平成 30 年度　卒業論文

プログラミング演習における
ブロックインタフェースの長期適用

平成 31 年 2 月 14 日

15111041

前田　斗志

指導教員 三浦 元喜 准教授

九州工業大学 工学部 総合システム工学科

概 要

ブロックベースのプログラミング環境がプログラミング初学者にとって良い影響を与えることは既に知られている．我々はブロックインタフェースを長期的に提供することで学習者にどのような影響を与えるかを調査した．本研究では，プログラミング言語 Fortran の学習とそれを用いての数値解析を学習する大学 2 年生を対象としたプログラミング講義の中で実験を行った．講義中の演習のプログラミング環境にブロックインタフェースを使用できるようにした．ブロックインタフェースを長期適用することでブロックエディタを必要とする学習者を特定した．

目 次

第1章　序論

本論文は，初学者におけるテキストベースとブロックベースのプログラミング環境について論ずるものである．

本章では本テーマを取り巻く背景，プログラミング環境，そして本研究の目的を説明する．

1.1　背景

2020 年度から，すべての小学校においてプログラミング教育が必修化される．文部科学省・総務省・経済産業省が連携して立ち上げたプログラミング教育を推進する官民協働の「未来の学びコンソーシアム」が公開したパンフレット [1] によると，この背景として「IT 力」が我が国の競争力を左右するとしている．ヨーロッパでは,「IT 力」が，若者が労働市場に入るために必要不可欠な要素であると認識されていて，この「IT 力」を子供の頃から育成しておかなければ今後，国際社会で勝ち抜けないとも言われている．

また同パンフレットによると，プログラミング教育で大切にすべき視点として，苦手意識を感じさせないことと楽しく学べる工夫が必要とされている．これらは教員の工夫だけでは実現は難しく，プログラミング環境の工夫が必要と考えられる．そのため，プログラミングをテキストで記述せず，視覚的に行えるビジュアルプログラミング環境が多数開発されている．ビジュアルプログラミングは学習者の負荷が少なく初学者に良い影響を与えるとされているが，デメリットも存在する．例えば，ビジュアルプログラミングを代表する Scratch[2] のようなブロックベースのプログラミング環境では，テキストでは容易に編集できる内容もブロックを外してつけ直すような余計なステップを踏まなければならない．

1.2　ブロックベースのプログラミング学習環境

初学者のプログラミングを学習するためのあらゆる負担を少なくするため，多くのブロックベースのプログラミング学習環境が開発されている．ここではその一部を紹介する．

1.2.1　Scratch

Scratch[2] は MIT メディアラボが開発したブロックベースのプログラミング学習環境である．8 歳から 16 歳向けに設計されており，150 以上の国と地域で利用され，40 以上の言語に対応している．

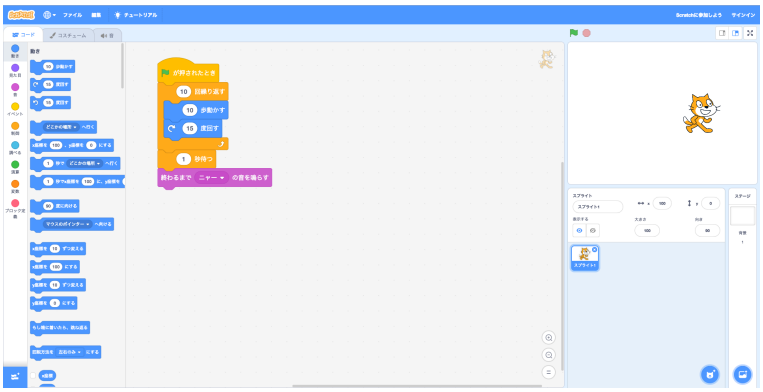

図 1.1: Scratch の外観

1.2.2　Blockly

Blockly[3] は Google が開発したブロックベースのプログラミングエディタを構築するための JavaScript ライブラリである．Blockly 自体は学習環境ではなく．学習環境を構築するためのライブラリで，Code.org や App Inventor など多くのアプリケーションが Blockly を基に設計されている．本研究において，我々が学習者に提供したブロックインタフェースも Blockly を基に設計されている．

