# すうけん

数学研究部 平成十九年度活動報告

# -目次-



## はじめに

本日は数学研究部の展示においでいただき有り難うございました。今年度から活動内容の発表の場としてこの小冊子「すうけん」を発行することになりました。数学研究部は現在はプログラミングを中心に活動しています。プログラムといってもほとんどがゲームを作ることになるのですが，ゲーム制作を通して，数学的な考え方やコンピュータサイエンスの入り口を学び，そして何よりも，「ものを作る楽しさ」を感じてくれたらと考えています。

今年は2作ともシューティングゲームですが，中学生はC言語の学習，高校生はコンパイラの制作が目標です。コンパイラは普段コンピュータを使っているときには意識しないものですが，ほとんど全てのプログラムはコンパイラを通して作られている重要なものです。サクサク動くゲームを作りたい一心で大学生のレベルに挑戦しています。内容をご一読いただければ幸いです。 （顧問 堀井雅司）

# コンピュータで考える「ことば」の仕組み

高等部2年　山内　健二

## *1*　自動翻訳使ったことありますか？

最近は外国語の自動翻訳ソフトなるものがあります。初期のころのは結構(今でもそれなりにあるけど)珍訳してきたものですが、このごろはそうでもなくなってきていて便利な存在であります。

今回僕はコンパイラというものを作ったのですが、これはすなわちコンピュータで使われるプログラミング言語の自動翻訳ソフトに相当します。プログラミング言語は英語や日本語などの自然言語ではないので、語順などが明解でまだ簡単に翻訳が可能なのです。

## *2*　プログラミング言語とは？

そもそもプログラミング言語とは何でしょう。
コンピュータが理解できるのは 0101011…という二進数です。
ですので人間がコンピュータにしてほしいことを命令する場合は、二進数(機械語)を使うしかありません。

しかしこれではあまりにも非効率です。
そこで考え出されたのがプログラミング言語とコンパイラなのです。

プログラミング言語はある程度人間が理解しやすいように作らた命令群です。例えば「Print テストプリント」とすると、なんとなく画面に「テストプリント」という文字が出力されるように感じませんか？
このように、人間の判断力にやさしい存在のコンピュータ言語ですが、しかしながらコンピュータは理解できません。そのためコンピュータが理解できる機械語に最終的に「翻訳」する存在が必要です。それがコンパイラです。

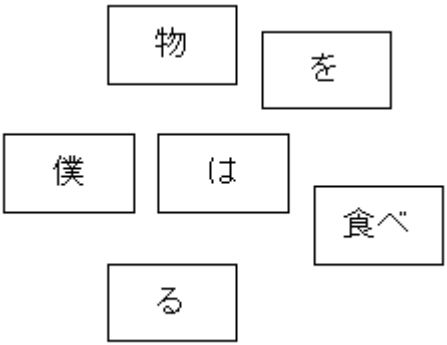
## *3* コンパイラを作ろう！ ～構造篇～

プログラミング言語の実行には、コンパイラというものが必要なのはわかりましたが、具体的にどのようなものを作るのでしょうか？
いきなりプログラミング言語で考えるのは、慣れていないと難しいので日本語(自然言語)で考えてみます。

例えば「僕は物を食べる。」という文があったとします。

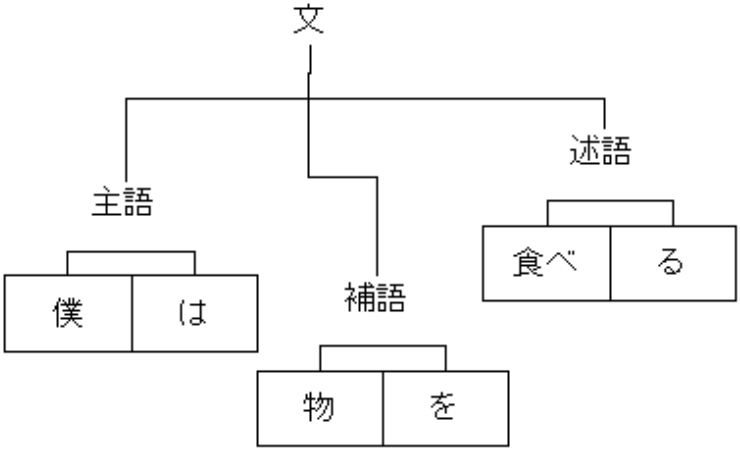
まずコンパイラは一文字一文字読み込んでいき、これは名詞、これは助詞、これは動詞というように単語に分解します(図Ⅰ)。
この作業のことを**字句解析**といいます。

