

2012 年度 前期 電子計算機 1

更新日時 2012-06-11 19:16:11 担当 和地 輝仁

目次

1	シラバス抜粋	1
2	授業のノート (Small Basic 編)	2
§1	表計算ソフトウェアによる統計処理	2
§2	最初の Small Basic プログラム	3
§3	変数	5
§4	条件分岐	7
§5	For 文による繰り返し	10
§6	While 文による繰り返し	12
§7	サブルーチン	15
§8	いろいろな関数	17
§9	配列	19
§10	グラフィックス	19
§11	練習問題	19
3	授業のノート (TeX 編)	21
§1	文書組版システム TeX の導入	21
§2	TeX の使用方法	21
§3	文書の構造	24
§4	箇条書き	26
§5	表組み	28
§6	数式	29
§7	相互参照・目次 (・索引)	35
§8	画像	37
§9	マクロ	38

1 シラバス抜粋

授業の目標 コンピュータを用いて文書を作成したり、資料の整理を行うために、いくつかのアプリケーションソフトを利用できるようになる。また、効率を上げるためにプログラムを利用したり、論理構造を重視した文書の作成ができるようになる。

到達目標

1. 表計算ソフトウェアで統計処理ができる。
2. 基本的なプログラムの作成方法を学び、作業の効率化を図れる。
3. 文書組版システムを用いて、論理構造をもつ文書の作成ができる。
4. コンピュータを用いた学習教材の作成ができる。

授業計画 順序を交換する場合もあるので注意すること。

- | | |
|---------------------|---------------------|
| 1. 表計算ソフトウェアによる統計処理 | 8. 文書組版システム TeX の導入 |
| 2. プログラム | と使用方法 |
| 3. 変数・演算 | 9. 文書の構造 |
| 4. 関数とプロシージャ | 10. 表・箇条書き |
| 5. 制御構造 1 | 11. 数式 |
| 6. 制御構造 2 | 12. 相互参照・目次・索引 |
| 7. 制御構造 3 | 13. 画像 |
| | 14. マクロ |
| | 15. 期末試験 |

成績評価 期末試験では、プログラムの作成と、文書組版システムを用いた文書の作成ができるかどうかを評価する (50%)。各回の授業では演習問題を出題し、その提出状況と内容も評価する (50%)。原則として全ての時間の出席を求めるが、やむを得ない理由で欠席をする (した) 場合はできるだけ速やかに申し出て、指示を受けること。

2 授業のノート (Small Basic 編)

§1 表計算ソフトウェアによる統計処理

(1.1) 課題 01 – 課題提出の練習 以下のことを実行して下さい。

- (1) 小手調べに「メモ帳」を起動して、適当な俳句を詠んで入力する。
- (2) 「0000 和地-01」のような「学生番号名前-課題番号」というファイル名¹で保存する。数字は半角。
- (3) 「提出用ページ」² をウェブブラウザで開き、「課題提出」の所に配布したパスワードを入力し、提出するファイルを選択する。そして「提出」ボタンを押すと提出が完了する。
- (4) 「提出用ページ」の「提出状況閲覧」の所に、パスワードと学生番号を入力し、「閲覧」ボタンを押すとこれまでのすべての提出状況が見られる。

また、提出した課題に修正を求められたときなどは、修正後のファイルを 1 回目と同様に提出して下さい。同一ファイル名でも区別して管理されるため、「提出状況閲覧」から提出状況を見ると、2 回提出したこともわかります。

(1.2) 課題の可否と再提出 「提出状況閲覧」では、提出したファイルが課題番号ごとに一覧できます。複数回提出した課題では、新しいファイルほど下に表示されます。ファイル名の右に「合格」とあれば、その提出課題は合格です。「再提出」とあれば、修正して再提出して下さい。

(1.3) Excel の式 Excel のセルには、= で始まる式を入力することができます。

Excel の式の文法

= 式

次の課題では「IF」という関数を用います。

¹テキストファイルの拡張子「.txt」も含めれば、「0000 和地-01.txt」となる

²URL は板書します

IF の文法

IF (論理式, 真の場合, 偽の場合)

使用方法は、Excel のヘルプでも見られます。

式にはセルの値も利用できるので、例えば、「= IF(A1 > 3, "3 より大きい", 小さい)」のような式が書けます。

(1.4) 課題 02 – Excel でプログラムの初歩 次の内容を記載した Excel のファイルを提出して下さい。ファイル名は、「0000 和地-02」のような「学生番号名前-課題番号」というファイル名にします。

- (1) 「提出用ページ」から、テキストファイル「grades.txt」をダウンロードし、その内容を、エクセルのシートの A 列の A1 から A100 までに埋める (コピーした後のコピー領域右下のアイコンを使うのが楽)。
- (2) A 列が 60 点以上ならば B 列に「合格」、そうでなければ「不合格」と表示する IF を用いた式を B 列に入力する。例えば、A1 のセルが 50 ならば B1 のセルに「合格」、A2 のセルが 70 ならば B2 のセルに「不合格」と表示されるような IF を用いた式を B1 から B100 までに入力する。
- (3) A 列の数値が、90 を越えていたら「!」(半角感嘆符) と表示されるような IF を用いた式を C1 から C100 までのセルに入力する。90 以下のときは何も表示しなくてよい。
- (4) A 列の数値が、10 以下ならば「SS10」(すべて半角) と表示されるような IF を用いた式を D1 から D100 までのセルに入力する。
- (5) A 列の数値が、7 の倍数ならば「なな」と表示されるような IF を用いた式を E1 から E100 までのセルに入力する。

完成したシートは以下のようなになる。ただし、B 列から E 列までは、このように手で入力するのではなく、このように表示される IF を用いた式を入力すること。

.	A	B	C	D	E
1	3	不合格		SS10	
2	49	不合格			なな
3	63	合格			なな
4	44	不合格			
5	84	合格			なな
6	65	合格			
7	94	合格	!		
8	22	不合格			
9	21	不合格			なな
:	:	:	:	:	:

§2 最初の Small Basic プログラム

(2.1) インストールと起動 Microsoft のサイト

[http://msdn.microsoft.com/ja-jp/beginner/ff384126\(en-us\).aspx](http://msdn.microsoft.com/ja-jp/beginner/ff384126(en-us).aspx) に、Small Basic のインストーラ SmallBasic.msi があります。これをダウンロードして実行すると Small Basic をインストールできます³。

Small Basic を起動するには、「スタートメニュー - すべてのプログラム - Small Basic - Microsoft Small Basic」と辿ります。

(2.2) 最初のプログラム 起動すると図のようなウィンドウが現れます。ウィンドウ上部にはボタンが配置された「ツールバー」があり、その下にはプログラムを入力する「エディタ」があります。

実際に次の 1 行からなるプログラムをエディタに入力してみます。これは、テキストウィンドウに文字列を表示して改行する命令です。

最初のプログラム

```
1 TextWindow.WriteLine("スイトピー赤と答えて年がばれる")
```

³Windows のみで実行可能です。また、Windows XP など古い OS だと、「.NET Framework 3.5 SP1」もインストールする必要があります。

すると入力中に、候補を表示したメニューが現れます。インテリセンスと呼ばれる入力補助で、クリックで候補を選択し、ダブルクリックで確定できます。また、TAB キーで候補を補完することもできます。

プログラムを実行するには、ツールバーの実行ボタンを押します。実行すると、テキストウィンドウを呼ばれるウィンドウが開いて⁴そこに俳句が表示されます⁵。

出力

```
スイトピー赤と答えて年がばれる
Press any key to continue...
```

プログラムを保存するには、ツールバーの保存ボタンを押します。保存すると、4 つのファイルが作成されます。

- .sb ファイル: プログラム本体です。メモ帳でも開けます。
- .exe ファイル: 実行ファイルです。ダブルクリックで起動します。
- .pdb ファイル: 各種データを保存しています。
- .dll ファイル: .exe ファイルの実行に必要なものです。

このうち、.dll ファイルは、複数のプログラムで共用できますので、同じフォルダに複数のプログラムを保存しても、.dll ファイルは 1 つだけ作成されます。

(2.3) 保存したプログラムの開き方 プログラムの書かれた .sb ファイルは、特定のアプリケーションに関連づけられていなければ、ダブルクリックしても開くことができません。Small Basic から開きたければ、Small Basic のメニューの「ファイル - 開く」から開けます。また、メモ帳から開きたければ、メモ帳のメニューの「ファイル - 開く」のダイアログで、ファイルの種類を「すべてのファイル」に変更すれば開けます。

.sb ファイルをダブルクリックして Small Basic で開くようにしたければ、.sb ファイルを Small Basic に関連づけるとよいです⁶。

⁴Small Basic のウィンドウの背面に隠れているかも知れません。

⁵終了するには、何かのキーを押します。

⁶方法は授業で言うかも知れません

(2.4) 課題 03 – Small Basic で俳句 何か俳句を考え、次の出力例のように俳句を表示するプログラムを書き、.sb ファイルを提出して下さい。つまり、「0000 和地-03.sb」のようなファイルを提出して下さい。

出力例

```
スイトピー
赤と答えて
年がばれる
```

(2.5) 構造「順次」 上の課題のように、複数の文を連続して実行するプログラムの構造を「順次」と呼びます⁷。

順次

```
文1
文2
:
```

(2.6) コメント プログラムの途中に 1 重引用符「'」を置くと、そこから行末までが無視されます。これを利用して、プログラムの中にコメントを書くことができます。例えば、次の項では、コメントを利用して説明を記しています。

(2.7) 文、オブジェクト、メソッド、プロパティ 下のプログラムの各部分は、オブジェクト、メソッド、入力 (引数とも言う) と呼ばれます。

用語の説明

```
1 TextWindow.WriteLine("スイトピー赤と答えて年がばれる")
2 '
3 'オブジェクト メソッド 入力 (引数とも言う)
```

⁷ これだけでは何が大事かわからないですが、順次はプログラムにおいて大事な 3 つの構造のうちの (最も簡単な) 1 つです。残り 2 つもそのうち紹介します。

また、ドットの後には、プロパティと呼ばれるものも来ます。例えば、TextWindow オブジェクトは、文字の色を表す ForegroundColor プロパティを持ち、下のようにすると文字の色⁸ を変更してから文字列を表示します。

プロパティ

```
1 'オブジェクト プロパティ
2 '
3 TextWindow.ForegroundColor = "Green"
4 TextWindow.WriteLine("スイトピー赤と答えて年がばれる")
```

まとめると次のようになります。

- 文: 命令の最小単位。上の例では 1 行が 1 文だが、複数行に渡ることもある。
- オブジェクト: 操作の対象となるもの。ドットの前にある。
- メソッド: 操作の名前。ドットの後にある。
- 入力 (引数): メソッドに与えるデータ。丸かっこでくるむ。
- プロパティ: オブジェクトの持つ状態の名前。ドットの後にある。

(2.8) 演算子 Small Basic では、次の演算子が使えます。商は、9 / 4 の結果が 2.25 になるなど、小数点以下も割り算を続ける商です。また、整数の商の演算子を用意されていませんが、Math.Remainder(割られる数, 割る数) というメソッドで計算できます⁹。

⁸Green の他に使えるものは、Black, Blue, Cyan, Gray, Magenta, Red, White, Yellow, DarkBlue, DarkCyan, DarkGray, DarkGreen, DarkMagenta, DarkRed, DarkYellow です。

⁹これらの表以外でよく使われる整数の商を求める演算子はなく、メソッドも用意されていません。また、論理否定 (not) もありません。

数値演算子	説明	比較演算子	説明
+	和	=	等しい
-	差	<>	等しくない
*	積	>	(左辺が) 大きい
/	商	>=	大きいか等しい
		<	小さい
		<=	小さいか等しい

論理演算子	説明	文字列演算子	説明
And	かつ	+	連結
Or	または		

(2.9) 演算の例 以下に演算の例を示します。TextWindow.Write メソッドは、TextWindow.WriteLine メソッドと同じくテキストウィンドウに文字列を表示しますが、表示した後に改行しない点が異なります。また、今までも既に用いてきましたが、文字列は 2 重引用符 (") で囲む必要があります。

演算の例

```

1 TextWindow.Write("1 + 2 = ")
2 TextWindow.WriteLine(1 + 2)
3 TextWindow.Write("文字列のa+文字列のb = ")
4 TextWindow.WriteLine("a" + "b")
5 TextWindow.Write("文字列の1+文字列の2 = ")
6 TextWindow.WriteLine("1" + "2")

