計算機科学'13 ＃1 — 文書作成とWYSIWYG/マークアップ

久野　靖*

2013.5.21

「計算機科学」は「計算機科学基礎」の続編として、「計算機科学基礎」では時間的に扱えなかった、個別の具体的な応用よりの話題を毎回取り上げます。ただしここでも、個別のアプリケーションではなく「原理・仕組み」に焦点を当てるようにしています。授業の進め方は基本的に「計算機科学基礎」と同様で、予定は次のようになっています。

- ＃1 — 文書作成と WYSIWYG/マークアップ (LaTeX、HTML+CSS を含む)
- ＃2 — マルチメディアと Web 上の描画
- ＃3 — ユーザインタフェース (ウィンドウシステム、GUI)
- ＃4 — スクリプト、Web アプリケーション (1)
- ＃5 — データベース、Web アプリケーション (2)

1　文書作成

1.1　テキストと文書の違いと位置付け

コンピュータはその最初期においては、数値を大量に「計算」する装置として使われました。しかし、人間の知的活動のうち、数値によって表される部分は実はごく一部分でり、大半の知的活動のアウトプットは**文書** (書籍、レポート、手紙、…) の形で表現されます。ですから、コンピュータの適用範囲が広がるにつれ、文書もコンピュータで扱われるようになりました。現在でも、整った文書を作成するためのソフトである**ワープロソフト** (や、後で説明する**文書整形系**) は、コンピュータにおいて最も多く使われているアプリケーションソフトの 1 つだといえるでしょう。

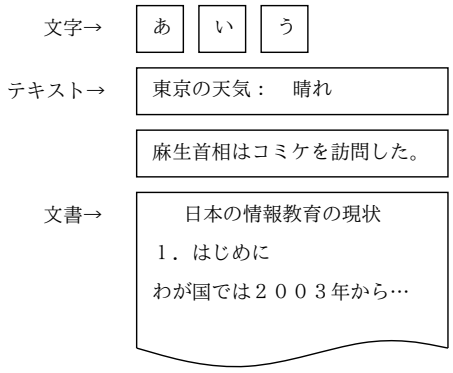

図 1: 文字/テキスト/文書

ここで、**テキスト**と文書の違いについて触れておきます (図 1)。テキスト情報とは「文字で表された情報」つまり自然言語 (日本語、英語など) で表現された情報を意味します。我々は多くの情報を言葉で表し扱いますから (思考、伝達、記録など)、そのためテキスト情報が大きな位置を占めるのは当

*経営システム科学専攻

然です。文書もテキストが土台である点は同じですが、まとまった内容を扱い分量があるので、見やすい整形や章立てなどが必要なもの、と考えるのがよさそうです。本、書類、この資料など、ほとんどのまとまったテキスト情報は文書です。[1]

では、コンピュータ上でテキスト情報を扱うソフトウェアとしてはどのようなものがあるでしょうか (図 2)。真っ先に「ワープロソフト」が思い浮かぶかも知れませんが、ワープロソフトは後で出てくるように、文書を扱うためのものです。単純なテキスト情報を対話的に入力/編集するためのソフトウェアは**テキストエディタ** (text editor) と呼び、Emacs、Windows の「メモ帳」、Mac OS X の「シンプルテキスト」などがその代表例です。

そのほか、計算機科学基礎でも Unix のフィルタを扱いましたが、これらもその多くはテキストを加工するのに使えます (ただし、日本語がうまく扱えないものがあることもあります)。さらに、後の回で扱いますが、**スクリプト** (script) と呼ばれる簡単なプログラムを書くことで、ある程度自動的にテキストの加工を行うことも可能になります。

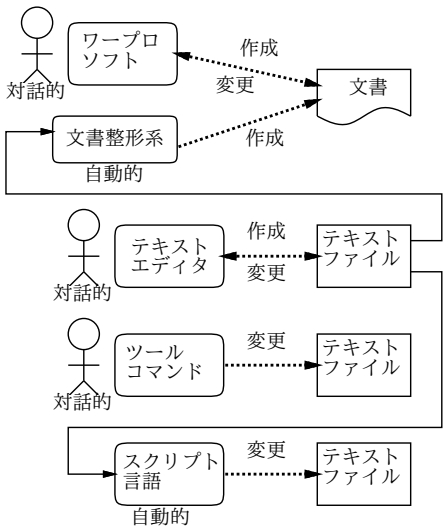

図 2: テキストを扱うソフトウェア

1.2 文書整形系

前述のように、実際に我々がまとまった情報を入手する場合はテキストというより文書を読むわけなので、それが美しく読みやすいことは大切です。そこで、「美しい文書とはどういうものか」「それをうまく作るにはどのようにシステムを設計したらいいのか」を考えてみましょう。あなたは次の問いに答えられますか?

- そもそも、美しい文書とはどういうものを言うのか?
- なぜ美しい文書が望ましいのか?

1 番目の問いに対して「文字がきれいな字で打たれている」とか「多様な字形 (フォント) が使われてる」などの回答が浮かんだ人もいるかと思います。しかし、文字がきれいでもそれがでたらめな規則で並べられていたら (たとえば 1 文字ごとにフォントが違うなど)、1 文字 1 文字を見て「ほう、この『あ』という文字は美しいですねえ」と鑑賞 (?) はできても、その文書を読んで理解する上ではちっとも嬉しくないはずです。

[1] 電話帳や辞書はまとまったテキスト情報ですがまとまった 1 つの内容というわけではないので、文書とは考えないのが普通でしょう。

そこで 2 番目の問いが問題になります。なぜ美しい文書が望ましいのでしょう? それは、文書が美しいとその内容を見て取る効率がよいからだと考えてはどうでしょうか。具体的には、文字の大きさが読みやすいなどに加えて、「ここは見出し」「これは図」など、文章の構造が見てとりやすく、必要なことがらが (たとえば見出しをざっとスキャンしていって) すぐ探せる、などの点も重要になります。これを整理すると次のようにまとめられます。

- 文書の構造が読み手に的確に伝わること。

逆の面から見ると、「美しい文書」を扱うなら、そこには内容であるテキスト (文字) に加え、その文書を美しくあらせるための**付加情報**も情報として含まれているはずです。なぜなら、付加情報があってはじめて、どの部分はどういう風にする (たとえば見出しだから目立つ字体にする)、という処理が可能だからです。同じ大きさの文字だけから成るプレーンテキストファイルと、ワープロなどで美しく仕上げた文書との違いは、この付加情報の部分にあるわけです。

1.3 文書整形のアルゴリズム

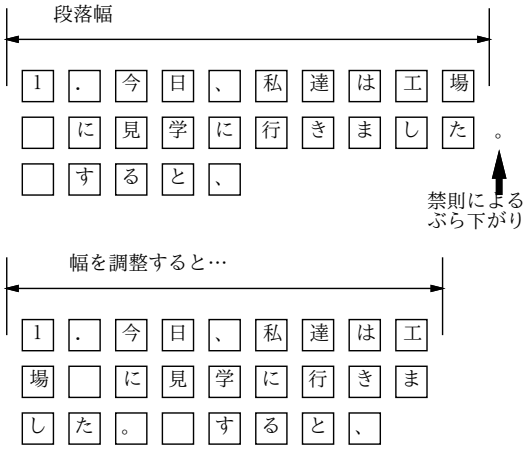

図 3: 「原稿用紙方式」の困った点

ワープロソフト (や文書整形系) が「どのように」各文字を紙面に配置しているかを言葉で説明できますか? たとえば、初期のワープロソフトなどでは「原稿用紙」モデル、つまり画面や紙の上に縦横のサイズが決まったます目があり、そこに 1 文字ずつ文字を詰めて行く、というアルゴリズムが使われていました。それだとどんな美しくないことが起きるか分かりますか?