**図Ⅰ**




しかしながら図Ⅰの様に単語に分けてもばらばらでは、訳がわかりません。これをつなげて文にするのが**構文解析**という作業です。簡単な方法だと前からこの作業は行われます。これはこの「僕は」は主語に相当するはずだ、とか「食べる」は述語に相当するな、というのを判断するものです(図Ⅱ)。

**図Ⅱ**




こうして「僕は物を食べる」というのが文であるというのはわかりました。しかしながらコンピュータからしてみれば「で？」と聞きたいところです。

文であることはわかったけれど一体どういう「意味」なんだろう…ということを考えなければなりません。

これをするのがそのままずばり**意味解析**という作業です。

例えば通常の現代日本語で「何々せざるを得る」という言い方はありません。つまり間違った文ということになるのです(叙述用法としての特殊な用法は考えないものとして)。

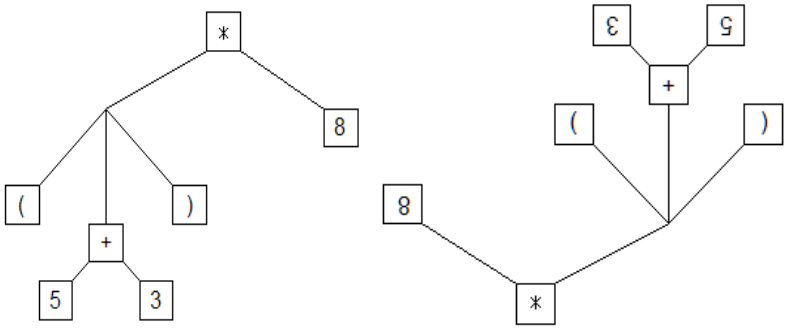
さて、ここでようやく「僕は物を食べる」というのが文であり、かつ「僕は何か物を食べる(ようとしている)」のだな…というのが理解することができるわけです。

かなり駆け足で見てきましたがこれがコンパイラの主な仕組みです。すなわち**字句解析→構文解析→意味解析**という流れに沿ってプログラミング言語は解析されていくのです。

さて、すこし自然言語の例では説明できなかったのですが、構文解析では通常、構文解析木というものを作成します。

これは数式で考えてみるとわかりやすいです。「(5+3)*8」という式があり、これを計算したいとします([*]はコンピュータの世界では[×]を意味します、[×]という記号はなぜかありません)。これの構文解析木を作ると図Ⅲ左のようになります。

**図Ⅲ**




木構造とは「根」(ここでは[*])というデータ部分があり、そこから「枝」、「葉」を下に生やして行ってデータ構造を作るためにこのように呼ばれます。
これの「根」からたどっていって、データを解析していくことによって、この場合で言えば「(5+3)*8」の答えである64が導き出せるのです。
ちなみに図Ⅲ右のように逆さにしてみると名前通り「木」みたいに見えません？

さて、ここまでがプログラミング言語を解析する方法なのですが、これを実行するにはいくつかの方法があります。
代表的なのは**インタプリタ**、**コンパイラ・インタプリタ**、**コンパイラ**の3つです。インタプリタは毎回言語のコード(ソースといいます)を読み込み、構文木を生成して実行します。

コンパイラ・インタプリタはソースを読み込み、構文木を生成するところまでは同じですが、この際に実行ではなく中間言語と呼ばれるものを別に出力します。これは機械語をそのまま直訳したようなもので、人間にはわかりにくいものです。理解しようと思えばできないこともないですが…。中間言語は、コンピュータには理解できないので、さ