```

出力

```

1 + 2 = 3
文字列のa+文字列のb = ab
文字列の1+文字列の2 = 3

```

3 行目の演算結果が、「12」にならない点は注意が必要です¹⁰。

¹⁰Perl などこのような結果になりますが、利点より害が大きい気がします。

(2.10) 課題 04 – 演算 上の例にならって、下の出力例のような出力を行うプログラムを作成し、.sb ファイルを提出して下さい。つまり、「0000 和地-04.sb」のようなファイルを提出して下さい。ただし、左辺は単に文字列を出力し、右辺は Small Basic の演算子を用いて計算した結果を出力すること。また、文字の色は、1 行目は White、2 行目は Green、3 行目は Blue、4 行目は Cyan で表示すること。

出力例

```

電子 + 計算機 = 電子計算機
37 * 3 = 111
100 / 8 = 12.5
30を7で割った余り = 2

```

§3 変数

(3.1) 変数 変数とは、数値や文字列の値を保存しておく入れ物です。変数名は、If などの Small Basic のキーワードでない限り、先頭が英字、その後は英数字であれば自由に付けられます。

変数

```

1 a = 3
2 b = "c"
3 TextWindow.WriteLine(a)
4 TextWindow.WriteLine("a")
5 TextWindow.WriteLine(b)

```

出力

```

3
a
c

```

このように、2 重引用符で囲むと文字列そのものを意味し、囲まなければ、その名前の変数を意味します。

- (4) 「Bさんが出発してから 分後に、x m 先で A さんに追い付きます。」と表示する (と x は計算してその値を表示させる)。

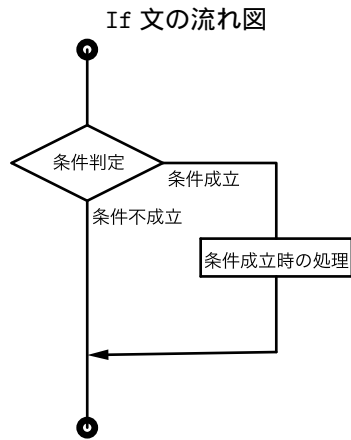
出力例

Aさんは分速72m、それより遅れて、Bさんは分速80m
で歩き始めました。
何分遅れたか入力
10
Bさんが出発してから90分後に、7200m先でAさんに追
い付きます。

ただし、上の出力例では、3行目は人間がタイプしたものです。また、紙面の都合で文の途中で折り返してありますが、プログラムではそうする必要はありません。

§4 条件分岐

- (4.1) If 文 ある条件が成立するときのみ、何かの処理をしたい場合 If 文 を用います。



分岐 (If..Then..EndIf)

```
If 条件 Then
    条件成立時の処理 (複数行でもよい)
EndIf
```

次の例では、Math.GetRandomNumber(100) というメソッドで、1 から 100 までの乱数を取得して、変数 point に代入して表示し、その値が 60 以上ならば「合格」などと表示します。

If 文の例

```
1 point = Math.GetRandomNumber(100)
2 TextWindow.WriteLine(a)
3 If point >= 60 Then
4     TextWindow.WriteLine("合格")
5     TextWindow.WriteLine("おめでとう")
6 EndIf
```

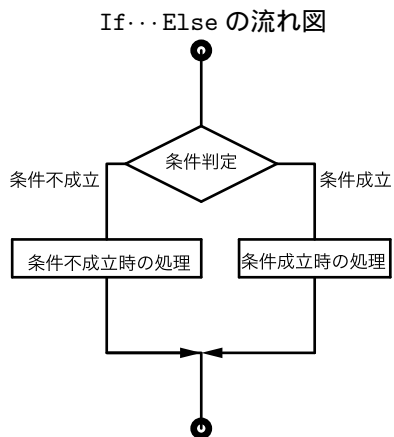
出力例その 1

95
合格
おめでとう

出力例その 2

59

(4.2) Else 節 If 文では、条件が成立したときだけでなく、Else を用いて不成立の場合も処理をさせることができます。



分岐 (If..Then..Else..EndIf)

```

If 条件 Then
    条件成立時の処理
Else
    条件不成立の時の処理
EndIf
    
```

構造「順次」は既に見ましたが、この If 文の構造は「分岐」と呼ばれます。

Else 節の例

```

1 t = TextWindow.ReadNumber()
2 If t >= 25 Then
3     TextWindow.WriteLine("夏")
4 Else
5     TextWindow.WriteLine("冬")
6 EndIf
    
```

出力例

```

24
冬
    
```

ただし、上の出力例では、1 行目は人間がタイプしたものです。

(4.3) プログラムのフォーマット If t >= 25 Then まで入力して、Enter を入力すると、カーソル位置は行頭に返るのではなく、スペース 2 つ分インデントした位置に来ます。

Then の後に Enter を入力したときのカーソル位置

```

If t >= 25 Then
    -   ここ
    
```

「条件成立時の処理」がひと固まりのブロックとして判別し易いように、ブロックごとインデントするのが作法とされているからです。編集作業の結果、インデントが狂ってしまった場合、エディタのウィンドウを右クリックして「プログラムのフォーマット」を選択すると、プログラム全体のインデントを適切に修正してくれます。

(4.4) 課題 07 – 条件分岐その 1 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「馬の年齢を入力」と表示する。
- (2) キーボードから数値の入力をする
- (3) 人間の年齢に換算すると何歳かを「人間だと 歳」と表示する。

ただし、馬の年齢 x を人間の年齢に換算するには次の式を用います。

$$(\text{人間に換算した年齢}) = \begin{cases} 6x & (x < 3) \\ 3x + 8 & (x \geq 3) \end{cases}$$

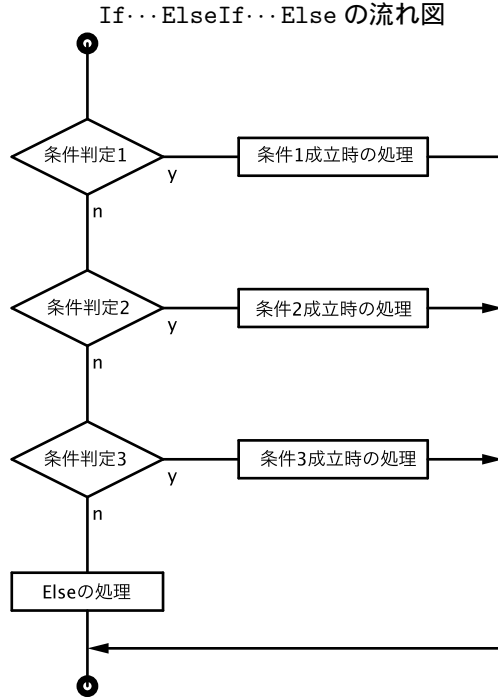
出力例

```

馬の年齢を入力
5
人間だと23歳
    
```

ただし、上の出力例では、2 行目は人間がタイプしたものです。

(4.5) ElseIf 節 If 文で ElseIf を用いると、いくつかの場合分けをして処理をさせることができます¹²。



分岐 (If..Then..ElseIf..Else..EndIf)

```

If 条件A Then
    条件A成立時の処理
ElseIf 条件B Then
    条件B成立時の処理
:
Else
    どの条件も不成立の時の処理
EndIf
    
```

¹²ここに示したものが、If 文の完全な文法です。ElseIf は何回出現しても (しなくても) 構いませんし、Else は最後に 1 回あるか、ないかのいずれかです。

複数の条件が成立するような場合でも、最初に成立した条件の処理だけを実行します。

下の例で、Clock.Hour メソッドは、現在時刻の時間の数値を 24 時制で返します。その値に応じて表示する文字列を変えています。

ElseIf 節の例

```

1 h = Clock.Hour
2 If h <= 7 Then
3     TextWindow.WriteLine("睡眠")
4 ElseIf h >= 9 And h < 18 Then
5     TextWindow.WriteLine("大学")
6 Else
7     TextWindow.WriteLine("暇")
8 EndIf
    
```

16:30 に実行した場合の出力例

大学

(4.6) 課題 08 – 条件分岐その 2 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。ElseIf を最低 1 回は用いること。

- (1) 「点数を入力」と表示する。
- (2) キーボードから数値の入力をする
- (3) 点数が 60 以上ならば「合格」と表示する。
- (4) 点数が 60 未満ならば「不合格」と表示する。
- (5) 点数が 90 以上ならば「A」と表示する。
- (6) 点数が 50 以上 60 未満ならば「F*」と表示する。

出力例その 1

点数を入力
70
合格

出力例その 2

```

点数を入力
95
合格
A

```

出力例その 3

```

点数を入力
45
不合格

```

ただし、上の各出力例では、2 行目は人間がタイプしたものです。

(4.7) 課題 09 – 最大値 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。ただし、Math.Max メソッドは用いないこと¹³。

- (1) 「数値を 3 つ入力」と表示する。
- (2) キーボードから数値の入力を 3 回行う。
- (3) 3 つの数値の最大値を求め、「最大値 = 」と表示する。

出力例

```

数値を 3 つ入力
1
6
-3
最大値 = 6

```

ただし、上の出力例では、2, 3, 4 行目は人間がタイプしたものです。

(4.8) 課題 10[#] – 最大値 (義務ではない) 課題 09 と同じ動作をするプログラムを作成し、.sb ファイルを提出して下さい。今度は、Math.Max メソッドを用いて作成すること。

¹³Math.Min メソッドも用いないこと。

(4.9) 課題 11[#] – 最大値 (義務ではない) 課題 09 と同じ動作をするプログラムを作成し、.sb ファイルを提出して下さい。ただし、Math.Max メソッドは用いてはならず、If 文は 2 つしか用いてはいけない。Else や ElseIf も用いないこと¹⁴。

§5 For 文による繰り返し

(5.1) For 文 同じような処理を繰り返す場合は For 文を用います。同じ俳句を 18 回表示したいときは、同じ命令を 18 回続けて書いてもよいですが、For 文を用いると次のように書けます。

For の例

```

1 For i = 1 To 18
2   TextWindow.WriteLine("莢豌豆むく背を見れば思い出す")
3 EndFor

```

i をカウンタ変数と呼び、この変数が開始値 1 から終了値 18 まで変わりながら、For と EndFor の間が繰り返されます。

繰り返し (For)

```

For 変数 = 開始値 To 終了値
  繰り返す処理
EndFor

```

次のように、繰り返す処理中でカウンタ変数を使用できます。

For 文でのカウンタ変数の利用

```

1 For j = 3 To 5
2   TextWindow.WriteLine(j + "番射出")
3 EndFor

```

¹⁴実際には、If 文、Math.Max メソッド、Math.Min メソッドのいずれも使わずに最大値を求められます。

出力

```
3 番 射 出
4 番 射 出
5 番 射 出
```

(5.2) 課題 12 – For 文 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「1 の 3 乗は 1 です」と表示する。
- (2) 「2 の 3 乗は 8 です」と表示する。
- (3) 「3 の 3 乗は 27 です」と表示する。
- (4) 以下順に表示し続け、最後に「55 の 3 乗は 166375 です」と表示する。

出力例

```
1 の 3 乗 は 1 です
2 の 3 乗 は 8 です
3 の 3 乗 は 27 です
( 中 略 )
54 の 3 乗 は 157464 です
55 の 3 乗 は 166375 です
```

(5.3) Step# For 文で Step を用いると、カウンタ変数の増分を指定できます。

Step

```
For 変数 = 開始値 To 終了値 Step 増分
  繰り返す処理
EndFor
```

Step の例

```
1 For j = 3 To 8 Step 2
2   TextWindow.WriteLine(j + "番射出")
3 EndFor
```

出力

```
3 番 射 出
5 番 射 出
7 番 射 出
```

(5.4) 課題 13# – Step (義務ではない) 次のような出力をするプログラムを作成し、.sb ファイルを提出して下さい。

出力例

```
1 の 平方根 は 1 です
4 の 平方根 は 2 です
7 の 平方根 は 2.64575131106459 です
10 の 平方根 は 3.16227766016838 です
( 中 略 )
55 の 平方根 は 7.41619848709566 です
```