- 禁則処理 (行頭に「。」などが来ては行けない等の処理) を行った結果、右端が「でこぼこ」になる。
- 英字を日本語と同じ文字間隔で詰めるとえらく間延びになる。かといって日本語 1 文字ぶんに英字 2 文字を詰めると窮屈 (しかも英字が奇数だと半分のアキが…)。

さらに箇条書きなど字下げがある段落を書いてから文字詰め幅を変更すると、空白がずれて悲惨なことになります (図 3)。

今日のソフトではこのようなことはなく、段落は「1 行目の左マージン」「2 行目以降の左マージン」「右マージン」の位置で形が指定されており、文字を普通に打って行くことでこれらのマージン内で自動的に文字が詰め合わされて行くようになっています (図 4)。

ただし、これをちゃんと使うためには、段落の形を変える時に「ルーラ」を出してマージンを設定しないといけません。もしかして、あなたは今でも字下げをするのに空白を挿入していますか? (とすると、今でも「悲惨な目」に逢っていることになりますね…)

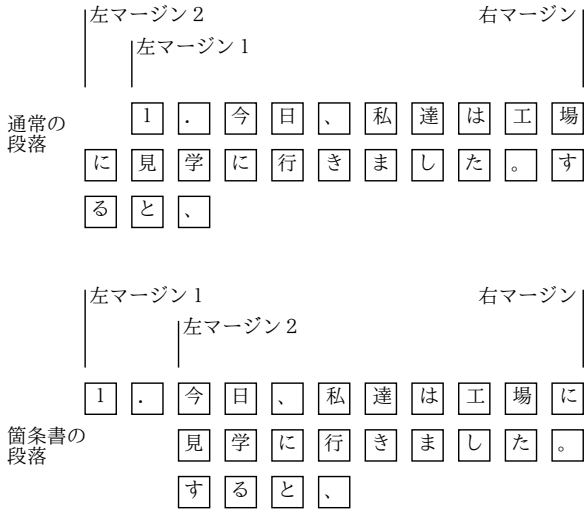

図 4: マージン指定による段落の詰め合わせ

さらに禁則なども考慮して整形するため、現在は次のようなアルゴリズムが使われます (図 5)。

- 段落ごとに「詰め合わせる幅」があり、それぞれの文字は「四角い箱」と考えてその箱を幅一杯まで詰めていく。
- 文字と文字の組み合わせごとに適切な「あき」が変わってくるので、組み合わせごとに適切な「にかわ (のり)」を詰めていく。
- 禁則処理などでその位置で行かえできないなら、少し余分に詰めるか詰めたものを取り除く。
- 右端が綺麗に揃うように全体を詰めたり伸ばしたりする。伸ばすとにかわの部分が伸びて全体が長くなる。
- このとき、すべてのにかわが均等に伸びるわけではなく、「(」の左、「)」の右、「。」の右など、伸びてもおかしくないところが余分に伸びるようにする。
- このようにして各行が出来てくると、今度はページにその行を上から詰めていく。ここでも「にかわ」がはさまれ、ページの切れ目の禁則 (節タイトルがページ下端に来ないとか) に応じて同様の処理が行われる。

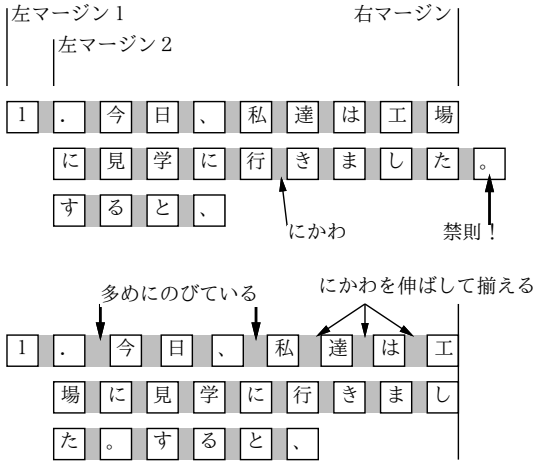

図 5: 整形のアルゴリズム

ここで重要なのは、「文字どうしのアキ (にかわの幅や伸び)」「段落間のアキ」「マージンの値」などをどう調整すれば見やすい文書になるか、ということです。これについては、長い歴史のある印刷・出版界が多くのノウハウを持っており、それを駆使して読みやすい書籍を作っています。そこで、LaTeX などの整形システムでも、これらのノウハウを借りて来てパラメタを調整することで、できるだけ見やすい整形となるように努めているわけです。

ここで後述する「意味づけ方式」が重要になってきます。ユーザがいちいちパラメタを指定して調整するのは手間が大変な上、そのユーザが十分なノウハウを持っていない限り、美しいレイアウトは困難です。しかし、ソフト側では「ここは箇条書き」「ここはタイトル」などの情報をユーザに教えてもらわなければ、どの部分が何を意味するかは分からないため、せっかくのノウハウが活かせません。意味づけ方式では「どの部分は何か」が記述されるので、ソフト側でその情報をもとに適切なパラメタを適用し、見やすく整形できます。

演習 6-1 Word、OpenOffice Write などのワープロソフトや (この後使い方を説明する) LaTeX などによる文書において、通常の段落や箇条書きの段落を作成し、整形アルゴリズムのふるまいを観察しなさい。その後、次のテーマから 1 つ以上 (できれば全部) 選んで追求してみなさい。

a. 通常の段落の「1 字下げ」が実際にはどれくらいの下げになっているか、またその下げ方が調整できるなら調整してどれくらいが見やすいのかについて、検討しなさい。
b. 箇条書きなどの通常でない形の段落について、その標準の形や、調整したときの見え方の違いを検討しなさい。[2]
c. 段落の「両端揃え (justify)」を ON にした場合のふるまいを、整形幅を変えたり文字を加減して観察しなさい。とくに、「。」などが行末に来る時や、英単語が行をまたがりそうな位置にある場合については、必ず検討すること。

「◎」の条件: (1) 小問から 2 つ以上が回答されており、(2) 適切な (と担当が考える) 観察・検討・考察が記されていること。

1.4 見たまま方式とマークアップ方式

美しい文書を作成するためには、テキストと付加情報をともに入力したり修正するなどの必要があることまでは分かりました。ではそれを具体的にどのようにして行なったらよいでしょうか? コンピュータの世界では、その方法として次の 2 種類が考えられ、現に使用されています (図 6)。

- 見たまま方式、**WYSIWYG**(What You See Is What You Get — 「あなたが見るものがあなたの得るものである」) 方式。この方式では、実際にプリンタで印刷したイメージにできるだけ近い画像を生成し、画面で表示します。そこでマウスやメニューなどを用いて、画面上に見える文書を対象に修正を施します。配置やフォントを変更したり、文章を直したりすると、その結果は直ちに画面に反映されます。Word などもこの仲間です。
- **マークアップ方式** — エディタを用いて作成するファイル中に、テキストに混ぜて特別な「印」(マークアップ) を入れ、これに従って**整形系** (formatter) がレイアウトを行います。この方式では、マークアップの指定次第でとても細かい制御が行なえますが、マークアップを覚える必要があります。また、レイアウト結果の確認にはその都度整形・プレビュー (画面提示) を指示する必要があります。マークアップ方式は、さらに次の 2 種類に分かれます。
  - **コマンド方式** — マークアップは「ここからこのフォント」「ここで改ページ」などのコマンドであり、その位置で命令を実行することで整形を行う。**troff**、**TeX** など古くからのソフトがこれの方式です。