らにこれを実行するものが必要です。

さらにコンパイラは中間言語ではなく、機械語を直接出力します(この作業自体をコンパイルといいます)。

題名にもありますが、僕が今回作ったのはコンパイラ・インタプリタでした。

## *4* コンパイラを作ろう！ ～実行篇～

実際に作ってみましょう、といいたいところですが、実際のプログラムのコードを載せても~~僕のソースが汚くて~~あんまりよくわからないと思うので、コンパイラ・インタプリタで中間言語をどのように実行するかを見てみたいと思います。

例として以下のようなコードを考えます。

/例コードⅠ

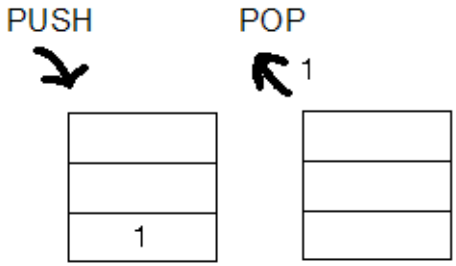
```
PUSH 1
PUSH 2
ADD
POP
```

…まったく意味がわかりませんね。これぐらい中間言語というものは意味不明です。ちなみにこれは「1+2」の実行を意味します。

実はこれを解説する前にひとつ重要なデータ構造を紹介しておかなくてはなりません。**スタック**といいます。

スタックはよくある例えとしては積み上げられた本にたとえられます。本を積み上げていくとき、普通下からつんでいきます。ここで積み上げられた状態を崩さずに本を取っていくには、特殊な技を使わなければ上から取っていくしかありません。ここで本をデータに置き換えたものがスタックなのです(図Ⅲ)。

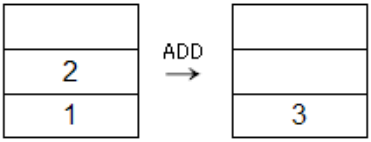
図Ⅳ




スタックにデータを入れていくことを PUSH、データを取り出すことを POP といいます。

さて、これで多少例コードの意味がわかるかもしれません。
PUSH 1 、PUSH 2 でまず以下の図Ⅴ左のようになります。

図Ⅴ




ここで ADD を行うと、スタックの底にある2つのデータを足し合わせます。すると図Ⅴ右のように、スタックには 2+1=3 の結果である 3 が残るわけです。

ところで PUSH、POP をつかわないでこれを示すと 23+になります。日本語で考えるとわかりやすいですが、2「と」3「を」「足す(+)」ということです。この表記だと「2+3*8」は「38*2+」となります。これを**逆ポーランド記法(後置記法)**といい、スタックの例で示したようにかなりパソコンとの相性がよいのです。

このようにほとんどのコンパイル言語はスタックを用いて作業をしていきます。さまざまなデータの取り扱いをすべてスタックで行います(内部的にはすべて数字で扱われる)。

これできちんと実行されればコンパイラの完成です。なんと自分でプログラミング言語を作ってしまいました！星の数ほどあるであろうプログラミング言語にひとつ追加されるわけです(実用的なものではありませんが…)

## *5* コンパイラを作ろう！ ～活用篇～

長々と説明してきたわけですが、そもそも何故このような言語処理を作ろうとするのでしょうか？

ひとつは純粋にコンピュータ処理の勉強のためにです。個人が作ったプログラミング言語なんて、実行速度とか色々考えてもプロが作ったものに比べればかなり劣りますから研究程度にしか使えません。

もうひとつはゲームを作るとき(普通の人はあまりやりませんが)にスクリプト、組み込み言語として使えるということです。これはかなり役に立ちます。例えば敵の動きを直接プログラムするのは実は結構面倒くさいんです。外部ファイルとしておいておいて、これをゲームを実行するときに読み込ませるようにすると、結構拡張性が高くなります。