平方根の計算には、Math.Sqrt メソッドを用います。

(5.5) For 文の利用 繰り返しの処理で変数の状態を変更することで、さまざまな処理が可能になります。

For 文の利用

```
1 sum = 0
2 For k = 1 To 100
3   sum = sum + k*k
4 EndFor
5 TextWindow.WriteLine("合計 = " + sum)
```

上のプログラムの sum = sum + k*k は、左辺の変数 sum の中身を、右辺の式 sum + k*k の値で置き換えるという意味です。つまり、For 文の処理を 1 回するごとに、変数 sum の値が、k*k 加算されたものに更新されていくため、合計として表示されるのは、 $1^2 + 2^2 + \dots + 100^2 = 338350$ です。

出力

```
合計 = 338350
```

(5.6) 課題 14 – 数列の和 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「数値を入力」と表示する。
- (2) 数値を入力する (入力した数値をここでは λ とおく)。
- (3) 1 から λ までの逆数の和を計算する。
- (4) 求めた和を「合計 = 」と表示する。

出力例

```
数値を入力
82
合計 = 4.9900200799090817163789271602
```

ただし、上の出力例では、2 行目は人間がタイプしたものです。

(5.7) 課題 15# – 階乗 (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「数値を入力」と表示する。
- (2) 数値を入力する。
- (3) 入力した数値の階乗を計算して、例えば「6 の階乗は 720 です」と表示する。

出力例

```
数値を入力
10
10 の階乗は 3628800 です
```

ただし、上の出力例では、2 行目は人間がタイプしたものです。

(5.8) 課題 16# – フィボナッチ数列 (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「数値を入力」と表示する。

- (2) 数値を入力する (入力した数値をここでは λ とする)。
- (3) フィボナッチ数列の第 1 項から第 λ 項までを表示する。

ここで、フィボナッチ数列は、漸化式

$$\begin{aligned} F_1 &= 1, \\ F_2 &= 1, \\ F_{n+2} &= F_{n+1} + F_n \end{aligned}$$

で定められる。

出力例

```
数値を入力
10
1 1 2 3 5 8 13 21 34 55
```

ただし、上の出力例では、2 行目は人間がタイプしたものです。

§6 While 文による繰り返し

(6.1) While 文 条件成立の間処理を繰り返すには While 文を用います。For 文との違いは、

- For 文はあらかじめ反復回数が決まっているが、
- While 文は、反復回数が事前にはわからない

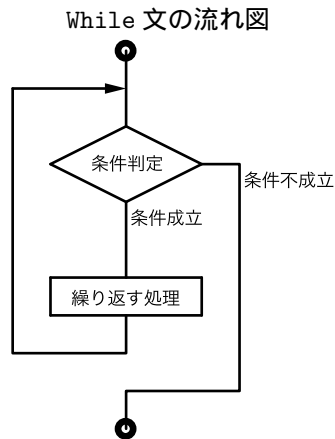
という点です。次のような構造を「反復」と呼びます¹⁵¹⁶。

繰り返し (While)

```
While 条件
  条件成立の間繰り返し処理
EndWhile
```

¹⁵For 文も広い意味での「反復」の構造と見なせます。

¹⁶最近では避けられる傾向にもありますが、処理の流れを流れ図 (フローチャート) で表すと便利ことがあります。If 文などの条件判定はひし形で、通常の処理は長方形で図示し、処理の流れを線で結びます。



次の例では、貯金の合計が 100 円を超えると、繰り返しが終了します。

While の例

```

1 sum = 0
2 While sum <= 100
3   TextWindow.WriteLine("貯金せよ")
4   money = TextWindow.ReadNumber()
5   sum = sum + money
6 EndWhile
7 TextWindow.WriteLine(sum + "円貯まりました")
    
```

下の実行例では、「人間の入力」とある行は、人間がタイプしたものです。

出力例

```

貯金せよ
30          ' 人間の入力
貯金せよ
45          ' 人間の入力
貯金せよ
30          ' 人間の入力
105円貯まりました
    
```

(6.2) 課題 17 – 割れるだけ割る – 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「数値を入力」と表示する。
- (2) 数値を入力する (入力した数値をここでは λ とする)。
- (3) $\lambda = 2^n \cdot a$ (a は奇数) と書いたときの a を、「 λ を 2 で割れるだけ割ると a です」と表示する。

a は、「 λ が偶数である間、 λ を $\lambda/2$ に置き換える」という繰り返しの繰り返しを行うと求められます。

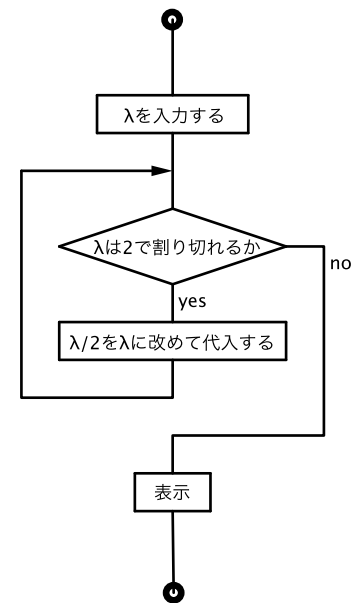
出力例

```

数値を入力
240
240を2で割れるだけ割ると15です
    
```

ただし、上の出力例では、2 行目は人間がタイプしたものです。

課題 17 の流れ図



(6.3) 無限ループ While 文の条件に、常に成立する条件 "true" (真)¹⁷を書くと¹⁸、いつまでもループし続ける、無限ループになります。

無限ループ

```
1 TextWindow.WriteLine("2乗計算機はじまり")
2 While "true"
3   a = TextWindow.ReadNumber()
4   TextWindow.WriteLine("2乗は" + a*a)
5 EndWhile
```

出力例

```
2乗計算機はじまり
3
2乗は9
5
2乗は25
-4
2乗は17
:
```

ただし、上の出力例では、2, 4, 6 行目は人間がタイプしたもので、ウィンドウを閉じない限りプログラムは停まらないので、途中までを例示してあります。

(6.4) 諸悪の根源 Goto これから学ぶ Goto 文は、みだりに使わないで下さい¹⁹。読み易いプログラムのためには Goto 文は害悪であり、代わりに、「順次」、「分岐」、「反復」という 3 つの基本構造の組合せでプログラムを記述するというのが、構造化プログラミングの考えです²⁰。

Goto 文は、プログラム中に定義されたラベルの位置にプログラムの制御を移す命令です。

¹⁷逆に、常に成立しない条件 (偽) は "false" です。

¹⁸"true"でなくても、1<2 などの常に成立する条件を書くと無限ループになります。

¹⁹過去には、使う、使わないで大論争を巻き起こしてもいますし、現在でもすぐに論争になります。

²⁰If 文、For 文、While 文において、インデントを用いてプログラムを書くことも、構造化プログラミングの考えに沿ったものです。

Goto

```
(A)
Goto hogehoge
(B)
hogehoge:
(C)
```

上の hogehoge のように、後ろにコロンを付けた文字列がラベルとして認識されます。上のプログラムだと (A) を実行した後、Goto 文で hogehoge: の位置に飛び、(C) を実行します。つまり、(B) は実行されません。

(6.5) ループからの脱出 While 文は、ループの開始時点で条件を判断しますが、それでは不便なこともあるので、Goto 文を用いて強制的にループを脱出することができます²¹。

次の例は、無限ループだった (6.3) とほぼ同じですが、0 を入力すると停止する点が異なります。

ループ脱出の例

```
1 TextWindow.WriteLine("停まる2乗計算機はじまり")
2 While "true"
3   a = TextWindow.ReadNumber()
4   If a = 0 Then
5     Goto finish
6   EndIf
7   TextWindow.WriteLine("2乗は" + a*a)
8 EndWhile
9 finish:
10 TextWindow.WriteLine("おわり")
```

出力例

```
停まる2乗計算機はじまり
3
2乗は9
```

²¹Small Basic ではループ脱出のための専用の命令がないので、仕方なく Goto 文を用いますが、ループ脱出のための命令 (たいてい break という命令) を持つプログラミング言語も多いです。

```
5
2乗は25
0
おわり
```

ただし、上の出力例では、2, 4, 6 行目は人間がタイプしたものです。

(6.6) 課題 18 – 入力した数の合計 – 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「貯金せよ」と表示する。
- (2) 数値を入力する。
- (3) 入力した数値が 0 でなければ、(1) に戻る。
- (4) 入力した数値が 0 ならば、今までの入力の合計を、「合計 = 」と表示する。

出力例

```
貯金せよ
30
貯金せよ
45
貯金せよ
30
貯金せよ
0
合計 = 105
```

ただし、上の出力例では、2, 4, 6, 8 行目は人間がタイプしたものです。

(6.7) 課題 19# – ユークリッドの互除法 (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「数値を 2 つ入力」と表示する。
- (2) 数値を 2 回入力する (入力した数値をここでは λ, μ とする)。
- (3) λ, μ の最大公約数を求め、「最大公約数 = 」と表示する。

ここで、最大公約数を求めるには、

- (i) λ を μ で割った余りを r と置く。
- (ii) $r = 0$ ならば、 μ は λ の約数になるので、最大公約数は μ 。
- (iii) $r \neq 0$ ならば、 $\lambda = \mu, \mu = r$ と置き直して、最初に戻る。

というユークリッドの互除法を用いるとよい。

出力例

```
数値を2つ入力
72
120
最大公約数 = 24
```

§7 サブルーチン

(7.1) サブルーチン サブルーチンとは、プログラムを実行したときの処理の流れ (メインルーチン) から独立させた、プログラム断片です。サブルーチンには、一般に次のような利点があります²²。

コード量の減少 複数回実行する処理を何度も記述するのではなく、サブルーチンにして呼び出すことで、コード量が減少する。

保守性 複数回実行する処理を何度も記述すると、修正を施すときに、間違いなくすべての該当箇所を修正するのが難しいが、サブルーチンに独立しておけば、修正箇所はサブルーチン中の 1 箇所済む。

可読性 メインルーチンに複雑な処理があると、全体として何をしているか把握しづらいが、意味のあるまとまりをサブルーチンにしてメインルーチンから呼び出すようにすると、処理内容を把握し易くなる。

再利用性 意味のあるまとまりをサブルーチンしておけば、別のプログラムで同じ処理が必要になった場合、再利用できる。

²² Small Basic には変数のスコープがないため、ここにあげた利点のうち、再利用性と実装の隠蔽については、十分に恩恵を受けられません。スコープとは変数の有効範囲のことで、Small Basic では、一度定義された変数はプログラム全体で有効です。つまりサブルーチンの中で変数を変更すると、メインルーチンの同名の変数は同一の変数なので変更されてしまいます。

実装の隠蔽 何らかの状態を表す変数を直接変更したり参照したりするのではなく、サブルーチンを介してのみ変更・参照を許すなど、触れて欲しくない部分に触れないようにできる。将来サブルーチンのコードを改良して、状態を表す変数の役割が変わってしまったりしても、サブルーチンを変更するだけで呼び出し側のコードは変更せずに済む。

(7.2) サブルーチンの定義と呼び出し Small Basic では次のようにサブルーチンを定義します。定義の場所は、いわゆるトップレベルならばどこでも構いませんが、メインルーチンの前か後でまとめて定義するのが良いでしょう。

サブルーチンの定義

```
Sub サブルーチン名
  処理 (複数行でもよい)
EndSub
```

定義されたサブルーチンの呼び出しは次のように行います。

サブルーチンの呼び出し

```
サブルーチン名()
```

下の例では、サブルーチン「Hantei」を定義して2回呼び出しています。

サブルーチンの例

```
1 TextWindow.WriteLine("あなたの点数を入力")
2 point = TextWindow.ReadNumber()
3 Hantei()
4 TextWindow.WriteLine("隣の人の点数を入力")
5 point = TextWindow.ReadNumber()
6 Hantei()
7
8 Sub Hantei
9   If point >= 60 Then
10    TextWindow.WriteLine("合格")
11   Else
12    TextWindow.WriteLine("不合格")
```

```
13 EndIf
14 EndSub
```

出力

```
あなたの点数を入力
80
合格
隣の人の点数を入力
55
不合格
```

ただし、上の出力例では、2, 5 行目は人間がタイプしたものです。

(7.3) 課題 20 – 誕生日占い 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。ただし、2 回ある表示する部分は、サブルーチンにすること。