[2] ソフトによっては簡単に調整できない場合もあります。

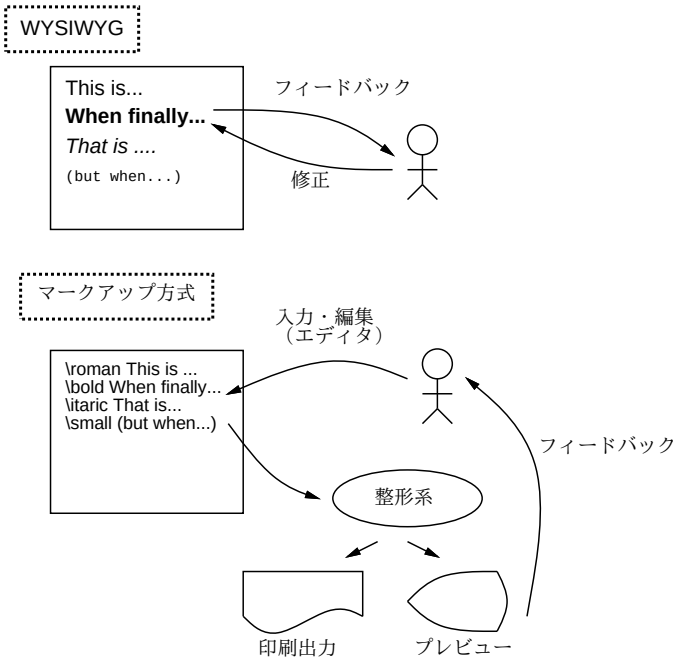

図 6: WYSIWYG 方式とマークアップ方式

- **意味づけ方式** (semantic encoding) — 同じマークアップでも、「どう整形する」ではなく「ここは章の切れ目」「ここからここまでは箇条書き」など、文書の「意味」を表す印を入れるもので、より覚えやすく、1 つのマークアップから 2 段組、大きな文字など多様なレイアウトを生成できます。HTML や以下で扱う LaTeX がこれに属します。

Word の隆盛を見れば分かるように、世の中のワープロソフトでは見たまま方式が主流なのに、この章ではマークアップ方式を取り上げる、その理由について説明しておきましょう。

- 見たまま方式は、個別の個所ごとに「このスタイル」という指定を繰り返し行い、そのマウスやメニュー操作はかなり手間が掛かります。マークアップなら、ここはこういうマークアップということを覚えればあとは「打ち込む」作業の一部であり手間はそれほどありません。
- 見たまま方式は、(大抵は)「こういう体裁に」ということを人間がデザインし制御するので、センスがないと美しくできません。意味づけ方式では、スタイルは標準化されているので、そのようなことに頭を悩ます必要はありません。[3]
- 見たまま方式は、1 つの体裁に合わせて文書を作るため、後で体裁を変更するのは大変です。意味づけ方式なら、整形用スタイルを複数用意することで、多様な体裁に対応できます。
- 見たまま方式は、ファイル形式が特定ソフトに依存しがちで、コンテンツの流用が難しくなります。[4]マークアップはもともと全部テキストファイルなので、(マークアップの統一的な書き換えなどの処理を行うなどでして) 流用が容易です。

「文書を作るのにいちいちコマンドを調べるなどとても耐えられない」「そんな面倒な方式は旧態依然だ」と思ったでしょうか? しかし実は、大量に文書を作成するプロほど、LaTeX のような整形系をメインに使っています。それはコマンド対メニューのように、ワープロソフトだと一見分かりや

[3]WYSIWYG でもスタイルシート機能を活用すればいいはずなのですが、それを使っている人を見たことがありません。それはたぶん、WYSIWYG ではどうしても「ここはこのフォントでこの大きさ」と指示したくなるからではないかと思います。

[4]テキストファイルに変換すれば他のソフトに持っていけますが、付加情報はすべて失われてます。

すくても操作に時間が掛かるため、大量に文書を作る人にとってはコマンドの方が楽で効率的だからです。また、LaTeX はスタイルのチューニングが徹底しているので、「おまかせ」で比較的美しい文書ができます。

1.5 LaTeX を動かしてみる

というわけで、今回は LaTeX を「とにかく」体験してみて頂きます。コマンドの書き方は少し後回しにして、先にサンプルファイルを「そのまま」整形してみましょう。LaTeX の入力ファイルは「.tex」で終わる名前をつけます。以下ではそれが「sample.tex」であるものとして説明します。

整形のためのコマンドは platex ですので、入力ファイルを指定してこれを起動します。

```
% platex sample.tex | nkf      ← nkf はメッセージ日本語表示に必要
This is pTeX, Version 3.141592-p3.1.9 (euc) (Web2C 7.5.4)
(./sample.tex
pLaTeX2e <2006/01/04>+0 (based on LaTeX2e <2003/12/01> patch level 0)
(/usr/local/ptex-3.1.9/share/texmf/ptex/platex/base/jarticle.cls
Document Class: jarticle 2002/04/09 v1.4 Standard pLaTeX class
(/usr/local/ptex-3.1.9/share/texmf/ptex/platex/base/jsize10.clo))
(./sample.aux)
 (/usr/local/ptex-3.1.9/share/texmf-dist/tex/latex/base/omscmr.fd) [1]
[2] [3] (./sample.aux) )
Output written on sample.dvi (3 pages, 7456 bytes).
Transcript written on sample.log.
%
```

後で自分でコマンドを打つときは、もしエラーがあると途中で「?」というプロンプトが出て処理が止まります。その時は「x[RET]」と打ち込んで実行を終わらせ、エラーメッセージを見て間違いを直し、再度実行してください。整形が成功すると.dvi で終わる名前のファイル (この場合だと sample.dvi) ができています。

整形結果はまず画面で確認します。それにはプレビューア xdvi を「xdvi sample.dvi[RET]」のようにして起動します。すると、図 7 のような窓が現れます。右側に縮尺やページめくりを指定するボタンが並んでいるので、これらのボタンをマウスで操作して各ページが意図どおりかどうか確認します。プリンタに出すには、DVI 形式から PostScript(PS) 形式に変換するためにコマンド dvips コマンドを使います。PS 形式ファイルはそのまま PS プリンタに送れば印刷されます。そのほか、PS ビュア (gv など) で見たりもできます。

```
% dvips -o sample.ps  sample.dvi    ← PS に変換 (「.dvi」は省略可)
This is dvips(k) p1.7 Copyright 2005 ASCII Corp.(www-ptex@ascii.co.jp)
based on dvips(k) 5.95a Copyright 2005 Radical Eye Software
(www.radicaleye.com) ' TeX output 2006.04.13:1544' -> sample.ps
<tex.pro><texps.pro>. <cmr10.pfb>[1]
% gv sample.ps &              ← PS ビュアで見る
% lpr -Pepsa sample.ps        ←プリンタ出力
```

また、PS 形式はコマンド ps2pdf を使って PDF 形式に変換できます。配布用などには PDF が一般的ですね。

```
% ps2pdf sample.ps sample.pdf     ← PDF に変換 (「.dvi」は省略可)
% acroread sample.pdf             ← Acrobat Reader で見る
```