まぁこんなものですね。実際には組み込み言語でも優秀なものがあるので、素人が作るのは酔狂な事なのですが趣味ですし。

## *6* 総括

というわけでこのような感じです。はっきり言ってかなりはしょってます。

わかってくれるとありがたい…というぐらいです。何せまともに文章に起こすと、本一冊では足りないぐらいな内容なのでこのように短くせざるを得ないわけです。

しかしもし、この拙い文章でプログラムに興味を持ってくれた人がいらっしゃれば、冥利に尽きます。

## *6.5* 参考文献

実際にプログラムを作る際に参考になった書籍・Web サイトです。
ここで感謝申し上げます、というかこちらを読んだほうが絶対ためになります。


//書籍

+石田 綾『スモールコンパイラの制作で学ぶプログラムの仕組み』
中田 育男 監修、東京、技術評論社、2004 年

+中田 育男『コンパイラ (新コンピュータサイエンス講座)』
東京、オーム社、1995 年

+randy『いまどきのプログラミング言語の作り方』
東京、毎日コミュニケーションズ、2005 年

//Web サイト

+前橋 和弥「プログラミング言語を作る」
(http://kmaebashi.com/programmer/devlang/index.html)

+GameDev.net(http://www.gamedev.net/)
――"Scripting Languages and Mod Development"
(http://www.gamedev.net/reference/list.asp?categoryid=76#118)

+HPCS laboratory[筑波大学ハイパフォーマンス・コンピューティング・システム研究室](http://www.hpcs.is.tsukuba.ac.jp/)
――佐藤 三久教授「 プログラミング言語処理 講義資料 一覧」
(http://www.hpcs.is.tsukuba.ac.jp/~msato/lecture-note/comp-lecture/)

その他 Squirrel、Lua、xtal、Io 等公開されている様々な言語のソースを参考としました、殆ど読めていませんが (ぇ。
色々と不完全なものですが、そこはご容赦ください。

//おまけ 本気で勉強してみたい人へ

+A. V. エイホ、R. セシィ、J. D. ウルマン『コンパイラ―原理・技法・ツール』〈1〉〈2〉
原田 賢一訳 東京、サイエンス社、1990 年

+中田 育男『コンパイラの構成と最適化』
東京、朝倉書店、1999 年

軽く自分のプログラミング能力の無さに絶望できる難易度です。少なくとも私は。でも本気でやるには避けられない良書であります

# THE SEA Ⅱ キャラクター説明

中2A 紅山丈明

今年は去年と同じくC言語でシューティングゲームをつくった。本を見ながらソースコードを入力し全面的に手を加えて、スクロールゲームと弾幕シューティングゲームの合わせ技のようなものにした。

**改良した点**

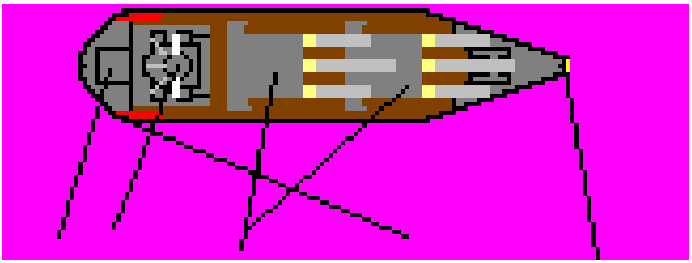
・キャラクターの絵をオリジナルのものにした。
・音楽も自分で作曲して作った。
・自キャラが一発で死なないようにした。
・自キャラの当たり判定の位置を小さくした。
・敵ボスの弾の出方を一方方向から放射状に出るように変えた。
・敵ボスの動きを45度刻みから15度刻みに変え、生物らしい動きにした。
・敵キャラクターをちょっとかわいらしいマスコット的キャラにして、難しいイメージを少なくするようつとめた。
・レベルを3段階作った。
　レベル1は初心者向けで簡単、
　レベル2では地形が複雑で道を間違えたら終わりという所もある。
　レベル3はザコ敵、特に自キャラを追いかけてくる敵が圧倒的に多く、万が一にそこを抜けても、リバイアサンの弾幕と自キャラ狙いの弾、そして予測不可能の動きには、製作者もついてこれない。