- (1) 「あなたの誕生日を入力」と表示する。
- (2) 誕生日の数値を入力する。
- (3) 2 月生まれならば「素晴らしい」、それ以外は「ごく普通」と表示する。
- (4) 「隣の人の誕生日を入力」と表示する。
- (5) 誕生日の数値を入力する。
- (6) 2 月生まれならば「素晴らしい」、それ以外は「ごく普通」と表示する。

出力例

```
あなたの誕生日を入力
2
素晴らしい
隣の人の誕生日を入力
3
ごく普通
```

ただし、上の出力例では、2, 5 行目は人間がタイプしたものです。

(7.4) サブルーチンの計算結果の利用 下のプログラムでは、「Fibonacci」という名前のサブルーチン²³を 2 回呼び出しています。

サブルーチンの例

```

1 TextWindow.WriteLine("数を2つ入力")
2 n = TextWindow.ReadNumber()
3 Fibonacci()
4 a1 = a ' 一旦退避
5 n = TextWindow.ReadNumber()
6 Fibonacci()
7 TextWindow.WriteLine(a1 + ", " + a)
8
9 Sub Fibonacci
10     x = Math.Power(1.618034, n)
11     y = Math.SquareRoot(5)
12     a = Math.Round(x / y)
13 EndSub

```

Math.Round(x) は、x を四捨五入した整数を返すメソッド、Math.Power(x, n) は、x の n 乗を返すメソッドです。

4 行目の「a1 = a」は、6 行目の「Fibonacci()」を実行したときに変数 a が変更されるので、その前に計算結果を保存するためのコードです。

出力例

```

数を2つ入力
4
5
3, 5

```

ただし、上の出力例では、2, 3 行目は人間がタイプしたものです。

(7.5) 課題 21 — フィボナッチ数列 上のプログラム例で定義されているサブルーチン「Fibonacci」を変更なしに再利用して、次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

²³フィボナッチ数列の第 n 項を計算して、変数 a に設定するサブルーチンです。この計算方法を課題 16 に用いてはいけません。ちなみに、この計算方法を即座に理解できる人は少ないと思いますが、入力に対して所望の出力 (今の場合はフィボナッチ数列の項) を返すことさえわかっていたら、ブラックボックスとして利用できます。これもサブルーチンの特徴のひとつです。

- (1) 「数値を入力」と表示する。
- (2) 数値を入力する (入力した数値をここでは λ とする)。
- (3) フィボナッチ数列の第 1 項から第 λ 項までを表示する。

出力例

```

数値を入力
10
1 1 2 3 5 8 13 21 34 55

```

ただし、上の出力例では、2 行目は人間がタイプしたものです。また、「Press any key to continue ...」が数列の直後に表示されるのは格好が悪いので、数列の次の行に表示されるようにすること。

§8 いろいろな関数

(8.1) 数学関数の一覧 Math オブジェクトに属する数学関数の一覧を、以下に示します。Small Basic のエディタで「Math.」とタイプすると、インテリセンスで確認することもできます。Math.Pi のみプロパティです (つまり後続の括弧が不要)。

式	戻り値
Math.Pi	円周率
Math.Cos(x)	$\cos x$
Math.Sin(x)	$\sin x$
Math.Tan(x)	$\tan x$
Math.ArcCos(x)	$\cos^{-1} x$
Math.ArcSin(x)	$\sin^{-1} x$
Math.ArcTan(x)	$\tan^{-1} x$
Math.Log(x)	$\log_{10} x$
Math.NaturalLog(x)	$\log_e x$
Math.Max(x, y)	x と y の最大値
Math.Min(x, y)	x と y の最小値
Math.Ceiling(x)	x 以上の整数のうち最小のもの
Math.Floor(x)	x 以下の整数のうち最大のもの
Math.Round(x)	x を四捨五入した整数
Math.Abs(x)	x の絶対値
Math.Power(x, y)	x^y
Math.SquareRoot(x)	\sqrt{x}
Math.Remainder(x, y)	x を y で割った余り
Math.GetRandomNumber(max)	1 から max の乱数
Math.GetDegrees(x)	x ラジアンを度に変換した値
Math.GetRadians(x)	x 度をラジアンに変換した値

三角関数、逆三角関数の引数や戻り値の単位はラジアンですので、度数法との間で変換したいときは、Math.GetDegrees や Math.GetRadians を使います。

(8.2) 文字列関数の一覧 Text オブジェクトに属する文字列関数の一覧を、以下に示します。Small Basic のエディタで「Text.」とタイプすると、インテリセンスで確認することもできます。

式	戻り値
Text.Append(text1, text2)	2 つの文字列の連結
Text.GetLength(text)	文字列の長さ
Text.ConvertToLowerCase(text)	小文字に変換した文字列
Text.ConvertToUpperCase(text)	大文字に変換した文字列
Text.GetCharacter(code)	文字コードが code の文字
Text.GetCharacterCode(ch)	文字 ch の文字コード
Text.IndexOf(text, subText)	文字列 text 中の subText の出現位置。現れなければ 0。
Text.GetSubText(text, start, len)	文字列 text の位置 start から len 文字分の文字列
Text.GetSubTextToEnd(text, start)	文字列 text の位置 start から最後まで文字列
Text.IsSubText(text, subText)	文字列 text 中に subText が現れれば True。
Text.StartsWith(text, subText)	文字列 text が subText から始まっていれば True
Text.EndsWith(text, subText)	文字列 text が subText で終わっていれば True

上の表で、文字コードは Unicode の文字コードです。Text.Append は演算子「+」と同じ動作ですが、文字列が数値を表しているときにも、数値として解釈して和を計算するのではなく、文字列として連結をします。

(8.3) 課題 22# — 大きい数の桁数 (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「a の b 乗の a と b を入力」と表示する。
- (2) 2 数を入力する。(入力した 2 数を、ここでは α, β とする)
- (3) α^β の桁数を求め、「 α の β 乗は 桁です」と表示する。

出力例

```
aのb乗のaとbを入力
7
1234
7の1234乗は1043桁です
```

ただし、上の出力例では、2, 3 行目は人間がタイプしたものです。また、上のように、1000 桁程度の結果も正しく表示されるようにすること²⁴。

(8.4) 課題 23# — 部分文字列 (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「3 文字以上の文字列を入力」と表示する。
- (2) 文字列を入力する (入力した文字列をここでは λ とする)。
- (3) λ が 3 文字に満たなければ、何もしない。
- (4) λ が 3 文字以上ならば、「先頭から 3 文字は です」、「末尾から 3 文字は です」と表示する。

出力例その 1

```
3文字以上の文字列を入力
hokkyodai
先頭から3文字はhokです
末尾から3文字はdaiです
```

出力例その 2

```
3文字以上の文字列を入力
tt
```

ただし、上の各出力例では、2 行目は人間がタイプしたものです。

§9 配列

今学期はやりません。

²⁴ ヒント: 対数を使えば 1000 桁程度の結果にも対応できます。

§10 グラフィックス

今学期はやりません。

§11 練習問題

(11.1) 課題 24# — カウントダウン (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「秒数を入力」と表示する (入力した数値を、ここでは λ とする)。
- (2) λ から始めて、1 ずつ減少しながら 0 まで、数値を 1 秒おきに表示する。

出力例

```
秒数を入力
5
5
4
3
2
1
0
```

ただし、上の各出力例では、2 行目は人間がタイプしたものです。3 行目以降の出力は、一斉に表示されるのではなく、1 秒おきに表示されるようにすること。

`Program.Delay(msec)` を用いると、msec ミリ秒何もせずに停止するので、これを用いると 1 秒間隔の表示が実現できます。また、変数が減少しながらの繰り返しは、For 文の Step に負の値 (今の場合ならば -1) を指定しても実現できますし、 i が 0 から λ まで増加するとき、 $\lambda - i$ は λ から 0 まで減少することを用いても実現できます。

(11.2) 課題 25# — 約数列挙 (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「整数を入力」と表示する (入力した数値を、ここでは λ とする)。
- (2) λ の正の約数をすべて表示する。

出力例

```
整数を入力
12
約数は 1 2 3 4 6 12
```

ただし、上の各出力例では、2 行目は人間がタイプしたものです。

- (11.3) 課題 26# — 最大の約数 (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「整数を入力」と表示する (入力した数値を、ここでは λ とする)。
- (2) λ の約数のうち、 λ 未満で最大のものを表示する。

出力例

```
整数を入力
12
最大の真の約数は 6
```

ただし、上の各出力例では、2 行目は人間がタイプしたものです。

- (11.4) 課題 27# — 公約数 (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「整数を 2 つ入力」と表示する。
- (2) 2 つの整数の公約数をすべて表示する。

出力例

```
整数を2つ入力
12
18
公約数は 1 2 3 6
```

ただし、上の各出力例では、2, 3 行目は人間がタイプしたものです。

- (11.5) 課題 28# — 素因数分解 (義務ではない) 次のような動作をするプログラムを作成し、.sb ファイルを提出して下さい。

- (1) 「整数を入力」と表示する。
- (2) 入力された整数の素因数分解を表示する。

出力例

```
整数を入力
1800
素因数分解は 2 2 2 3 3 5 5
```

ただし、上の各出力例では、2, 3 行目は人間がタイプしたものです。

3 授業のノート (T_EX 編)

§1 文書組版システム T_EX の導入

T_EX(テフ, テック)とは「組版ソフトウェア」です。組版とは活字を組んで版を作ることなので、これは文書を作成するソフトウェアを意味しています。以下に T_EX の大事な特徴をあげます。

- フリーソフトウェアであり、かつ、あらゆる OS で動作する。
- 文書の論理構造を明確にでき、相互参照も強力。
- 数式が美しい。

これらほど大事ではない特徴も脚注²⁵ や傍注 にあげます。

- 高精度
- 目次が作れる
- 索引が作れる
- 1980 年頃に誕生
- 自動で hyphenation
- office ではなく office になる

大学のコンピュータ室や、数学研究室のコンピュータには、いくつかある T_EX の亜種のうち、最も広く使われている L^AT_EX2_ε (ラテフツーイー、ラテックツーイー) が導入されています。自宅のコンピュータでも T_EX を利用したい場合、以下のような方法があります。上ほどお勧めです。

- (1) 書籍「[改訂第5版]LaTeX2e 美文書作成入門」(ISBN 978-4-7741-4319-4)の付録 DVD-ROM からインストールする。Windows にも Macintosh にも対応。
- (2) 阿部紀行氏の Windows 用 TeX インストーラを利用する²⁶。
<http://www.math.sci.hokudai.ac.jp/~abenori/>
- (3) 角藤亮氏のサイト
<http://www.fsci.fuk.kindai.ac.jp/kakuto/win32-ptex/> の Windows 用簡易インストーラを利用する。
- (4) Macintosh の場合、桐木紳氏のサイト
<http://math.kyokyo-u.ac.jp/~kiriki/ptex/> や、小川弘和氏のサイト
<http://www2.kumagaku.ac.jp/teacher/herogw/> から入手する。

²⁵ どの OS や、どのコンピュータでも出力が同一。

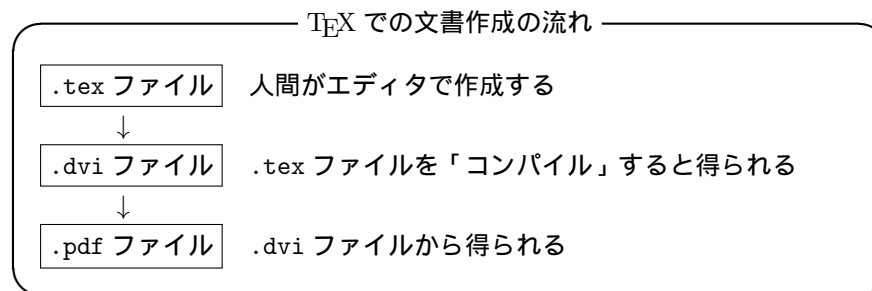
- T_EX 方式の数式入力方法を採用するソフトウェアが多い。
- T_EX で書かれた本も多い。
- WYSIWYG (what you see is what you get. 画面のままの印刷が得られる) ではなく「バッチ式」であり、この点は HTML に近い。

²⁶ただし、惜しいことに texworks の日本語環境が整わない。いつの間にか対応してくれるかも知れませんが。

- (5) Linux の場合 apt-get (synaptic) などを利用した簡単なインストール方法が用意されている。始めからインストールされていることも多い。