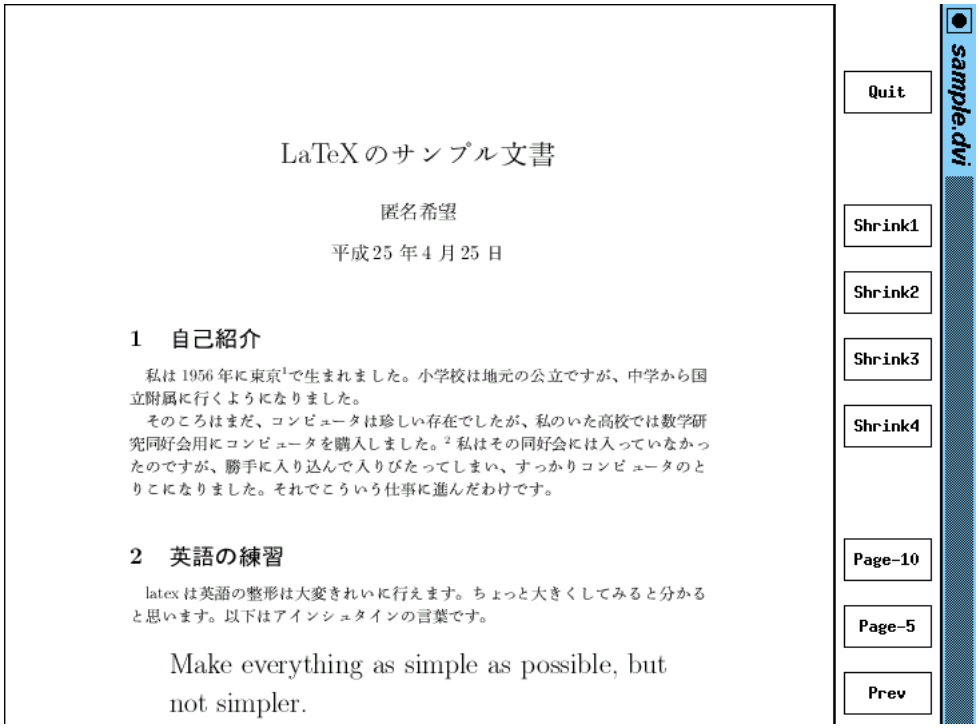

図 7: XDVI によるプレビュー

大変だったでしょうか? 確かに最初はとっつきにくいですが…では、先頭の行を次のように直すとどうなるか、試してみてください。

```
\documentclass[12pt,a4j]{jarticle}        ←フォントサイズ変更
\documentclass[twocolumn,a4j]{jarticle}   ←2段組みに変更
```

このような取り換えが簡単なのが意味づけ方式の利点だということが理解して頂けるかと思います。

演習 6-2 `sample.tex` を入手して整形し、結果を画面で確認してみなさい。うまく行ったら、冒頭の documentclass 指定について次の変更を 1 つ以上 (できれば全部) 行ってみなさい。

a. twocolumn 指定の有無による違いを比べてみる。
b. フォントサイズ指定を変えた時の違いを比べてみる (フォントサイズは `9pt`、`10pt`、`11pt`、`12pt` が指定できる)。
c. 紙サイズ指定を変えた時の違いを比べてみる (紙サイズは `a5j`、`b5j`、`a4j`、`b4j` が指定できる)。なお、`xdvi` は紙サイズ変更に対応しないので、次のように `dvips` を使って紙サイズを指定して PS ファイルを作りプレビューしてください。

```
% dvips -o sample.ps -ta6 sample.dvi  ←紙サイズ A6
% gv sample.ps &
```

いずれも、指定を変更した時にきれいに見えるようにどのような工夫が見られるか、文字サイズに比べて整形幅が狭くなった場合に整形時にどのようなメッセージが出てどのような整形結果になるかをきちんと見ること。

「◎」の条件: (1) 小問のうち 2 つ以上をやってあり、(2) 上記「いずれも」が守られていて、(3) 適切な (と担当の判断する) 説明・考察が書かれていること。

ここに示すように、LaTeX の文書はまずこの文書がどんなスタイルの文書であるかを宣言する部分から始まる。スタイルの例としては「記事 (jarticle)」、「本 (jbook)」、「報告 (jreport)」などがある。ここでは一番簡便な jarticle を例に使用している。これに加えて字の大きさ、図形の取り込み機能などをオプションとして指定できる。

続いて「文書開始」の宣言があり、この中に文書の本体が入る。この部分には様々な記述が可能だが、一番簡単にはここに示すように段落ごとに 1 行あけて次々に文章を書いて行くだけでも地の文が普通にできる。つまり、「特に指定がない」ならば「地の文」である。

あとは文書の本体が終わったら最後に必ず「文書終了」の宣言がある。最低限必要なのはたったこれだけである。

図 8: pLaTeX の出力

2 LaTeX 入門

2.1 文書の基本構造

この節では、意味付け方式の文書整形システムの一つである **LaTeX** (コマンド方式の TeX を拡張する形で構成されています) について、実例中心で一通り解説することを通じて、意味付け方式 (と広義の指令方式) というのはどんなものかの感触を持っていただこうと思います。まずともかく、LaTeX の文書に最低限必要な基本構造の例を見ていただきましょう。次のようなものは、一つの完結した LaTeX 文書です (内容も読んでみてください)。

```
\documentclass{jarticle}
\begin{document}
ここに示すように、LaTeX の文書はまずこの文書がどんなスタイルの文書である
かを宣言する部分から始まる。スタイルの例としては「記事 (jarticle)」、
「本 (jbook)」、「報告 (jreport)」などがある。ここでは一番簡便な
jarticle を例に使用している。これに加えて字の大きさ、図形の取り込み
機能などをオプションとして指定できる。

続いて「文書開始」の宣言があり、この中に文書の本体が入る。この
部分には様々な記述が可能だが、一番簡単にはここに示すように段落ごとに 1 行
あけて次々に文章を書いて行くだけでも地の文が普通にできる。つまり、
「特に指定がない」ならば「地の文」である。

あとは文書の本体が終わったら最後に必ず「文書終了」の宣言がある。
最低限必要なのはこれだけである。
\end{document}
```

これを LaTeX に掛けて打ち出すと図 8 のようになります。ところで、ここで少し補足しておくと、まず宣言 (指令) は

```
\指令名 [オプション指定]{パラメタ}
```

のような形をしています。ここでオプション指定がないときは [...] はなく、またパラメタがないときはさらに{...}も不要です。ということは\はそのままでは文書に含められないわけです。LaTeX では、このような特別な (そのままでは使えない) 記号としては次のものがあります。[5]

```
# $ % & ~ _ ^ \ { }
```

これらはとりあえず「そのままでは使わない」ようにしてください。

[5] この他に<、>など記号自体に特別な意味はないのですが、TeX の標準フォントの関係で出力すると別の字になってしまうものがいくつかあります。

2.2 表題、章、節

いきなり本文が始まって延々と続くだけではあんまりですから、表題と章立てをつけましょう。その場合の例を次に示します。

```
\documentclass{jarticle}
\begin{document}
\title{あなたがつけた表題}
\author{書いた人の名前}
\maketitle

\section{表題について}
表題をつけるには title、author など必要な事項を複数指定した後、最後
に maketitle というとそれらの情報をもとに表題が生成される。その際指
定されなかった事項は出力されないかまたは適当なものが自動生成される
(たとえば日付を指定しないと整形した日付が入る。)

\section{章立てについて}
ここにあるように section 指令を使って各節の始まり、およびその表題を
指定する。節の中でさらに分ける場合には section というのも使え、
さらに subsection というのまで可能である。ちなみに jbook/jreport スタイル
では section の上位に chapter があるが jarticle では section からである。
ところで、節番号 etc. は自動的に番号付けされるのに注意。

\section{より細かいことは}
より細かいことは、参考書を参照してください。
\end{document}
```

これを整形すると、文書の冒頭にタイトル、著者、日付 (整形した日付) がそれなりの形式で用意され、見出しも前後にアキを取ってそれなりのフォントで出力されます。なお、section より小さいレベルの見出しとして subsection、subsubsection も使えます。

2.3 いくつかの便利な環境

表題と章建てができればこれだけで結構普通の文書は書けてしまうはずですが、しかし全部地の文ではめりはりがつきません。それに、プログラム例など行単位でできているものまできれいに詰め合わされてしまうのでは困りますね。このように、部分的にスタイルが違う部分を指定するのには次のような形 (**環境**と呼びます) を使います。

```
\begin{環境名}
....
....
\end{環境名}
```

では、よく使う環境について、説明していきましょう。まず、**verbatim 環境** (そのまま) というのは文字通り入力をそのまま整形せずに埋め込みます。たとえば

```
\begin{verbatim}
This is a pen.
 That is a dog.
\end{verbatim}
```