結構苦労して作ったと思うが、やはり、まだまだ不完全な所が多いと思う。例えば、ボスは一つ倒すと次のボスが出てくるようにできたら良かったと思う。

来年はロールプレイングゲームなどにも挑戦してみたいと思う。

キャラクター説明

NANI-WA Ⅲ 図



①　②　③　④　⑤

① ライト：夜には点灯して居場所を知らせる。ゲームに全く関係なし。

② 艦橋：当たり判定はここにある。操舵室、発射台がある。

③ 砲塔：計6門の50サンチ砲がある。両脇にレーダーがついている。

④ カタパルト：飾り。ゲームに関係なし。

⑤ 紋章：ビッグスロープシティの金メッキの紋章がついている。


ビッグスロープシティの紋章

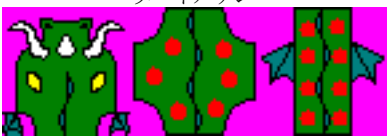
50サンチ砲弾

かわいいザコ敵たち

①　②　③　④

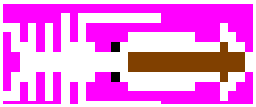
① お魚：口をパクパクさせながら敵を追いかける。
② たこ：THE SEA のマスコット的キャラクター。上下に飛ぶ。
③ ヒトデ：回転する。下に降りながら弾を発射する。
④ ふぐ：ジグザクに動く。

リバイアサン



①　②　③

① 頭部：放射状に弾を発射する。
② 胴体：特になし。
③ ひれ：弾を敵にむけ発射する。数はレベルによってことなる。



ザコ敵の弾　イカ

リバイアサンの弾　ヒレ

# 折り紙で作る正十二面体

数学科　堀井雅司

「数学研究部でなぜコンピュータを？」という質問をされることがある。
確かにコンピュータといえば，ゲームであり，インターネットであり，（面白くもない）数学との間に関係など考えられないかもしれない。しかし，プログラムを組むことは十分に数学的思考を要求する作業で，「創造集団」をめざす数学研究部の活動の中心はまさにこのプログラミングなのだ。そもそも歴史的に見ても，数学者[注1]が世界に先駆けコンピュータの原理を考案していて，これは世界最初のコンピュータといわれるEANIC[注2]の10年前であり・・・・・・，まあ，どうでも良い。
しかし，少しは数学的なことを書いてみよう。