§2 T_EX の使用方法

T_EX は Microsoft Word などとは異なり、入力から印刷までを単一のソフトウェアで行うわけではありません。Pdf ファイルで閲覧・印刷を行う場合、その過程は以下ようになります。



実際には、これらを統合する TeXworks というソフトウェアがインストールされており、複数のプログラムが動いていることは意識せずとも T_EX を利用できるようになっています。TeXworks を起動するには (他にも方法はありますが)、Windows XP ならば、スタートメニューから「ファイル名を指定して実行」に、「texworks」と入力し Enter を押します。Windows Vista 以降ならば、スタートメニューから「検索を開始」の所に「texworks」と入力して検索してプログラムの TeXworks を実行します。

- (2.1) 課題 51 – 最初の T_EX 文書 L^AT_EX2_ε の書式に従った最小の tex ファイルは次のようになります。バックスラッシュ (\) はキーボードの円記号 (¥) のキーを押すと入力でき、コンピュータの環境によっては画面上でも円記号です。

0000 和地-51.tex

```

1 | \documentclass{jarticle}
  
```

```

2 \begin{document}
3 夏空と駆ける先には蝸牛
4 \end{document}

```

`\documentclass{jarticle}` は、「jarticle」という文書クラスで文書を作成するという命令です。用途に応じて次のような文書クラスがあります。

用途	欧文	和文横書き	和文縦書き	新和文横書き
論文・レポート	article	jarticle	tarticle	jsarticle
長い報告書	report	jreport	treport	(ない)
本	book	jbook	tbook	jsbook

また、`\documentclass[a4j,12pt]{jarticle}` のように、オプションを 1 つ、またはカンマ区切りで複数個与えることができます²⁷。

`\begin{document}` ~ `\end{document}` は document 環境で、この中に書かれたものが出力されます。

次の作業の結果得られた pdf ファイルを提出して下さい。

- (1) 上の例の俳句を他のものに替えて 0000 和地-51.tex のようなファイル名で保存する。
- (2) コンパイルして pdf ファイルを作成し、0000 和地-51.pdf のようなファイル名の pdf ファイルを提出する。

(2.2) コメント tex ファイル中で、パーセント記号 (%) から行末までは無視されます (改行も無視)。これはコメントと呼ばれ、出力はしたくないけれど、tex ファイル中に残しておきたいことなどを書けます。

入力

```

1 \documentclass{jarticle}
2 \begin{document}
3 夏空と駆ける先には蝸牛 % 字余り
4 \end{document}

```

²⁷ 例えば次のようなオプションが利用可能です。a4j, b5j: 用紙サイズの指定。無指定の時はやや余白の広い a4paper を指定したことになる。landscape: 用紙を横置きで使う指定。10pt, 11pt, 12pt: フォントのサイズの指定。twocolumn: 2 段組にする指定。

出力

夏空と駆ける先には蝸牛

(2.3) 改行や空白文字の扱い Microsoft Word では、改行したりスペースを入力すれば、その通りの画面表示と印刷結果になります。しかし、 $\text{T}_{\text{E}}\text{X}$ での扱いはこれとは異なり次のようになります。

- 行末が和文文字 (全角) の時、改行は無視される。
- 行末が欧文文字 (半角) の時、改行は空白 (半角スペース) と解釈される。
- 連続する空白 (半角スペース) は 1 つの空白と同等。
- 行頭や行末の空白 (半角スペース) は何個あっても無視される。
- 空白だけの行 (空行) は、段落の切れ目になる。

このように、半角スペースは良い組版をするため特別の扱いを受けますので、全角スペースで空白を空けることは必要でなければいけない方がよいです。

入力

```

1 \documentclass{jarticle}
2 \begin{document}
3 tex ファイルを書くときは、
4 適宜改行を入れて見易く整えるのが
5 習慣になっています。
6
7 She wore a yellow ribbon.
8 \end{document}

```

出力

tex ファイルを書くときは、適宜改行を入れて見易く整えるのが習慣になっています。

She wore a yellow ribbon.

(2.4) 初めての命令 `\documentclass{jarticle}` のように \ (バックスラッシュ)。環境によっては $\$ の付いたものが \TeX の命令です。例えば、`\TeX` という命令は、 \TeX を出力します。末尾が英字や和字である命令の後の空白については、次の例のように注意が必要です。

入力

```
1 \documentclass{jarticle}
2 \begin{document}
3 \TeX is joyful. % 命令の後の空白は無視される
4 \TeX{} is joyful. % こうすればよい
5 {\TeX} is joyful. % あるいはこう
6 \end{document}
```

出力

\TeX is joyful. \TeX is joyful. \TeX is joyful.

反対に、 \TeX の直後に空白なしに続けたいときに、

`\TeX の楽しみ % エラー`

と入力すると「`\TeX の楽しみ`」までで 1 つの命令とみなされてエラーになります。

(2.5) 特殊文字 パーセント記号を出力したいとき、単に `tex` ファイルに書いてもコメントの始まりとみなされて出力されません。バックスラッシュ (円記号) などと同様で、このような特殊文字には以下ものがあります。

`# $ % & _ { } \ ^ ~ < > |`

これらのうちいくつかは、バックスラッシュを付加すると次のように出力できます。

入力

```
1 \documentclass{jarticle}
2 \begin{document}
3 \# \$ \% \& \_ \{ \}
4 \end{document}
```

出力

`# $ % & _ { }`

また別の方法として、入力をそのまま出力する `\verb` 命令を用いることもできます。`\verb|foo|` のように、任意に選んだ同じ文字 (ここでは `|`) で挟まれた部分をそのまま出力します。

入力

```
1 \documentclass{jarticle}
2 \begin{document}
3 \verb@|@ \verb|$29 > $28|
4 \end{document}
```

出力

`\ $29 > $28`

また、`< > |` は後述の数式モードでは出力できます。

(2.6) 課題 52 – 特殊文字の出力 次のような出力を与える `tex` ファイル (ファイル名は `0000 和地-52.tex` のように) を作り提出して下さい。ただし、わずかな (無害の) スペースの有無などにより、作成したファイルの出力結果と下の出力例を比べたときに、 \TeX が自動的に決定する行の折り返し位置が異なる場合がありますが、それは構いません。逆に、改行位置を下の出力例に合わせるために、無理な操作をするのは良くありません。全角スペースの利用は特に有害です。

出力

\TeX で `%` を出力するためには `\%` と入力します。`\verb` 命令を使った場合の出力は `%` ですが、書体が異なります。

上の出力では、1 つ目の `%` と他の `%` は、`tex` ファイルでの記述が異なるため、書体が違うことにも注意して下さい。最初の \TeX は、単に `TeX` と入力するわけではありません (少し前を見て下さい)。また、上の出力では掲載を省略し

ていますが、用紙の下部にページ番号も出力されています。

§3 文書の構造

文書には、章があり、その中にいくつかの節 (セクション) があり、さらに、小節 (サブセクション) や段落を含むといった論理構造があります。他方、章の見出しは文字サイズを大きくし、節の見出しはそれより少し小さくし、また、これらの見出しは太字にするといった物理構造 (レイアウト) もあります。TeX では、論理構造の指定のみ行い、物理構造の指定は陽には行わないことが推奨されます。つまり、「文字サイズを大きくする」と指定するのではなく、「ここから節を始める」と指定します。どのような物理構造になるかは `jarticle` などの文書クラスによって決まります。

(3.1) 論理構造を指定する命令 `jarticle` クラスで論理構造を指定するには、主に次のような命令があります。ただし、段落については、空行を作れば段落の区切りになります。段落に見出しの必要がないときは、普通は `\paragraph` 命令を用いずに、空行で段落を作成します。

命令	意味	使用方法
<code>\section</code>	節	<code>\section{節の名前}</code>
<code>\subsection</code>	小節	<code>\subsection{小節の名前}</code>
<code>\subsubsection</code>	小小節	<code>\subsubsection{小小節の名前}</code>
<code>\paragraph</code>	段落	<code>\paragraph{段落の名前}</code>

入力

```

1 \documentclass{jarticle}
2 \begin{document}
3 \section{論理構造を指定する命令}
4 \subsection{節と小節}
5 \verb@\section@命令で節、
6 \verb@\subsection@命令で小節の開始を指定します。
7 \subsection{段落}
8 単に空行を作るだけでも段落を開始できますが、

```

```

9 \verb@\paragraph@命令ならば
10 見出し付きの段落を開始します。
11
12 段落先頭のインデントは自動で付くため、
13 入力の必要はありません。
14 見出しの節番号も自動で付きます。
15 \end{document}

```

出力

1 論理構造を指定する命令

1.1 節と小節

`\section` 命令で節、`\subsection` 命令で小節の開始を指定します。

1.2 段落

単に空行を作るだけでも段落を開始できますが、`\paragraph` 命令ならば見出し付きの段落を開始します。

段落先頭のインデントは自動で付くため、入力の必要はありません。見出しの節番号も自動で付きます。

また、上の出力では掲載を省略していますが、用紙の下部にページ番号も出力されています。

(3.2) 物理構造を指定する命令 (下線、改ページ、改行) TeX では物理構造を陽に指定しないことが推奨されるものの、実際には文字サイズなどの物理構造を指定したいこともあります。

文章の一部に下線を引くこと、ページを改めること (改ページ)、行を改めること (改行) はどれも物理構造の指定ですが、次の命令で実現できます。

入力	出力
<code>\underline{下線}</code>	<u>下線</u>
<code>\newpage</code>	改ページする
<code>\\</code>	改行する

TeX が自動的に決定する改行位置とは無関係に、強制的に改行したいとき `\\` 命令を使います。しかし、(数式などではない地の文で) 強制的に改行したいのは、人間の方が間違っていることが多いので、`\\` 命令は極力使わないようにします。

また、`\\[1cm]` のように後ろにブラケットがあると、囲まれた寸法だけ行を送ります。この命令も、微調整が必要な時にはやむを得ないこともありますが、極力使わないようにします。`\\` の後にブラケットを書きたい場合は、こう解釈されないように、何もしない命令 `\relax` を用いて、`\\ \relax []` とするなどの注意が必要です。

(3.3) 物理構造を指定する命令 (文字サイズ、書体) 文字サイズも物理構造です。次の命令で文字サイズが変更できます。

命令	見本
<code>\tiny</code>	サイズ Sample
<code>\scriptsize</code>	サイズ Sample
<code>\footnotesize</code>	サイズ Sample
<code>\small</code>	サイズ Sample
<code>\normalsize</code>	サイズ Sample
<code>\large</code>	サイズ Sample
<code>\Large</code>	サイズ Sample
<code>\LARGE</code>	サイズ Sample
<code>\huge</code>	サイズ Sample
<code>\Huge</code>	サイズ Sample

サイズ変更の影響範囲を限定するためには、

`{\small 小さい}` 普通

のようにブレース (`{ }`) で囲みます。

入力

```

1 \documentclass{jarticle}
2 \begin{document}
3 {\small ここだけ小さい、}
4 普通の大きさ、
5 \large ここだけ大きい、
6
7 つもりだったが以降ずっと大きい。
8 \end{document}

```

出力

ここだけ小さい、普通の大きさ、ここだけ大きい、
つもりだったが以降ずっと大きい。

また、次の命令で文字の書体が変更できます。上の `\small` などの命令とは、ブレースの付き方が違うことに注意が必要です。

入力	意味	出力
<code>\textrm{Roman}</code>	ローマン	Roman
<code>\textbf{Bold}</code>	ボールド	Bold
<code>\textit{Italic}</code>	イタリック	<i>Italic</i>
<code>\textsl{Slant}</code>	斜体	<i>Slant</i>
<code>\textsf{Sans Serif}</code>	サンセリフ	Sans Serif
<code>\texttt{Typewriter}</code>	タイプライター	Typewriter
<code>\textsc{Small Caps}</code>	スモールキャップス	SMALL CAPS
<code>\textgt{ゴシック体}</code>	ゴシック	ゴシック体

(3.4) 課題 53 – 文書の論理構造と物理構造 次のような出力を与える tex ファイル (ファイル名は 0000 和地-53.tex のように) を作り提出して下さい。