は、つぎのような出力になります。プログラム例などを示すのに最適です。

```
  This is a pen.
    That is a dog.
```

ちなみに、独立した行にする代わりに、文章のなかに一部「そのまま」を埋め込みたい場合もあります。そのような時には verbatim の類似品で`\verb|....|`という書き方を使うことができます。これで縦棒にはさまれた部分がそのまま出力できます。中に縦棒を含めたい時は、両端に縦棒以外の適当な記号を使います。なお、`verbatim` も `verb` も脚注 (後述) の中には入れられません。

つぎに、**itemize 環境** (箇条書き) について説明しましょう。この場合は、環境のなかに複数「`\item ....`」というものが並んだ格好になっていて、その一つずつが箇条書きの 1 項目になります。たとえば

```
\begin{itemize}
\item あるふぁはギリシャ文字の一番目です。
\item ベータはギリシャ文字の二番目です。
\item ガンマはギリシャ文字の三番目です。
\end{itemize}
```

は、つぎのような出力になります。

- あるふぁはギリシャ文字の一番目です。
- ベータはギリシャ文字の二番目です。
- ガンマはギリシャ文字の三番目です。

次に、この itemize を **enumerate 環境** (数え上げ) に変更すると、出力の際に項目ごとに 1, 2, 3... と自動的に番号付けされるようになります。入力はほとんどまったく同じだから出力のみ示します。

1. あるふぁはギリシャ文字の一番目です。
2. ベータはギリシャ文字の二番目です。
3. ガンマはギリシャ文字の三番目です。

点や番号でなくタイトルをつけたい場合には **description 環境** (記述) を使います。これは

```
\begin{description}
\item[あるふぁ] これはギリシャ文字の一番目です。
\item[ベータ] これはギリシャ文字の二番目です。
\item[ガンマ] これはギリシャ文字の三番目です。
\end{description}
```

のように、各タイトルを `[]` の中に指定するもので、上の整形結果は次のようになります。

あるふぁ これはギリシャ文字の一番目です。

ベータ これはギリシャ文字の二番目です。

ガンマ これはギリシャ文字の三番目です。

この他にもいくつか環境がありますが、とりあえずこれくらいで十分使えると思います。

2.4 脚注

脚注の作り方はとても簡単で、単に好きなところに`\footnote{.....}`という形で注記をはさんでおけばそれがページの下に集められて脚注になり、そこへの参照番号は自動的につけられます。[6]

2.5 文字サイズ

文字サイズ/字体を指定したい場合は、`{\LARGE 大きく}`のように書くと{\LARGE 大きく}なります。このように LaTeX では`{...}`が範囲を区切り、字体や文字サイズを変更した効果はその範囲を出ると終わります。字体としては **bf**(ボールド)、rm(立体)、*it*(斜体)、`tt`(タイプライタ体)、文字サイズとしては{\huge huge}、{\LARGE LARGE}、{\Large Large}、large、normalsize、small、footnotesize、scriptsize、{\tiny tiny} が指定できます。

2.6 表

表は情報を整理して提示する強力なツールです。LaTeX では表は **tabular 環境**で作り出します。その先頭では、

- 表のカラム数
- それぞれのカラムを左/中央/右揃えのどれにするか
- 各カラムの境界および左右に罫線を引くかどうか

を 1 つの文字列で指定します。すなわち、1 つのカラムごとに揃え方を `l`/`c`/`r` のうち 1 文字で指定します (`l`/`c`/`r` の文字数がカラム数になります)。また、これらの文字の間や左右端に「`|`」を挿入すると、そこに縦罫線が入ります。具体例で見てみましょう (**center 環境**は表全体を中央揃えするために使っています)。

```
\begin{center}          ←見ばえのため
\begin{tabular}{c|ll}
AND 演算 & 0 & 1 \\
\hline                  ←横罫線
0       & 0 & 0 \\
1       & 0 & 1 \\
\end{tabular}
\end{center}
```

上のようにすると、次のように表示されます。見れば分かる通り、表の内部では「`&`」がカラムの区切り、「`\\`」が 1 行終わり、「`\hline`」が横罫線を表します。

AND 演算	0	1
0	0	0
1	0	1

ところで、表の一部について「セルを結合」したいことがよくあります。横方向の結合は`multicolumn`コマンドを使ってできます (縦結合もできますがここでは省略)。たとえば、先の例の 2 つ並んだ「0」を 1 個にまとめたい場合は、0 の行を次のようにします。

```
0       & \multicolumn{2}{|c}{0} \\  ← 2 セル結合、罫線+中央揃え、内容は「0」
```

結果は次のようになります。

[6] たとえば、こんな具合になりますね。

AND 演算	0	1
0	0	
1	0	1

2.7 数式

実は、TeX 属の整形系はもともとは**数式**を美しく打ち出したい、という目的のもとに開発されたものなので、数式機能はとても充実しています (そのため凝り出すと大変ですから、ここでは簡単に説明します)。まず、数式を文中にはさむときは`$`で囲みます。たとえば、

…と書くと`$x^2 - a_0$`のような具合になります。

と書くと $x^2 - a_0$ のような具合になります。ここで出てきたように、肩字は`^`、添字は`_`で表せます。その他、数学に出てくる記号はたとえば

```
\equiv \partial \subset \bigcap \cdots \sum \int
```

のように書くと $\equiv \partial \subset \bigcap \cdots \sum \int$ のように出てきます。また、文中ではなく独立した行にしたければ、`$`の代わりに`$$`ではさむか、`\begin{displaymath}` …`end{displaymath}`で囲みます。たとえば

…と書くと`$$f(t) = \sum_{j=1}^m a_j e^{i\lambda_j} t$$`になります。

と書くと

$$f(t) = \sum_{j=1}^{m} a_j e^{i\lambda_j} t$$

になります。このように、肩字や添字のグループ化には`{}`を使います (丸かっこは数式本体のために使います)。もっと網羅的な記号一覧などは TeX の入門書などを参照してください。

演習 6-3 LaTeX の数式機能を使って、次のようなものを 1 つ以上 (できれば全部) 指定してみなさい。

a. $a_0 + a_1 + \cdots + a_n$

b. $\sum_{i=1}^{N} \int_0^i f(x)dx$

c. $x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3}}}$

できれば、それぞれについてもう少し複雑な「技」を追加してみるとなおよいです。

「◎」の条件: (1) 小問 2 つ以上に回答してあり、(2)「できれば」の部分までやっており、(3) どう工夫したか、何が分かったかなどの考察が (担当から見て) 適切であること。

演習 6-4 LaTeX を使って自分の自己紹介の文書を作成してみなさい。タイトルと章立ては必ずおこなうこと。加えて次の機能から 1 つ以上 (できれば全部) を活用すること。文書内容は「あああ」みたいな意味のないものではなく、それらの機能を使う必然性のある形で使ってください。

a. 箇条書きや引用。できれば、箇条書の中に箇条書を入れて何が起きるか観察できるとなおよいでしょう。
b. 脚注。できれば、脚注の中で使えない機能として何があるか試してみるとなおよいでしょう。
c. 表。できれば、複数カラムの結合も使ってみられるとなおよいでしょう。

「◎」の条件: (1) 小問 2 つ以上に回答してあり、(2)「できれば」の部分までやっており、(3) どう工夫したか、何が分かったかなどの考察が (担当から見て) 適切であること。

3 HTML+CSS

3.1 WWW とマークアップ言語

マークアップの理屈は分かったけれど、やはり見たまま (WYSIWYG) 方式の方が分かりやすいし使いやすい、と思われたかも知れません。しかし実は、見たまま方式が「使えない」場合というのも存在します。しかも、皆様がいつも目にしているもの — Web ページがまさに「そういう場合」なのです。なぜだか分かりますか?

