-unx.tar.gz` をダウンロードします。

```
$ curl -OL http://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz
```

`install-tl-unx.tar.gz` を展開します。

```
$ tar xvf install-tl-unx.tar.gz
```

図 1

まず、ミラーサイトから `install-tl-unx.tar.gz` をダウンロードするために、ターミナルを開く。開いたターミナルから

```
curl -OL http://
mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz
```

と入力する（改行は印刷の都合上なので、続けて入力する。図 1 のサイトを開いて、コピペするのが良い）。

```
Last login: Wed Oct 27 16:36:11 on console
[1-Last@LastloginnoMacBook-Air ~ % curl -OL http://mirror.ctan.org/systems/texlive/tlnet/install-tl-unx.tar.gz
% Total    % Received % Xferd  Average Speed   Time   Time     Current
          Dload  Upload Total   Spent    Left Speed
100  341  100  341    0      0  404      0 --:--:--:--:--:-- 404
100 6651k  100 6651k    0      0 2963k      0:00:02 0:00:02 --:--:-- 732/k
1-Last@LastloginnoMacBook-Air ~ %
```

ダウンロードが正しく終われば上図のような表示となる。

続いて、install-tl-unx.tar.gz を展開するために、ターミナルから

```
tar xvf install-tl-unx.tar.gz
```

と入力する。

展開が開始されると右図のように表示される。

展開後、インストーラのあるディレクトリに移動する。

```
cd install-tl-2*
```

移動後、root 権限で TeXlive のインストールを開始する。(詳しくは図 1 のサイトを参照。)

```
sudo ./install-tl -no-gui -repository
http://mirror.ctan.org/systems/texlive/tlnet/
```

(改行は印刷の都合上なので、続けて入力する。なお、改行されている部分は半角の空白を入れる。)

インストールを開始すると以下の画面となる。

問題が無ければ、I(アイ) を入力してハードディスクへのインストールを続ける。

インストールが終了したら、/usr/local/bin ディレクトリ配下にシンボリックリンクを追加する。

```
sudo /usr/local/texlive/????/bin/*/tlmgr path add
```

上記の ???? は 2023, * は universal-darwin にマッチするはずであるが、うまく動作しない場合は以下のようにして具体的なディレクトリ名を指定する。

```
sudo /usr/local/texlive/2023/bin/universal-darwin/tlmgr path add
```

それぞれの場合、上手く行かない場合は図 1 のサイトを参照し、エラーの対処を行う。

5.1.4 MacOSへのTeXWorksのインストール

Googleで“Releases texworks github”と検索し“TeXworks”的ページを開く(図2)。

図2

【2023年9月現在TeXworks 0.6.8 [Lastest]である。】

このサイトの中にコンテンツ(Assets)

File	Size	Published
TeXworks-macos10.12-0.6.7-202202261113-git_23c4c74.dmg	25.3 MB	26 Feb 2022
TeXworks-macos10.15-0.6.7-202202261118-git_23c4c74.dmg	26.5 MB	26 Feb 2022
TeXworks-win-0.6.7-202202261139-git_23c4c74.zip	25.1 MB	26 Feb 2022
TeXworks-win-setup-0.6.7-202202261139-git_23c4c74.exe	25.2 MB	26 Feb 2022
Source code (zip)		26 Feb 2022
Source code (tar.gz)		26 Feb 2022

があるので、macOS 10.15以降の場合は、

TeXworks-macos10.12-0.6.7-202*****-git_23*****.dmg

を、macOS 10.15以前の場合は、

TeXworks-macos10.15-0.6.7-202*****-git_23*****.dmg

をダウンロードし、インストールする。

場合によっては、Rosettaのインストールを促されるので、インストールを行う。

5.1.5 Linux の L^AT_EX 環境

Linux の場合、既にインストールされている場合がある。そこで、ターミナルから

```
$ latex
```

と入力してみる。インストールされている場合

```
Terminal - user@user:-
ファイル(F) 儿童(E) 表示(V) ターミナル(T) タブ(A) ヘルプ(H)
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

user@user:~$ latex
コマンド 'latex' が見つかりません。次の方法でインストールできます:
sudo apt install texlive-latex-base
user@user:~$
```

This is pdfTeX, Version 3.14159265-2.6-1.40.18 (TeX Live 2017/W32TeX)
(preloaded format=latex) restricted \write18 enabled.
**

のような表示がある。また、インストールされていない場合、

bash: latex: コマンドが見つかりませんでした...
コマンド'latex'を提供するためにパッケージ'texlive-latex-bin-bin'をインストールしますか? [N/y]

または、

Xubunt 20.10 の場合 —————

コマンド 'latex' が見つかりません。次の方法でインストールができます:
sudo apt install texlive-latex-base

のようにインストールの問合せやインストール方法の紹介が表示される場合がある。

ここでは、

```
$ sudo apt install texlive-latex-recommended texlive-latex-extra
texlive-fonts-recommended texlive-fonts-extra texlive-lang-japanese
texlive-lang-cjk
```

を実行する(3行に分かれているが、続けて入力)。パスワード入力後、依存関係のパッケージなども含めてインストールが行われる。

(補足) T_EX のパッケージ関連で「oooo not found.」というエラーが出たらその都度パッケージのインストールを行う必要がある。

それが面倒で、ハードディスクの空き容量があれば、

```
$ sudo apt install texlive-full
```

としてフルパッケージをインストールする。

5.1.6 LinuxへのTeXWorksのインストール

[以下は X Window System が起動していることを前提としている]

TeXworks をインストールするために、先ほどと同様にターミナルから

```
$ sudo apt install texworks
```

を実行する。インストールが終われば

```
$ texworks &
```

と入力して何か TeX ソースファイルを作りコンパイルしてみる。

5.2 竈門禰豆子

L^AT_EXにおいて TeXWorks では表示されていても、変換すると表示されない文字がある。また、(OS やバージョン、TeX の種類も含めて) 環境によって表示される文字が異なることがある。

まず、環境(機種)依存文字(例えば、I, ①, リツ, (株), (上), 鈴, ❶など)はそのままでは使用できない。

これらは、プリアンブルに

```
\usepackage[bxutf8]{inputenc}
\usepackage{bxbase}
\usepackage{otf}
```

を追加することによって表示させることができる(ことがある)。

表示されない、エラーが出る場合は \ajRoman{1}, \ajMaru{1}, \ajLig{リットル}, \ajLig{(株)}, \ajLig{○上}, \ajLig{令和}, \ajKuroMaru{1} を試してみる。

次に、高, 崎, 濱, 德 なども上記の 3 行をプリアンブルに書くことによって表示させることができるものがあるが、これら文字を TeXWorks で表示させることが困難である。

そこで、これらは“UTF”または“CID”と似た字(高、崎、濱、徳など)で検索すると UTF または CID の文字コードがみつかると思うので、\UTF{9AD9} または \CID{8705} とすることで高が表示される。

この方法でも表示させることができない文字があり、竈門禰豆子は 環境によって pL^AT_EX で表示させることができない。その場合、竈門禰豆子となってしまう。

(禰 (\CID{3295}) も禰 (\CID{7769}) も同じ禰が表示される。)

そのような環境で禰を表示させたい場合、今のところは lualatex を使う方法しかない。

♣ 補足 この pdf を作成する環境は Windows 10, TeXLive 2022 pL^AT_EX(ptex2pdf) である。