出力

1 節など

1.1 節・小節・段落

節を始めるには`\section` 命令を使い、小節を始めるには`\subsection` 命令を使います。

このように新しい段落を始めるためには、単に空行を作ればよいです。また、行の途中で改行もできます。

1.2 見出し付き段落

段落に見出しを付けたいときは、`\paragraph` 命令を使います。

見出し付き段落の例 この段落の上方直前の空きは、`\paragraph` 命令を使うと自動的に付くものです。

2 文字サイズや書体

2.1 文字サイズ

例えば`\footnotesize` 命令を使うと、小さい文字になります。

2.2 書体

例えば`\textsf` 命令を使うと、read me!のような書体になります。太字で大きい文字サイズにしたいときは、`\textbf` と `\Large` を組み合わせると、**like this** のように出力されます。

上の出力では掲載を省略していますが、用紙の下部にページ番号も出力されています。ただし、 $\text{T}_{\text{E}}\text{X}$ が自動的に決定する行の折り返し位置が、無害な半角スペースなどで多少異なっても構いません。

§4 箇条書き

(4.1) 環境 `\begin{document} \dots \end{document}` のように`\begin` と `\end` で囲まれた構造を環境と呼びます。document 環境の他には、

環境名	意味
<code>quote</code> 環境	引用を行う。段落の先頭はインデントしない。
<code>quotation</code> 環境	引用を行う。段落の先頭をインデントする。
<code>flushright</code> 環境	右寄せする。
<code>flushleft</code> 環境	左寄せする。
<code>center</code> 環境	中央揃えする。

などもあります

入力

```

1 \documentclass{jarticle}
2 \begin{document}
3 \begin{quote}
4   引用します
5 \end{quote}
6 \begin{quotation}
7   引用します。インデントもします。
8 \end{quotation}
9 \begin{flushleft}
10  \Large 大きい字で左寄せ
11 \end{flushleft}
12 \begin{flushright}
13   右寄せ
14 \end{flushright}
15 \begin{center}
16 中央揃え
17 \end{center}
18 \end{document}

```

出力

引用します

引用します。インデントもします。

大きい字で左寄せ

右寄せ

中央揃え

上の例でわかるように、環境は{...}と同じように、`\Large` 命令の影響範囲を限定しています。

(4.2) 箇条書き環境 次のような、意味の異なる箇条書き環境があります。

環境名	意味
<code>itemize</code>	番号の付かない箇条書き
<code>enumerate</code>	番号の付く箇条書き
<code>description</code>	見出しの付く箇条書き

これらの環境の本体では、`\item` 命令に続いて箇条書きする項目を書きます。また、`description` 環境では、`\item[見出し]` として見出しを指定します。`itemize` と `enumerate` 環境でもこうすると見出しを変更できます。

入力

```

1 \documentclass{jarticle}
2 \begin{document}
3 \begin{itemize}
4   \item かき氷とともに頬ばる日进行
5   \item 向日葵を見上げる君の大き顔
6 \end{itemize}
7 \begin{enumerate}
8   \item 母の持つ砂場の蚯蚓に後ずさる
9   \item 炎天下滑りて昇りまた滑り
10 \end{enumerate}

```

```

11 \begin{description}
12   \item [仲夏] 逃げ惑い泣き疲れ蠅を夢に見る
13   \item [晩夏] 水風船親の心を子は知らず
14 \end{description}
15 \end{document}

```

出力

- かき氷とともに頬ばる日进行
 - 向日葵を見上げる君の大き顔
1. 母の持つ砂場の蚯蚓に後ずさる
 2. 炎天下滑りて昇りまた滑り

仲夏 逃げ惑い泣き疲れ蠅を夢に見る

晩夏 水風船親の心を子は知らず

また、例えば、`document` 環境の中に `itemize` 環境を入れるなど、多くの場合環境は入れ子にできます。

(4.3) 課題 54 – 箇条書きなどの環境 次のような出力を与える `tex` ファイル (ファイル名は `0000 和地-54.tex` のように) を作り提出して下さい。ただし、`TeX` が自動的に決定する行の折り返し位置が、無害な半角スペースなどで多少異なっても構いません。また、下の出力では掲載を省略していますが、用紙の下部にページ番号も出力されています。

出力

本日のまとめ

0000 和地輝仁

上のタイトルは`\Large`の大きさです。今日学んだ環境をまとめると以下のようになります。

引用 `quote`, `quotation`

寄せ・揃え `flushleft`, `flushright`, `center`

箇条書き `itemize`, `enumerate`, `description`

このうち、番号付きの箇条書きの例は下のようになります。

1. `itemize` 環境は番号なし
2. `enumerate` 環境は番号付き
3. `description` 環境は見出し付き

§5 表組み

(5.1) `tabular` 環境 表組みは `tabular` 環境で行います。

`tabular` 環境の文法

```
\begin{tabular}{列指定}
  表本体
\end{tabular}
```

列指定には、1列につき、l (左寄せ)、c (中央揃え)、r (右寄せ) のどれかを1文字指定します。これらの各文字の前後に| (縦罫線) が指定できます。

表本体において、列の区切りは&、行の終わりは\\で表します。行の先頭に`\hline`を書くと横罫線が引かれ、`\cline{2-3}`のように書くと部分的な横罫線 (この場合2から3列目まで) が引かれます。

横方向にコマを連結したいときは、`\multicolumn{連結するコマ数}{列指定}{コマの内容}` を用います。

入力

```
1 \documentclass{jarticle}
2 \begin{document}
3 \begin{tabular}{|l|c|rr|}
4 \hline \multicolumn{4}{|c|}{りんごの表} \\
5 \hline
6 \hline 品名 & サイズ & 単価 & 数量 \\
7 \hline      りんご & 大きい & 180 & 15 \\
8 \cline{2-4}      & ミニ & 90 & 8 \\
9 \hline
10 \end{tabular}
11 \end{document}
```

出力

りんごの表			
品名	サイズ	単価	数量
りんご	大きい	180	15
	ミニ	90	8

(5.2) 課題 55 – 表組み 次のような出力を与える `tex` ファイル (ファイル名は 0000 和地-55.tex のように) を作り提出して下さい。下の出力では掲載を省略していますが、用紙の下部にページ番号も出力されています。

出力

tabular 環境のまとめ		
列指定	揃え	l, c, r
	縦罫線	
表本体	列区切り	&
	改行	\\
	横罫線	\hline, \cline
	コマの連結	\multicolumn

§6 数式

(6.1) インライン数式と別行立て数式 数式モード²⁸には次の 2 つがあり、1 つの数式には一方のみを使います。

- インライン数式: $\$ \dots \$$
- 別行立て数式: $\backslash[\dots \backslash]$

つまり、

$\$ \backslash[y = -2x - 1 \backslash] \$$ (これは間違い)

のような使い方はしません。

入力

```

1 \documentclass{jarticle}
2 \begin{document}
3 テキストモードだと、
4 a = -2x - 1 という出力になってしまいますが、
5 ドルで囲んでインライン数式モードにすると、
6 $a = -2x - 1$ となります。
7 また、
8 \[ a = -2x - 1 \]
```

²⁸今まで使っていた、数式ではないモードは、テキストモードと呼ばれます。

```

9 とすると別行立て数式になります。
10 \end{document}
```

出力

テキストモードだと、 $a = -2x - 1$ という出力になってしまいますが、ドルで囲んでインライン数式モードにすると、 $a = -2x - 1$ となります。また、

$$a = -2x - 1$$

とすると別行立て数式になります。

後述の amsmath パッケージには、他にも便利な別行立て数式の環境があります²⁹。

(6.2) 累乗、添字 上付きの累乗は「 \wedge 」、下付きの添字には「 $_$ 」を用います。累乗や添字が 1 文字でない場合は、 $\{ \dots \}$ でグループ化します。累乗や添字も入れ子にできます。

入力	出力
$e^x + e^{-x}$	$e^x + e^{-x}$
$a_1^{b_1} + a_2^{b_2}$	$a_1^{b_1} + a_2^{b_2}$

(6.3) 点 1 つの点や複数の点を出力する以下のような命令があります。すべて数式モードでのみ使えます。これらを用いると前後の空白を自動調節してくれるので、全角の「 \cdot 」や「 \dots 」で代用すべきではありません。

\backslashldots と \backslashcdots の使い分けは、下の表にあるものが正当とする考えが普通ですが、日本の高校までの教科書に見られるように、すべて \backslashcdots で済ませる場合もあります。

²⁹ amsmath パッケージを使う方がおすすめです。複数行に渡る数式や、それらをイコールの位置で揃えたいとか、数式に番号を付けたいときは、多くの改善が施された amsmath パッケージを必ず使って下さい。

入力	意味	出力	使用例
<code>\cdot</code>	掛け算記号の点など	\cdot	$a \cdot b$
<code>\cdots</code>	長い演算の途中の省略	\cdots	$a + b + \cdots + z$
<code>\ldots</code>	長い列挙の途中の省略	\dots	a, b, \dots, z
<code>\vdots</code>	縦方向の省略	\vdots	
<code>\ddots</code>	斜め方向の省略	\ddots	

(6.4) 根号 「`\sqrt{根号の中身}`」でルート of 記号を出力できます。3 乗根などを出力する場合は、「`\sqrt[3]{根号の中身}`」のようにします。

入力	出力
<code>\sqrt{x^2+1}</code>	$\sqrt{x^2 + 1}$
<code>\sqrt[3]{a_1 + a_2}</code>	$\sqrt[3]{a_1 + a_2}$

(6.5) 分数 「`\frac{分子}{分母}`」で分数を出力できます。入れ子にすれば繁分数も可能です。この命令は、インライン数式と別行立て数式で文字の大きさなどが異なります。

入力	出力 (インライン)	出力 (別行立て)
<code>\frac{1}{2}</code>	$\frac{1}{2}$	$\frac{1}{2}$
<code>\frac{1+x}{\sqrt{1+x^2}}</code>	$\frac{1+x}{\sqrt{1+x^2}}$	$\frac{1+x}{\sqrt{1+x^2}}$
<code>1+\frac{2}{3+\frac{4}{5}}</code>	$1 + \frac{2}{3 + \frac{4}{5}}$	$1 + \frac{2}{3 + \frac{4}{5}}$

(6.6) シグマ記号 和のシグマ記号は「`\sum`」で出力します。「`_`」と「`^`」で、パラメータの動く範囲を書くことができます。この命令は、インライン数式と別行立て数式で記号の大きさや、範囲の付く場所が異なります。

入力	出力 (インライン)	出力 (別行立て)
<code>\sum_{k=1}^n k^2</code>	$\sum_{k=1}^n k^2$	$\sum_{k=1}^n k^2$

またシグマ記号と同様の振舞いをする大型演算子には以下のようなものもあります。

入力	出力	入力	出力	入力	出力
<code>\prod</code>	\prod	<code>\bigcap</code>	\bigcap	<code>\bigcup</code>	\bigcup

(6.7) 積分記号 積分記号は「`\int`」です。積分の上端・下端は「`^`」と「`_`」で指定します。この命令は、インライン数式と別行立て数式で記号の大きさや、範囲の付く場所が異なります。

また、そのままだと被積分関数と「 dx 」などの間が詰まりすぎるので、少しの空白を空ける命令「`\, ,`」を挿入するのが適当です。

入力	出力 (インライン)	出力 (別行立て)
<code>\int x dx % 詰まりすぎ</code>	$\int x dx$	$\int x dx$
<code>\int x \, dx</code>	$\int x dx$	$\int x dx$
<code>\int_0^{10} \sqrt{x} \, dx</code>	$\int_0^{10} \sqrt{x} dx$	$\int_0^{10} \sqrt{x} dx$

(6.8) 課題 56 – 数式その 1 次のような出力を与える tex ファイル (ファイル名は 0000 和地-56.tex のように) を作り提出して下さい。

出力

r が 1 ではないとき、等比数列の和の公式は、

$$a + ar + ar^2 + \dots + ar^{n-1} = \sum_{k=0}^{n-1} ar^k = \frac{a(1-r^n)}{1-r}$$

である。また、 $\frac{1}{\sqrt{x}}$ の不定積分は次のようになる。

$$\int \frac{1}{\sqrt{x}} dx = \int x^{-\frac{1}{2}} dx = -2x^{\frac{1}{2}} + C = -2\sqrt{x} + C$$