ワープロであれば、出力する紙サイズが決まっていて、その紙に合わせて配置や文字のサイズを決めて行くことで「見たまま」を自由に調整できます。しかし、Web ページはどうでしょうか? Web ページには「紙サイズ」は存在しませんから、「1 行何文字詰めで文書を作る」ことは不可能です。マシンによって使えるフォントも文字サイズも変ってきますから、「この表題は MS 明朝の 24 ポイント」と指定しても、そのフォントがないかも知れません。

ではどうすればいいのでしょうか? できることは「ここは表題」「ここは段落」「ここは箇条書き」などと「構造を」指定しておいて、「ブラウザが画面に表示する時に」窓の幅や使えるフォントに合わせて整形してもらうことです。つまり意味づけ方式でマークアップするしかないわけです。

「でも私は WYSIWYG ツールで Web ページを作っているが」という人もいるかも知れません。しかし、実はそれは「WYSIWYG みたい」なだけで、本当の WYSIWYG ではないのです。というのは、あなたが使っているマシンと表示能力や画面サイズの違うマシンに行ったら、どのみち「その通りに」表示することは不可能なのですから。

なお、WYSIWYG ツールが無意味だというつもりはありません。とりあえず「自分のマシンならこんな仕上り」という様子を見ながら編集できるのはそれなりに便利だと思います。しかし、Web アプリケーションなどでプログラムから HTML を生成する場合には、どのみち直接 HTML を扱う必要がありますから、HTML+CSS についてどのようなものかを知っておくことはやはり必要でしょう。

3.2 HTML の概要

HTML(HyperText Markup Language) については前回も既にやったので、ここではごく簡単なまとめから始めます。HTML ではマークアップの部分を**タグ** (tag) と呼び、マークアップで囲まれた文書の範囲のことを**要素** (element) と呼びます。要素の構成は次の 2 種類です。

<名前 オプション…> …内容… </名前> ←タグが対になる要素
<名前 オプション…> ←単独タグだけの要素

HTML の用語では「オプション」の部分を**属性** (attribute) と呼びます。次にごく簡単な HTML を再掲しておきます。HTML が指定するのは文書の「構造」だということを確認しましょう。

```
<!DOCTYPE html>
<html>
<head>
<title>○○'s page</title>
</head>
<body>
<h1>○○です。</h1>
<p>…挨拶ないし自己紹介を書く…</p>
</body>
</html>
```

ここにあるものも含め、前回までに出て来た要素には次のものがありました。

- `<!DOCTYPE html>` — 以下が HTML 文書であることをあらわす。
- `<html>…</html>` — HTML 文書全体をあらわす。
- `<head>…</head>` — ヘッダ (この文書に関する情報を記述する部分) をあらわす。
- `<title>…</title>` — 文書のタイトルをあらわす。
- `<body>…</body>` — 文書本体 (ブラウザの窓の内側に表示される内容全体) をあらわす。
- `<h1>…</h1>` — レベル 1 の見出し (大見出し) をあらわす。さらに小さいレベルの見出しとして`<h2>…</h2>`〜`<h6>…</h6>`まで使うことができる。
- `<p>…</p>` — 通常の段落をあらわす。
- `<pre>…</pre>` — 要素内の空白や改行をそのままにする。
- `<em>…</em>` — 要素の範囲を強調表示する。☆
- `<a href="`URL`">…</a>` — リンクをあらわす。☆

HTML では「<」、「>」、「&」は特別な意味があり、そのままでは使えません。代わりに「`<`」、「`>`」、「`&`」と書いてください。

なお、本体 (body の中) に使う要素には、**ブロック要素**と**インライン要素**の 2 種類があります。ブロック要素は段落相当のものであり、その前後では改行されますし、段落 (p 要素) 内・インライン要素内に入れられません。一方、インライン要素は段落内の文字範囲に相当するので、前後で改行されず、段落や他のインライン要素内に入りますが、ブロック要素を囲むことはできません。ここまでに出たインライン要素は☆のついたもの (a 要素と em 要素) だけです。

3.3 構造と表現の分離、スタイルシート

ここまで「HTML は文書の構造を規定する」と説明してきました。しかし、実際に世の中の Web ページを見ていると、色や配置などの「表現」がざまなまに工夫されています。自分のページでもこれらの表現を行なうにはどうすればよいでしょうか? 実は過去には HTML にも「色をつける」「フォントを変える」「中央そろえ」など表現を指定する要素がありました。しかし、HTML 4.0 からはこの種の機能はすべて「非推奨」になり、代りに「スタイルシート」と呼ばれる方法で表現を指定するようになりました。

なぜでしょうか? たとえば HTML では「大見出しを全部集めて来て一覧を作る」などの作業は (grep などのツールを使って) 簡単に行なえますが、そのとき見出しの中に「ここは青い色」など別のタグが混ざっているとうまく取り出せなかったり、または取り出したものにタグが混ざるなど、面倒なことが起きます。

それに、大見出しを青い色にするとしたら、全部の大見出しをそのように統一したいわけですが、すべての大見出しの所に余分に「ここからここまで青」というタグをつけて行くのも無駄な話です。コンピュータで処理するのだから、「すべての大見出しは青」と「ひとこと」言えば済むようであるべきではないでしょうか? (図 9)。スタイルシートとはちょうどそのように、つまり文書の構造のそれぞれについて「このような部分はこのような表現」という形で表現を指定する機能なわけです。

HTML と組み合わせるスタイルシート指定言語としては **CSS**(Cascading Style Sheet) が使われます。HTML に CSS の指定を追加する方法としては、次の 3 通りがあります。

(1) CSS 指定を次のように **style 要素**の内側に書く。style 要素はヘッダ部分に入れる必要がある。

```
<style type="text/css">
CSS 指定…
…
</style>
```

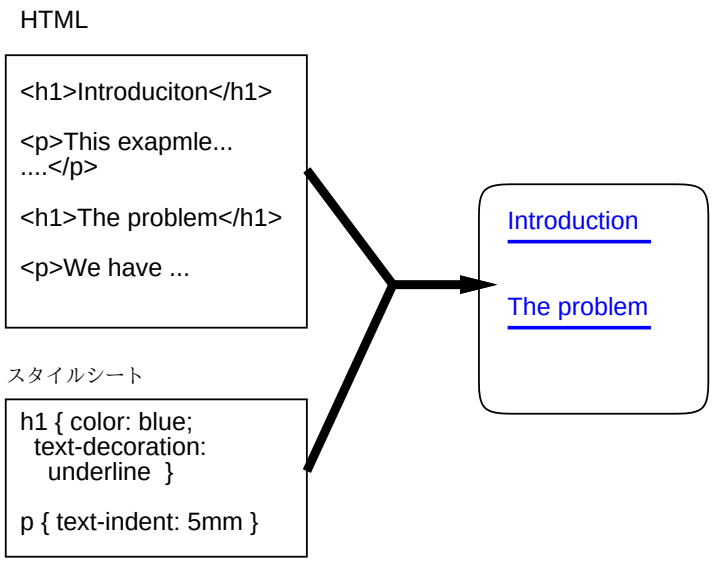

図 9: スタイルシートの概念

(2) CSS 指定を別ファイルに入れ、HTML のヘッダ部分に次のような **link 要素**を入れる (ここでは CSS 指定が `mystyle.css` というファイルに入っているものとしました)。この方法はやや繁雑だけれど、1 つの CSS ファイルを複数の HTML ページに適用させられる。

```
<link rel="stylesheet" href="mystyle.css" type="text/css">
```

(3) HTML の各要素に **style 属性**を指定し、その値として CSS 指定の本体部分を書く。特定の要素だけに表現を指定する場合に使う。

```
<p style="color: blue">この段落は青い。</p>
```