## 正多面体は5種類

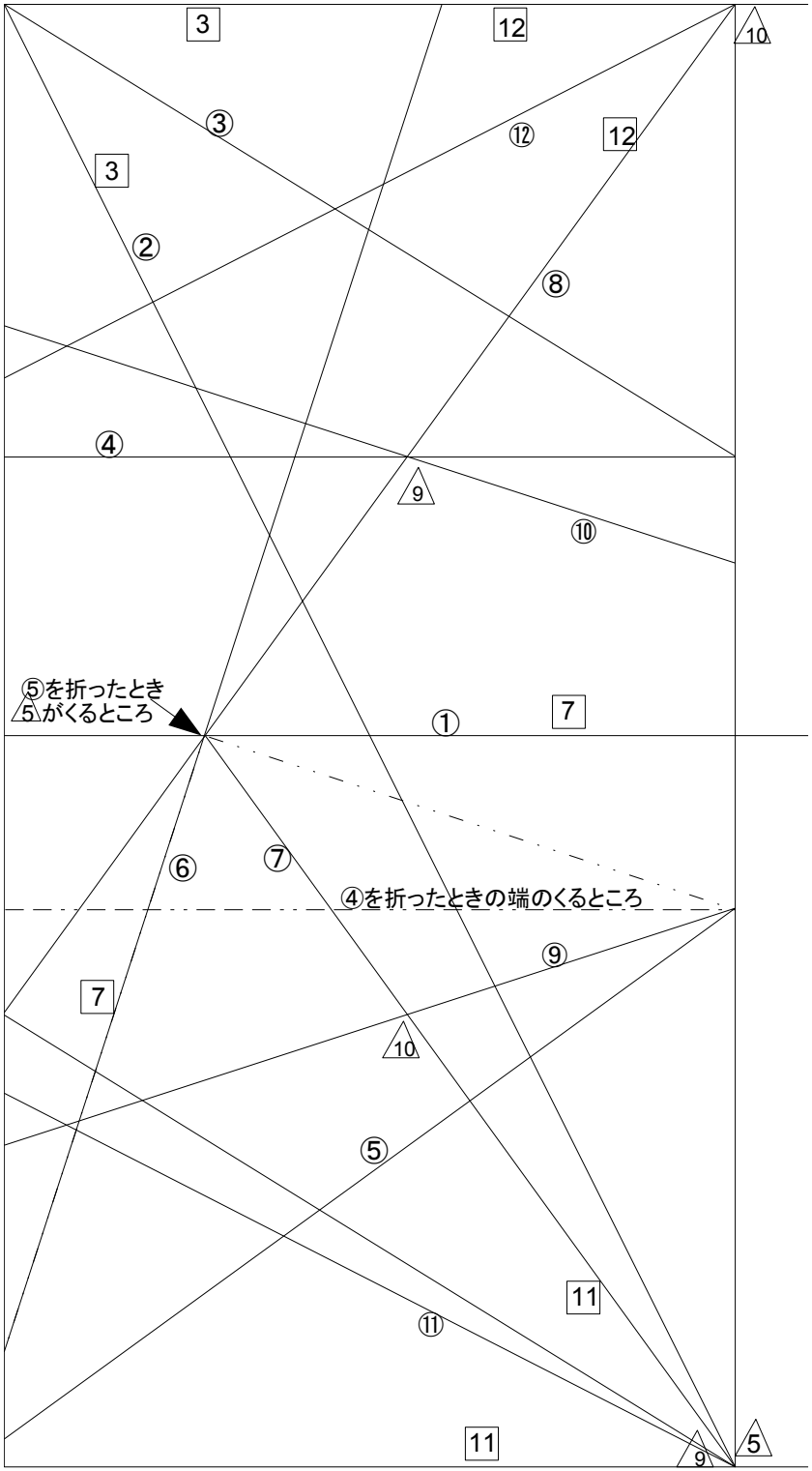
星光学院では中学1年で正多面体を作る授業をすることが多い。正多面体とは，1種類の正多角形からできた多面体のうち，どの頂点も同じ形をしているもののことで，「正四面体」「立方体」「正八面体」「正十二面体」「正二十面体」の5種類しかない。（なぜだろう？）そのなかで，一番イメージしにくい正十二面体を折り紙で作ることにしよう。ただし，1枚の紙で作るわけではない。正五角形のパーツを12個つくって正十二面体を組み立てる方式をとる。

## 正五角形を折る

折り紙は標準的なものでOK。20cm角のものが「お徳用」として売られている。次のページの図はその実物大の折り目だ。正方形を半分に折った長方形から折りはじめる。できるだけ正確に折り，爪でこすってきつく折り目をつけるときれいな作品ができる。
図の○番号は折る順番を表している。図を見て折り方の分かるものは○番号だけしか書いていないが，折り方の分かりにくいものについては□番号や，△番号で折り方を示している。例えば，○2の折り目は□2の辺が重なるように折り，○7の折り目は△7の頂点と交点が重なるように折る。

---

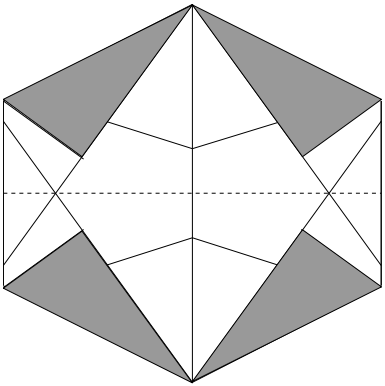
注1：1936 年にチューリングの論文で発表されている。
注2：現在ではアタナソフ＆ベリー・コンピュータ(1939) が最初だといわれている。