(6.9) log 型関数 対数関数を「 $\$ \log x \$$ 」と入力してしまうと、「 $\log x$ 」となります。しかし、数学の習慣では \log 等のできあいの関数名は斜体ではなくローマン体で書きます。また、 x の前に適切な空きもありませんから、決してこうしないで下さい。

正しくは「 $\backslash \log$ 」命令を使い、「 $\$ \backslash \log x \$$ 」とします。主な log 型の関数を以下に記します。

入力	出力	入力	出力	入力	出力
$\backslash \log x$	$\log x$	$\backslash \arg x$	$\arg x$	$\backslash \exp x$	$\exp x$
$\backslash \sin x$	$\sin x$	$\backslash \deg x$	$\deg x$	$\backslash \lim x$	$\lim x$
$\backslash \cos x$	$\cos x$	$\backslash \det x$	$\det x$	$\backslash \max x$	$\max x$
$\backslash \tan x$	$\tan x$	$\backslash \dim x$	$\dim x$	$\backslash \min x$	$\min x$

シグマ記号と同様に「 \wedge 」や「 $_$ 」を用いると、上限、下限の数式を添えられますが、関数によって、あるいは、インラインか別行立てかによって上限、下限の付く場所が異なります。

入力	出力 (インライン)	出力 (別行立て)
$\backslash \log_a x$	$\log_a x$	$\log_a x$
$\backslash \lim_{n \to \infty} a_n$	$\lim_{n \rightarrow \infty} a_n$	$\lim_{n \rightarrow \infty} a_n$

$\backslash to$ と $\backslash infty$ は、それぞれ矢印と無限大の記号を出力する命令です。

(6.10) mod mod には、使い方に合わせて 2 通りの命令があります。

入力	出力
$x \backslash bmod 3$	$x \bmod 3$
$x \backslash equiv 4 \backslash pmod{3}$	$x \equiv 4 \pmod{3}$

(6.11) 賢いカッコ 分数など縦に大きな数式に合わせて、かっこを自動的に伸縮できます。開きかっこは「 $\backslash left($ 」、閉じかっこは「 $\backslash right)$ 」とすると、その間の数式の高さに合わせてかっこが伸縮します。「 $\{ \}$ 」や「 $[]$ 」など他のかっこも「 $\backslash left, \backslash right$ 」をつけると伸縮します。ただし、ブレースは特殊文字なので、 $\backslash left\{$ ではなく $\backslash left\{$ と、バックスラッシュを前置します。また、左右の片方だけにかっこを出力したいときは、「 $\backslash left.$ 」のように、出力しないかっこの代わりにピリオドを用います。以下では別行立て数式の出力のみ掲げます。

入力	出力 (別行立て)
$\backslash \{ \backslash frac{1}{2}, \backslash frac{3}{4} \}$	$\left\{ \frac{1}{2}, \frac{3}{4} \right\}$
$\backslash left\{ \backslash frac{1}{2}, \backslash frac{3}{4} \backslash right\}$	$\left\{ \frac{1}{2}, \frac{3}{4} \right\}$
$\backslash left(\backslash frac{1}{2} \backslash right)$	$\left(\frac{1}{2} \right)$
$\backslash left[\backslash frac{1}{2} \backslash right.$	$\left[\frac{1}{2} \right]$

(6.12) 数式の表組み 地の文では tabular 環境を用いて表組みしましたが、数式モードでは array 環境を用います。列指定などは tabular 環境のときと同じです。

入力

```

1 \[
2 \left(
3 \begin{array}{cc}
4 1 & 2 \\
5 3 & 4
6 \end{array}
7 \right),
8 \left\{
9 \begin{array}{l}
10 2x + 3y = 5 \\
11 x - 6y = -5
12 \end{array}
13 \right.
14 \]
```

出力

$$\left(\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right), \begin{cases} 2x + 3y = 5 \\ x - 6y = -5 \end{cases}$$

ただし、行列や連立方程式は、後述の `amsmath` パッケージを使う方がすっきりと書けます³⁰。

(6.13) 上下に付けるもの 数式モードだけで使えるアクセント記号には、おもに以下のものがあります。

入力	出力	入力	出力
<code>\hat{a}</code>	\hat{a}	<code>\tilde{a}</code>	\tilde{a}
<code>\bar{a}</code>	\bar{a}	<code>\vec{a}</code>	\vec{a}

以下のものは、数式の長さに応じて伸縮します。

入力	出力	入力	出力
<code>\overline{a+b}</code>	$\overline{a+b}$	<code>\underline{a+b}</code>	$\underline{a+b}$
<code>\widehat{a+b}</code>	$\widehat{a+b}$	<code>\widetilde{a+b}</code>	$\widetilde{a+b}$
<code>\overbrace{a+b}</code>	$\overbrace{a+b}$	<code>\underbrace{a+b}</code>	$\underbrace{a+b}$
<code>\overleftarrow{a+b}</code>	$\overleftarrow{a+b}$	<code>\overrightarrow{a+b}</code>	$\overrightarrow{a+b}$

`\overbrace` と `\underbrace` は、上限・下限が真上・真下に付きます。

入力	出力
<code>\overbrace{a+\cdots+z}^{26}</code>	$\overbrace{a+\cdots+z}^{26}$
<code>\underbrace{a+\cdots+z}_{26}</code>	$\underbrace{a+\cdots+z}_{26}$

(6.14) その他の記号 これまでに紹介したもの以外にも、記号を出力する命令が大量に定義されています。おもなものを以下に記します。

矢印

入力	出力	入力	出力
<code>\leftarrow</code>	\leftarrow	<code>\rightarrow</code>	\rightarrow
<code>\uparrow</code>	\uparrow	<code>\downarrow</code>	\downarrow
<code>\updownarrow</code>	\updownarrow	<code>\leftrightarrow</code>	\leftrightarrow
<code>\Leftarrow</code>	\Leftarrow	<code>\Rightarrow</code>	\Rightarrow
<code>\Uparrow</code>	\Uparrow	<code>\Downarrow</code>	\Downarrow
<code>\Updownarrow</code>	\Updownarrow	<code>\Leftrightarrow</code>	\Leftrightarrow
<code>\nearrow</code>	\nearrow	<code>\swarrow</code>	\swarrow
<code>\searrow</code>	\searrow	<code>\nwarrow</code>	\nwarrow

³⁰ 必ず `amsmath` パッケージを使って下さい。

ギリシア文字

入力	出力	入力	出力	入力	出力
<code>\alpha</code>	α	<code>\beta</code>	β	<code>\gamma</code>	γ
<code>\delta</code>	δ	<code>\epsilon</code>	ϵ	<code>\zeta</code>	ζ
<code>\eta</code>	η	<code>\theta</code>	θ	<code>\iota</code>	ι
<code>\kappa</code>	κ	<code>\lambda</code>	λ	<code>\mu</code>	μ
<code>\nu</code>	ν	<code>\xi</code>	ξ	<code>\omicron</code>	\omicron
<code>\pi</code>	π	<code>\rho</code>	ρ	<code>\sigma</code>	σ
<code>\tau</code>	τ	<code>\upsilon</code>	υ	<code>\phi</code>	ϕ
<code>\chi</code>	χ	<code>\psi</code>	ψ	<code>\omega</code>	ω
<code>\varepsilon</code>	ε	<code>\varphi</code>	φ		
<code>\Gamma</code>	Γ	<code>\Delta</code>	Δ	<code>\Theta</code>	Θ
<code>\Lambda</code>	Λ	<code>\Xi</code>	Ξ	<code>\Pi</code>	Π
<code>\Sigma</code>	Σ	<code>\Upsilon</code>	Υ	<code>\Phi</code>	Φ
<code>\Psi</code>	Ψ	<code>\Omega</code>	Ω		

二項演算子

入力	出力	入力	出力	入力	出力	入力	出力
<code>\pm</code>	\pm	<code>\mp</code>	\mp	<code>\times</code>	\times	<code>\div</code>	\div
<code>\circ</code>	\circ	<code>\cap</code>	\cap	<code>\cup</code>	\cup		

関係演算子 ³¹

入力	出力	入力	出力	入力	出力	入力	出力
<code><</code>	$<$	<code>\subset</code>	\subset	<code>\in</code>	\in	<code>\equiv</code>	\equiv
<code>></code>	$>$	<code>\supset</code>	\supset	<code>\ni</code>	\ni	<code>\cong</code>	\cong
<code>\le</code>	\le	<code>\subseteq</code>	\subseteq	<code>\notin</code>	\notin	<code>\sim</code>	\sim
<code>\ge</code>	\ge	<code>\supseteq</code>	\supseteq	<code>\neq</code>	\neq	<code>\perp</code>	\perp

³¹`\le` や `\ge` といった命令の名前は、less than や greater than から来ている。

その他

入力	出力	入力	出力	入力	出力
<code>\infty</code>	∞	<code>\partial</code>	∂	<code>\ell</code>	ℓ

(6.15) 課題 57 – 数式その 2 次のような出力を与える tex ファイル (ファイル名は 0000 和地-57.tex のように) を作り提出して下さい。

出力

$0^\circ \leq \theta < 180^\circ$ とすると $\sin \theta \geq 0$ だから、 $\sin \theta = \sqrt{1 - \cos^2 \theta}$ である。また、 $a, b \in \{3n + 1 \mid n = 1, 2, \dots\}$ のとき、 $a + b \equiv 2 \pmod{3}$ である。

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e.$$

(6.16) 行列、行列式 行列を記述する、より簡単な方法があります。`\left(`、`\right)` と `array` 環境の組合せによる記述と比較すると、はるかに簡潔であることがわかります。

入力

```

1 \documentclass{jarticle}
2 \usepackage{amsmath}
3 \begin{document}
4 \[
5   \begin{pmatrix}
6     1 & 2 & 3 \\
7     4 & 5 & 6
8   \end{pmatrix},
9   \begin{vmatrix}
10    a & b \\
11    c & d
12  \end{vmatrix}
13 \]
14 \end{document}

```

出力

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

`pmatrix` 環境は上のように入りますが、`\begin{document}`の前に `\usepackage{amsmath}` と記述して、`amsmath` パッケージを読み込む必要があります。後述のように `amsmath` パッケージには、他にも便利な数式の命令が多数定義されています。

また、`vmatrix` 環境は行列式のための環境です。

(6.17) 場合分けのある数式 連立方程式や場合分けを記述するための `cases` 環境が、`amsmath` パッケージに定義されています。`\left\{`、`\right.` と `array` 環境の組み合わせによる記述と比較して下さい。

入力

```
1 \documentclass{jarticle}
2 \usepackage{amsmath}
3 \begin{document}
4 \[
5 \begin{cases}
6 2x + 3y = 5 \\
7 x - 6y = -5
8 \end{cases}
9 \]
10 \end{document}
```

出力

$$\begin{cases} 2x + 3y = 5 \\ x - 6y = -5 \end{cases}$$

(6.18) 複数行の別行立て数式 長い式変形や複数行の別行立て数式のための環境が、いくつか `amsmath` パッケージに定義されています。これまでに学

んだものと合わせてまとめると、次のようになります。再び注意ですが、1つの数式にはこれらの1つのみを使います。

- インライン数式: `$ \dots $`
- 別行立て数式: `\[\dots \]`
- 複数行で揃えのある別行立て数式: `\begin{align} \dots \end{align}`
- 複数行で中央揃えの別行立て数式: `\begin{gather} \dots \end{gather}`

つまり、

```
\[ \begin{align} y = -2x - 1 \end{align} \] (間違い)
```

のような使い方はしません (最初の `\[` と最後の `\]` が余分です)。

入力

```
1 \documentclass{jarticle}
2 \usepackage{amsmath}
3 \begin{document}
4 \begin{gather}
5 x^2+x+1 = 1 \\
6 y = 2
7 \end{gather}
8 \begin{align*}
9 f(x) &= x^2+x+1 \\
10 y &= 2
11 \end{align*}
12 \begin{align}
13 (\sqrt{x})' \\
14 &= (x^{\frac{1}{2}})' \\
15 &= \frac{1}{2} x^{-\frac{1}{2}} \\
16 &= \frac{1}{2\sqrt{x}}
17 \end{align}
18 \end{document}
```