以下では (1) の方法を使うようにします。CSS 指定を入れたページの例を示します (図 10)。

```
<!DOCTYPE html>
<html>
<head>
<title>sample</title>
<style type="text/css">
body { background: rgb(220,200,255) }
p  { background: rgb(200,255,240) }
p  { padding: 3mm 5mm }
</style>
</head>
<body>
<h2>スタイルシートとは</h2>
<p>HTML は文書の「どこからどこまでが何を意味するか」を指定します
が、その見え方 (色やあけ方など) は指定しません。</p>

<p>現在では、HTML を補完する形で「スタイルシート」と呼ばれる機能
を組み合わせることで見え方を指定します。</p>
</body>
</html>
```

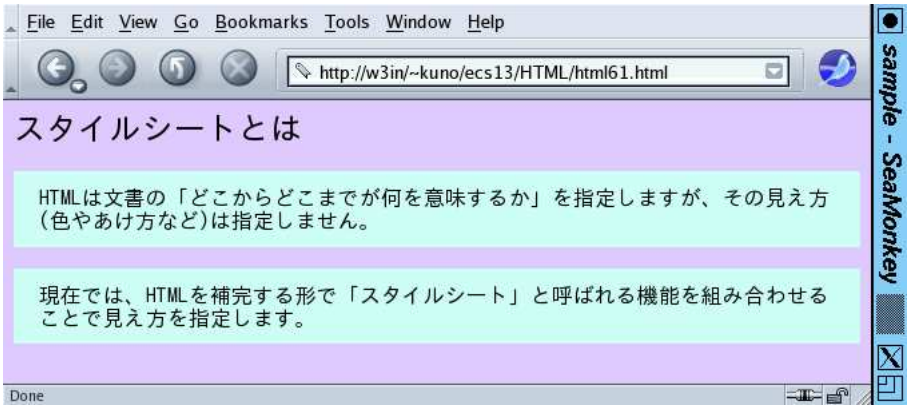

図 10: スタイルシートを使ったページ

3.4 CSS の指定方法

順序が逆になりましたが、CSS の指定方法について説明しましょう。まず、CSS の指定は「規則」の集まりで、1 つの規則は次の形をしています。

セレクタ { プロパティ: 値; プロパティ: 値; … }

セレクタは、とりあえず HTML のタグと思ってください。つまり「この要素はこう表現する」という指定です。**プロパティ**は、色や字下げなどです (すぐ後で説明します)。そして、それに対する値を指定します。値の指定方法は次の通り。

- 文字サイズの指定方法: `12pt` (ポイント数)、`xx-small`、`x-small`、`small`、`medium`、`large`、`x-large`、`xx-large`、百分率 (下記。本来のサイズの何倍/何%という形で指定)。
- 長さの指定方法: `1px`(画面上の点)、`1cm`(センチ)、`1em`(文字「m」の幅 1 個ぶん) などがある。
- 百分率: `%`をつけた数値。ページ幅に対する割合などの指定に使用。
- 色の指定方法: `black`、`blue`、`gray`、`green`、`maroon`、`navy`, `olive`、`purple`、`red`、`silver`、`white`、`yellow`、`rgb`(赤, 緑, 青) ただし赤/緑/青は 3 原色の強さを 0〜255 の数値で表す。
- ファイルや URL: `url`(ファイル名)、`url`(*URL*)。

CSS プロパティの代表的なものとしては次のものがあります。

- `color`: 色 — 文字色を指定。
- `background`: 色 — 背景色を指定。
- `margin`: 長さ — 要素の周囲のマージン (余白) 幅を指定。4 つの長さを指定すると「上、右、下、左」の長さを指定したことになる。2 つだと「上下、左右」、1 つだと 4 周全部がその長さに。
- `padding`: 長さ — 要素の周囲のパディング (詰めもの) 幅を指定。指定方法は margin と同様。
- `border`: 形状 色 長さ — 要素の枠を指定。形状として、`solid`(均一)、`dashed`(点線)、`double`(2 重線)、`ridge`(土手)、`groove`(溝)、`inset`(くぼみ)、`outset`(出っぱり) 等が指定できる。色は枠の色、長さは枠の幅を指定。
- `text-indent`: 長さ — 段落先頭の字下げ幅を指定。
- `text-align`: 種別 — `left`、`right`、`center` で左そろえ、右そろえ、中央そろえを指定。
- `text-decoration`: 文字飾り。`underline`(下線)、`blink`(点滅) 等を指定。
- `font-size`: 文字の大きさを指定。

演習 6-5 先の例の HTML をまずそのまま表示し、次に段落や本体の背景色を変更してみなさい。その後、以下の課題から 1 つ以上 (できれば全部) 選んでやってみなさい。

a. 先頭の見出しの文字色を変更し、下線つきにしなさい。さらに、文字サイズを本来の 3 倍にしなさい。さらに太い枠線で囲みなさい。できれば、枠線がページ幅一杯でなくページの中央部 50%の範囲になるとなおよいです。[7]

b. 段落先頭の字下げや、段落の左右マージン、上下マージンを調整して、段落が連続しているときも区切りが見やすいようにするにはどのように調整すればいいかを検討しなさい。できれば、h4〜h6 のような小見出しに続いて段落がある場合についても (小見出しのスタイルも含めて) 検討できるとなおよいです。

c. em 要素を使って段落の一部を囲んで強調の様子を見てみなさい。さらにそれがより自然なようすに見えるように調整を試みなさい。できれば、a 要素についてもスタイルを変更して同様に検討してみるとなおよいでしょう。[8]

「◎」の条件: (1) 小問 2 つ以上に解答されており、(2) いずれも「できれば」の部分までやってあり、(3) 適切な (と担当が判断する) 説明・考察が書かれていること。

3.5 HTML 要素とスタイル指定の追加

HTML の追加として、LaTeX にもあった箇条書きと表について説明しておきます。まず番号なしの箇条書きは次のようになります。

```
<ul>
<li>.........................</li>
<li>.........................</li>
</ul>
```

ここで`<ul>...</ul>`を`<ol>...</ol>`にすると番号つきになります。LaTeX でいうと **ul 要素**は itemize 環境、**ol 要素**は enumerate 環境に相当します。箇条書きの項目はどちらも **li 要素**として含めます。

次に表について説明しましょう。こちらはもう少し複雑で、表全体を表す **table 要素**、その中の 1 行ずつを表す **tr 要素**、そして行の中に入る各セルを表す **th 要素**または **td 要素**の 3 レベルから成ります。HTML の上で次のように縦横に揃えて書くと分かりやすいでしょう (「`border="2"`」という指定は表の罫線を幅 2 で描くという指定。無いと罫線も描かれない)。

```
<table border="2">
<tr><td>A</td><td>B</td><td>C</td></tr>
<tr><td>1</td><td>2</td><td>3</td></tr>
</table>
```

th と td の違いは、前者が見出し用、後者が一般用で、見出し用の方が少し強調表示になります。また、th も td も「`colspan="個数"`」「`rowspan="個数"`」という属性が指定でき、これによって LaTeX と同様横や縦にまたがったセルが作れます。

最後にもう 2 つだけ、範囲指定のタグも紹介しておきます。

- `<div>…</div>` — ブロックの範囲を指定。
- `<spam>…</span>` — インラインの範囲を指定。☆

[7] ヒント: 左右のマージンをページ幅の 25%ずつにすればいいはずですね。

[8] ただし、実際に Web サイトで使う時には、a 要素のスタイルを変更してしまうとリンクだということが分かりづらくなるので、やらない方がいいとされています。このことも含めて確認してください。

範囲指定というのは何でしょうか? それは、これまで出て来た要素がすべて「こういう意味のもの」という要素だったのに対し、特定の意味が決められていなくて「範囲」だけを指定する要素ということです。