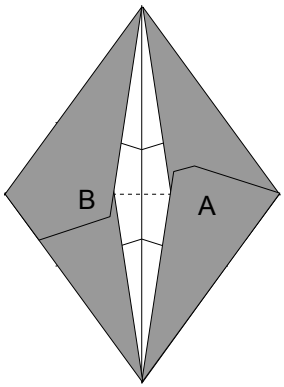

3
12
10
③
⑫
12
3
②
⑧
④
9
⑩
⑤を折ったとき
5 がくるところ
①
7
⑥
⑦
④を折ったときの端のくるところ
⑨
7
10
⑤
11
⑪
11
9
5

まずは正方形を半分に折って，長方形を作る。②以外は2枚重ねて折ると素早く折れる。ただし，正確に折りたいなら1枚ずつ折る。

① 長方形を二等分する線。この線の上に正五角形の頂点がくる。
② 長方形の対角線。2枚に重なった長方形のうち，上の方の長方形だけ折る。余計な折り目を入れないための工夫。
③ □3（上の辺と②の対角線）を重ねるように折る。2枚重ねて折る。
④ ③の下の折り目と上の辺が平行になるように折っている。このとき上の辺のくるところ（図中の破線）を⑤で使う。
⑤ △5が①の線上にくるように折る。これが正五角形の上の頂点になる。
⑥ ⑤で下の辺がくる線に合わせて折る。
⑦ □7が重なるように折る。正五角形の上の辺。
⑧ 正五角形の上の頂点と角を結ぶ線で折っている。
⑨ △9（長方形の角と④⑧の交点）が重なるように折る。
⑩ △10（長方形の角と⑦⑨の交点）が重なるように折る。
⑪ □11を重ねるように折る。
⑫ □12を重ねるように折る。　⑪と⑫は余分な部分の

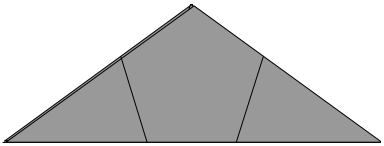
以上で折り目がついたので，一度広げて下のように折りたたんでいく。






四隅の無駄な部分を折りたたみ（左図），右図のAの部分とBの部分を組み合わせながら半分に折ると完成。

完成したら，「腕」の部分を折り曲げて、しっかりと折り目をつける。これでパーツの1つが完成。



## 正十二面体を組む

このパーツを12個折り，上手に組み合わせていくと正十二面体が出来上がる。組むときには，このパーツを「カニ」にみたて，とがった「頭」に「ハサミ」をつっこんで組む。ポケットのない「おしり」には「おしり」が向かい合うことになり，ここはすき間ができる。

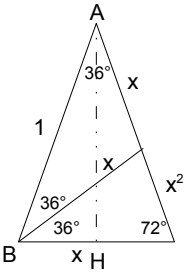
この折り方は同志社大学理工学部の岡崎龍太郎准教授から教えていただいたものだが，「おしり」のところがつながらないので，構造的には少し弱いということだ。このようなすき間のできない折り方もあるようだが，正しい正五角形ではなく，近似形らしい。好みの分かれるところだ。