出力

$$x^2 + x + 1 = 1 \quad (1)$$

$$y = 2 \quad (2)$$

$$f(x) = x^2 + x + 1$$

$$y = 2$$

$$(\sqrt{x})' = (x^{\frac{1}{2}})' \quad (3)$$

$$= \frac{1}{2} x^{-\frac{1}{2}} \quad (4)$$

$$= \frac{1}{2\sqrt{x}} \quad (5)$$

gather 環境は中央揃え、align 環境は&の位置で揃えられます。こういった数式の揃えは amsmath パッケージを使わなくても可能ではありますが、amsmath パッケージを利用するのが一番良い方法です。gather, align 環境とも数式番号が表示されます。これは*が付加されると抑制されます。

(6.19) 課題 58 – 数式その 3 次のような出力を与える tex ファイル (ファイル名は 0000 和地-58.tex のように) を作り提出して下さい。

出力

$$\begin{aligned} \left| \begin{array}{ccc} 1 & 3 & 2 \\ 3 & 4 & 5 \\ 1 & -1 & 3 \end{array} \right| &= \left| \begin{array}{ccc} 1 & 3 & 2 \\ 0 & -5 & -1 \\ 0 & -4 & 1 \end{array} \right| \\ &= \left| \begin{array}{cc} -5 & -1 \\ -4 & 1 \end{array} \right| \\ &= -5 \cdot 1 - (-1) \cdot (-4) \\ &= -9 \end{aligned}$$

§7 相互参照・目次 (・索引)

(7.1) 相互参照 「第7節を参照のこと」とか、「33 ページにある式 (2) により」のように節などの番号を文中に記述したいとき、じかに「32 ページ」のように書くと、原稿の修正でページがずれたときや、節を新たに挿入したときに手作業ですべて直す必要があります。label, ref, pageref の命令を使うと、TeX が自動的に番号を適切に決定してくれます。

入力

```

1 \documentclass{jarticle}
2 \usepackage{amsmath}
3 \begin{document}
4
5 \section{目次と索引}
6 \begin{align}
7   \label{eq:parab}
8   y = x^2
9 \end{align}
10
11 \section{相互参照}
12 \label{sec:sogo-sansho}
13
14 \subsection{節番号やページ番号}
15 \label{subsec:setsu-page}
16 ここは、第\ref{sec:sogo-sansho}節の
17 \ref{subsec:setsu-page}です。
18 式(\ref{eq:parab})は
19 \pageref{subsec:setsu-page}ページにあります。
20
21 \end{document}

```

出力

1 目次と索引

$$y = x^2 \quad (1)$$

2 相互参照

2.1 節番号やページ番号

ここは、第2節の2.1です。式(1)は1ページにあります。

align 環境の他にも、gather 環境のように式番号の付く環境では、相互参照が可能です。

(7.2) 課題 59 – 相互参照 \TeX の相互参照の機能を用いて、次のような出力を与える tex ファイル (ファイル名は 0000 和地-59.tex のように) を作り提出して下さい。

出力

1 解と係数の関係

$x^2 - x + 1 = 0$ の2つの解を α, β とすると、

$$\alpha\beta = 1, \quad \alpha + \beta = 1 \quad (1)$$

であるから、

$$\alpha^2 + \beta^2 = (\alpha + \beta)^2 - 2\alpha\beta \quad (2)$$

$$= 1 - 2 \cdot 1 \quad (3)$$

$$= -1 \quad (4)$$

である。

2 補足

第1節の(2)式から(3)式への変形では(1)式を用いた。

2つ目の別行立て数式では、align 環境を用います。また1つ目の別行立て数式の、2つの式の間空白は、\quad 命令で空けます。これは以前用いた、命令よりも広い空きを作ります。

(7.3) 目次 \tableofcontents 命令を書くと、そこに目次が作成されます。

入力

```

1 \documentclass{jarticle}
2 \begin{document}
3 \tableofcontents
4 \section{目次と索引}
5 最初の節
6 \section{相互参照}
7 次の節
8 \subsection{節番号やページ番号}

```

```
9 | いろいろ書く。
10 | \end{document}
```

出力

目次

1	目次と索引	1
2	相互参照	1
2.1	節番号やページ番号	1

1 目次と索引

最初の節

2 相互参照

次の節

2.1 節番号やページ番号

いろいろ書く。

§8 画像

(8.1) 画像の挿入 Windows の dviout で BMP 画像ファイル表示させることを中心に説明します。他の OS、出力方法 (pdf に変換するなど)、画像形式 (JPEG, PNG, EPS, PDF など) でも基本は同じですが、別途プログラムをインストールしなくてはならないこともあります³²。

³² BMP 形式ではない画像を BMP 形式に変換したいならば、例えば、画像を Windows の「ペイント」で開き、ファイルメニューの「名前を付けて保存」で BMP 形式を選んで保存する

T_EX の文書に画像を挿入するには、graphicx パッケージを利用します。dviout で出力するためには、dviout オプションが必要です。挿入したい所で、\includegraphics 命令を用いますが、BMP などの画像形式では、bb オプション (バウンディングボックス。画像のサイズ) を指定する必要があります。

includegraphics 命令の文法

```
\includegraphics{画像ファイル名}
\includegraphics[オプション]{画像ファイル名}
```

includegraphics 命令のオプションの例

オプション	意味
scale=1.2	画像を 1.2 倍に拡大して出力する
bb=0 0 100 200	サイズが横 100 縦 200 ピクセルである指定

入力

```
1 | \documentclass{jarticle}
2 | \usepackage[dviout]{graphicx}
3 | \begin{document}
4 | 0.3倍の大きさ。
5 | \includegraphics[bb=0 0 240 320,scale=0.3]{t.bmp}
6 | \end{document}
```

出力



0.3 倍の大きさ。

と、新しく BMP ファイルが出来ます。

§9 マクロ

今学期はやりません。

索引

Clock.Hour, 9
Else, 8
ElseIf, 9
EndSub, 16
false, 14
For, 10
Goto, 14
IF (Excel), 2
If, 7
Math.Abs, 17
Math.ArcCos, 17
Math.ArcSin, 17
Math.ArcTan, 17
Math.Ceiling, 17
Math.Cos, 17
Math.Floor, 17
Math.GetDegrees, 17
Math.GetRadians, 17
Math.GetRandomNumber, 7, 17
Math.Log, 17
Math.Max, 10, 17
Math.Min, 10, 17
Math.NaturalLog, 17
Math.Pi, 17
Math.Power, 17
Math.Remainder, 17
Math.Round, 17

Math.Sin, 17
Math.SquareRoot, 11, 17
Math.Tan, 17
Program.Delay, 19
Step, 11
Sub, 16
Text.Append, 18
Text.ConvertToLowerCase, 18
Text.ConvertToUpperCase, 18
Text.EndsWith, 18
Text.GetCharacter, 18
Text.GetCharacterCode, 18
Text.GetIndexOf, 18
Text.GetLength, 18
Text.GetSubText, 18
Text.GetSubTextToEnd, 18
Text.IsSubText, 18
Text.StartsWith, 18
TextWindow.ForegroundColor, 4
TextWindow.Read, 6
TextWindow.ReadNumber, 6
TextWindow.Write, 5
TextWindow.WriteLine, 3
true, 14
While, 12
<, 33
>, 33

`\,`, 30
`\\`, 25
`\alpha`, 33
`\arg`, 31
`\bar`, 32
`\beta`, 33
`\bigcap`, 30
`\bigcup`, 30
`\bmod`, 31
`\cap`, 33
`\cdot`, 30
`\cdots`, 30
`\chi`, 33
`\circ`, 33
`\cline`, 28
`\cong`, 33
`\cos`, 31
`\cup`, 33
`\ddots`, 30
`\deg`, 31
`\Delta`, 33
`\delta`, 33
`\det`, 31
`\dim`, 31
`\div`, 33
`\documentclass`, 22
`\Downarrow`, 32
`\downarrow`, 32
`\ell`, 33
`\epsilon`, 33
`\equiv`, 31, 33
`\eta`, 33
`\exp`, 31
`\footnotesize`, 25
`\frac`, 30
`\Gamma`, 33
`\gamma`, 33
`\ge`, 33
`\hat`, 32
`\hline`, 28
`\Huge`, 25
`\huge`, 25
`\in`, 33
`\includegraphics`, 37
`\infty`, 31, 33
`\int`, 30
`\iota`, 33
`\item`, 27
`\kappa`, 33
`\label`, 35
`\Lambda`, 33
`\lambda`, 33
`\LARGE`, 25
`\Large`, 25
`\large`, 25
`\ldots`, 30
`\le`, 33
`\left`, 31
`\Leftarrow`, 32
`\leftarrow`, 32

`\Leftrightarrow`, 32
`\leftrightharpoonrightarrow`, 32
`\lim`, 31
`\log`, 31
`\max`, 31
`\min`, 31
`\mp`, 33
`\mu`, 33
`\multicolumn`, 28
`\nearrow`, 32
`\neq`, 33
`\newpage`, 25
`\ni`, 33
`\normalsize`, 25
`\notin`, 33
`\nu`, 33
`\nrightarrow`, 32
`\Omega`, 33
`\omega`, 33
`\overbrace`, 32
`\overleftarrow`, 32
`\overline`, 32
`\overrightarrow`, 32
`\pageref`, 35
`\paragraph`, 24
`\partial`, 33
`\perp`, 33
`\Phi`, 33
`\phi`, 33
`\Pi`, 33
`\pi`, 33
`\pm`, 33
`\pmod`, 31
`\prod`, 30
`\Psi`, 33
`\psi`, 33
`\quad`, 36
`\ref`, 35
`\rho`, 33
`\right`, 31
`\rightarrow`, 32
`\rightarrow`, 32
`\scriptsize`, 25
`\searrow`, 32
`\section`, 24
`\Sigma`, 33
`\sigma`, 33
`\sim`, 33
`\sin`, 31
`\small`, 25
`\sqrt`, 30
`\subsection`, 24
`\subset`, 33
`\subseteq`, 33
`\subsubsection`, 24
`\sum`, 30
`\supset`, 33
`\supseteq`, 33
`\swarrow`, 32
`\tableofcontents`, 36

`\tan`, 31
`\tau`, 33
`\TeX`, 23
`\textbf`, 25
`\textgt`, 25
`\textit`, 25
`\textrm`, 25
`\textsc`, 25
`\textsf`, 25
`\textsl`, 25
`\texttt`, 25
`\Theta`, 33
`\theta`, 33
`\times`, 33
`\tiny`, 25
`\to`, 31
`\underbrace`, 32
`\underline`, 25, 32
`\Uparrow`, 32
`\uparrow`, 32
`\Updownarrow`, 32
`\updownarrow`, 32
`\Upsilon`, 33
`\upsilon`, 33
`\usepackage`, 34
`\varepsilon`, 33
`\varphi`, 33
`\vdots`, 30
`\verb`, 23
`\widehat`, 32
`\widetilde`, 32
`\Xi`, 33
`\xi`, 33
`\zeta`, 33
`^`, 29
`_`, 29
`o`, 33
10pt, 22
11pt, 22
12pt, 22
`align*`環境, 35
`align` 環境, 34
`amsmath` パッケージ, 34
`array` 環境, 31
BMP 形式, 37
`cases` 環境, 34
`center` 環境, 26
`description` 環境, 27
`document` 環境, 22
`dvi` ファイル, 21
`enumerate` 環境, 27
EPS 形式, 37
`flushleft` 環境, 26
`flushright` 環境, 26
`gather*`環境, 35
`gather` 環境, 34

graphicx パッケージ, 37

itemize 環境, 27

JPEG 形式, 37

landscape, 22

PDF 形式, 37

pmatrix 環境, 33

PNG 形式, 37

quotation 環境, 26

quote 環境, 26

tabular 環境, 28

TeXworks, 21

tex ファイル, 21

twocolumn, 22

vmatrix 環境, 33

インテリセンス, 3

インライン数式, 29

演算子, 4

改行, 24

改ページ, 24

カウンタ変数, 10

下線, 24

空行, 24

環境, 26

空白, 22

グループ化, 29

構造化プログラミング, 14

サブルーチン, 15

順次, 4

数式モード, 29

スコープ, 15

添字, 29

テキストモード, 29

特殊文字, 23

流れ図, 13

反復, 12

フローチャート, 13

プログラムのフォーマット, 8

分岐, 8

別行立て数式, 29

変数, 5

無限ループ, 13

ユークリッドの互除法, 15

ラベル, 14

累乗, 29