実際には範囲を指定した後で、「この範囲はこういう意味だからこういう見せ方にする」という個別指定をすることで使います。実は個別指定は他の要素でも使えます。つまり、ここまで説明してきた方法は、「全段落をこうする」「全見出しをこうする」などの指定だったのに対し、次の 2 通りの方法で「ここはこうする」と言えます。[9]

(1) 「`<div id="p01">...</div>`」のように要素の開始タグに **id 属性**を指定し、セレクタで次のように「`#`」に続いて ID を指定する。

```
#p01 { color: red }
```

(2) 「`<span class="important">...</span>`」のように要素の開始タグに **class 属性**を指定し、スタイル指定で次のように「`.`」に続いてクラス名を指定する。

```
.important { color: red }
```

なお、id と class の違いですが、id は「要素ごとに異なる値を指定する」ものなので「特定のこれ」という形の指定に使い、class 属性は複数要素に同じものを指定できるので「このクラスを指定したものすべて」という形で使います。

新規の HTML 要素は以上ですが、主に table と div で活用する CSS 指定についても追加しておきます (最初のものが table 用、残りは div でよく使います)。

- `border-collapse: collapse` — table 要素専用で、これを指定した表は隣接するセルの境界が 1 本の線になる (これを使う時は th や td の border も併せて指定することが多い)。
- `position:` 位置指定。`absolute`(絶対位置を指定)、`fixed`(画面上での絶対位置を指定)、`relative`(本来あてはまる位置からのズレを指定) のいずれかが指定できる。
- `top:`、`left:`、`bottom:`、`right:` 要素の上端、左端、下端、右端の位置を指定。
- `width:` 長さ、`height:` 長さ — この要素を整形する幅や高さを指定できる。div に対して使うことが多い。
- `float:` 流し込みの指定。`left`(左に寄せて右に流し込む)、`right`(右に寄せて左に流し込む)、`none` が指定できる。

色々出てきましたが、これらを使った HTML の例を示しておしまいにします (図 11)。[10]

```
<!DOCTYPE html">
<html>
<head>
<title>sample</title>
<style type="text/css">
.important { color: red }
table { border-collapse: collapse }
th, td { padding: 1em }
p { text-indent: 1em }
#col1 { float: right; width: 50%; border: green double 4px }
```

[9] ここでは div と span で例を示していますが、他の任意のブロック要素/インライン要素に対してもこのような指定ができます。

[10] 本当は、これらの機能を使って「どのようにページの見栄えを設計するか」が大切なのですが、今回はそこまで時間が無いので機能の説明までになりました。

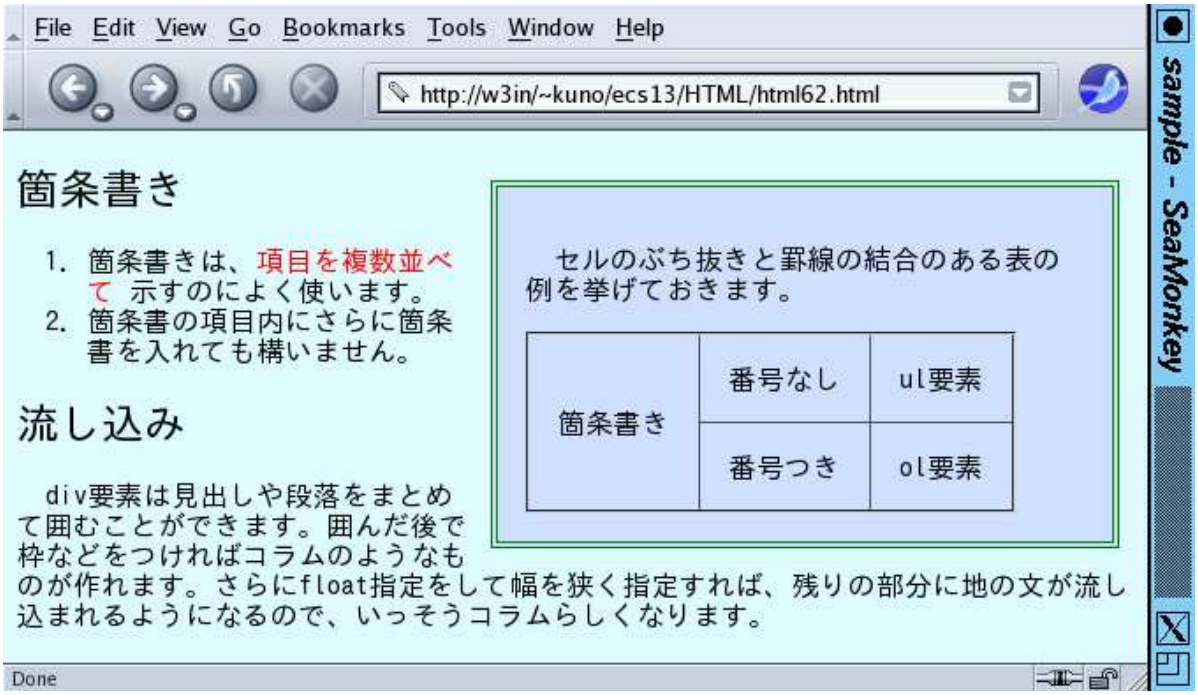

図 11: さまざまな機能を使ったページ

```
#col1 { padding: 1em; margin: 1ex; background: rgb(200,220,250) }
</style>
</head>
<body>
<div id="col1">
<p>セルのぶち抜きと罫線の結合のある表の例を挙げておきます。</p>
<table border="2">
<tr><th rowspan="2">箇条書き</th><th>番号なし</th><td>ul 要素</td></tr>
<tr><th>番号つき</th><td>ol 要素</td></tr>
</table>
</div>

<h2>箇条書き</h2>
<ol>
<li>箇条書きは、<span class="important">項目を複数並べて</span>
    示すのによく使います。</li>
<li>箇条書の項目内にさらに箇条書を入れても構いません。</li>
</ol>

<h2>流し込み</h2>
<p>div 要素は見出しや段落をまとめて囲むことができます。囲んだ後で
枠などをつければコラムのようなものが作れます。さらに float 指定をし
て幅を狭く指定すれば、残りの部分に地の文が流し込まれるようになる
ので、いっそうコラムらしくなります。</p>
</body>
</html>
```

演習 6-6 先の例の HTML をまずそのまま表示し、次に箇条書きの種類や表のセルの構成を変更してみなさい。その後、以下の課題から 1 つ以上 (できれば全部) やってみなさい。

a. 春学期か秋学期の適当なモジュール 1 つを選び、そのモジュールに自分が選択する科目の一覧表 (どんな科目かを自分の言葉で説明する) を作りなさい。できれば見やすく工夫されているとなおよい。

b. 箇条書きの中に箇条書きを入れることができるので、実際に試してみなさい。できれば、番号付き箇条書きの中に番号つき箇条書きを入れたら番号のようすがどうなるか、番号なし箇条書きの中に番号なし箇条書きを入れたらどうなるかなどを探究できるとなおよい。

c. `<div id="gssm">GSSM</div>`という要素をページのどこかに入れた後、その要素を `position: absolute`、`top: 200px`、`left: 200px` で指定してページの中に浮いて現れるようにしなさい。できればその後、文字を大きく枠を太くして超目立つようにできるとなおよい。

「◎」の条件: (1) 小問 2 つ以上に解答されており、(2) いずれも「できれば」の部分までやってあり、(3) 適切な (と担当が判断する) 説明・考察が書かれていること。

4 まとめ

今回はテキストと文書について取り上げ、整形方式や見たまま/マークアップの区分について説明した後、LaTeX と HTML という 2 種類のものを対象にマークアップの具体例を見てみました。どちらもマークアップなので似た面もありますが、用途 (紙の文書作成と Web ページ作成) の違いのため、異なっていることも結構あることがお分かりかと思います。