## 正五角形ができるわけ

正五角形で重要な角は72°や36°で，これをどのように作るかを考える。右のような二等辺三角形を考えると，X の値は

$\frac{-1+\sqrt{5}}{2}$ となる。このとき，BH= $\frac{-1+\sqrt{5}}{4}$ で，

AB:BH=4: $(-1+\sqrt{5})$ となり，これで72°ができる。



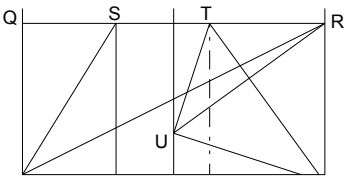

一方，折り紙の方では PS が∠QPR の2等分線になっているので，QS:SR=1: $\sqrt{5}$ となるので，（中略），

QS= $\frac{\sqrt{5}-1}{2}$ TR=TU= $3-\sqrt{5}$ ，

VT= $\sqrt{5}-2$ となり，

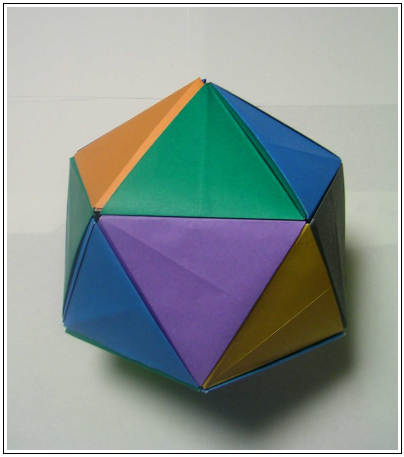
$\frac{\sqrt{5}-2}{3-\sqrt{5}}=\frac{\sqrt{5}-1}{4}$ ゆえ，∠UTV=72°で

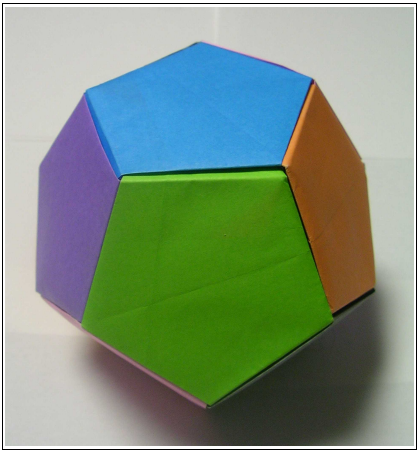
∠TRU=36°となることが分かる。



## おわりに

ユニットによる多面体制作は，多くの愛好家により様々な折り方が考案されている。今回の折り方は，「正方形の折り紙から作れる」「正しい正五角形」と「わりと丈夫」な立体ができることからおすすめである。結構手間ではあるが，折っているとはまってしまう。実際に手を動かしてものを作ることの楽しさを，少だけではあるが味わえる一瞬なのかもしれない。結局，遊んでいるうちに，正二十面体まででできてしまった。皆さんもやってみてはいかがだろうか？「たかが折り紙」と笑うことなかれ。世の中には，これらの立体を1枚の折り紙から折ってしまうというすごい人もいるという。この世界も奥が深いのだ。そして「もの作り」が楽しいと感じたら，そのときは数学研究部をのぞきにきてほしい。





参考文献

「和紙の折紙で楽しむ多面体」 岡崎龍太郎（数理システム学科）
「見方で変わる数学の学び」講演会 同志社大学理工学部

オリガミックス 幾何図形折り紙 I 芳賀和夫 日本評論社

ユニットあらかると ユニット折り紙 2 布施知子 筑摩書房

多面体の折紙 正多面体・準正多面体およびその双対
川村みゆき 日本評論社

## 編集後記

どうも初めまして、そうでない人は…多分いないと思いますがこんにちは。
数学研究部部長をやらせていただいています山内と申します。

高二で最後ということで何か今までやったことなくて、それなりにやりがいのあることを…ということで僕はコンパイラ制作に挑戦。

感想はというと、やっぱり作っていて楽しかったですね。
自分の作ったものが動くっていうのは気持ちがいいものです。
後はものすごく疲れました。いや、それはもうものすごく。
その分完成時の気分もいいものですが。
うん、本当に動いてよかった。

えぇと、実は現在入稿ギリギリ状態です。ゲームのほうもギャー
とりあえずこのパンフレットが 11 月 3 日当日皆さんの元へ無事届いている事を願って。

[職員室] λ ..... さて行くか

2007-10-31 数学研究部室にて
大阪星光学院 数学研究部部長 山内 健二
及び数学研究部員一同

**読んでいただきありがとうございました。**