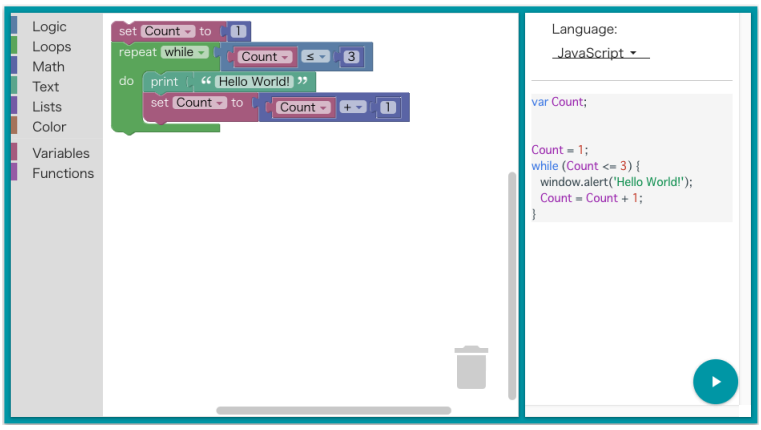

図 1.2: Blockly のサンプルの外観

1.2.3　プログラミン

プログラミン [4] は文部科学省が開発したブロックベースのプログラミング学習環境である．子どもたちに創ることの楽しさと，方法論を提供することを目的としている．設計にあたって前述の Scratch を参考にしている．

図 1.3: Blockly のサンプルの外観

1.3 本研究の目的

本論文の目的は，プログラミング学習者に対してブロックベースのプログラミング環境を長期的に提供することで学習者に与える影響を調査することである．大学のプログラミングの講義の中で，一般的なテキストベースのインタフェースとブロックベースのインタフェースを用意し，それぞれのエディタの編集ログを記録した．また講義を受講した学生にアンケート調査を実施し，学生にそれぞれのインタフェースの比較と評価を行ってもらった．これらのデータと学生の成績を比較し，それぞれのエディタの評価を行う．

第2章　関連研究

本論文ではこれ以降，Scratch や Blockly のようにブロックで配置することでプログラムを作成することができるプログラミング環境のインタフェースをブロックインタフェースと呼ぶ。本章ではブロックの関連研究を紹介する．

2.1　Scratckly：プログラミング初学者のためのブロック型学習法

三重野らは，理科系の大学生を対象に，ブロック型プログラミング学習システム「Scratckly」の実験・評価を行った [5]．初学者に演習型の授業において異なるインタフェースを使用してもらい，プログラミングの導入部分の評価を行った．演習の結果や採取したデータ，アンケートからブロック型のインタフェースはテキスト型インタフェースよりも初学者にとって利用した印象が良いということがわかった．

2.2　ブロックとタイプフェイスの間：ハイブリッドプログラミング環境の設計と評価

Weintrop らは，高校生を対象に，ブロックとテキストのハイブリッドプログラミング環境「Pencil.cc」は，プログラミングやコンピュータサイエンスの学習者の態度や理解に関して，同型のブロックベースおよびテキストベースの環境と比べてどのように機能するかを実験した [6]．ハイブリットプログラミング環境はブロック・テキストに対して明確に優位な点は得られなかった．

2.3　学習できるプログラミング：ブロックとその先

Bau らは，想起に対する認識と認知負荷の低減，エラーの防止によって，ブロック環境が初学者の学習能力をどのように改善するかを調査した [7]．経験豊富な開発者のための技術的なツールではなく，初学者を迎えるユーザインタフェースとしてのプログラミング言語を提供する場合，プログラミング環境で何をすべきかについての新しい概念を明らかにした．

2.4　関連研究まとめ

この章ではブロックインタフェースに関連する研究の紹介を行った．ブロックインタフェースがプログラミング初学者に良い影響を与えることは既に知られている．その中で．より良い影響を与える，もしくはその後のテキストでの一般的なプログラミングへの移行支援のための研究が行われている．しかしどの研究もブロックインタフェースを使用するのは，1 回限りの演習や講義前半の 5 回だけであった．そこで我々はセメスターの講義の中で長期的にブロックインタフェースを提供することで学習者にどのような影響が与えるかを調査する．

第3章　プログラミング環境

本章では，実験で使用したプログラミング環境とそれらの編集ログの記録について説明する．

3.1　プログラミング環境

我々はブロックエディタとテキストエディタの 2 つのプログラミング環境を用意した．どちらも Web ブラウザ上のみで利用できる Web IDE である．演習課題をどちらで行うかは学生が自由に選択することができる．

3.2　テキストエディタ

3.2.1　インタフェース

テキストエディタの外観を図 3.1 に示す．文字サイズの変更や自動インデントのような一般的な機能のボタンがエディタ上部にある．左側には行番号を表示している．講義で学習するプログラミング言語の「Fortran」は一般的に行頭に 6 文字分のスペースを空けるため，スペースを 3 文字を水色にハイライトして 6 文字のスペースを視覚的にわかりやすくしている．「ブロックエディタを起動する」ボタンを押すことで後述するブロックエディタを起動することが出来る．

3.2.2　自動補完機能

テキストエディタの最も重要な機能の一つである自動補完機能について説明する．自動補完機能とは，オートコンプリートとも称され，キー入力の途中で，補完候補を一覧表示したり，それを選択することによって，キー入力操作を省力化する機能である．こ

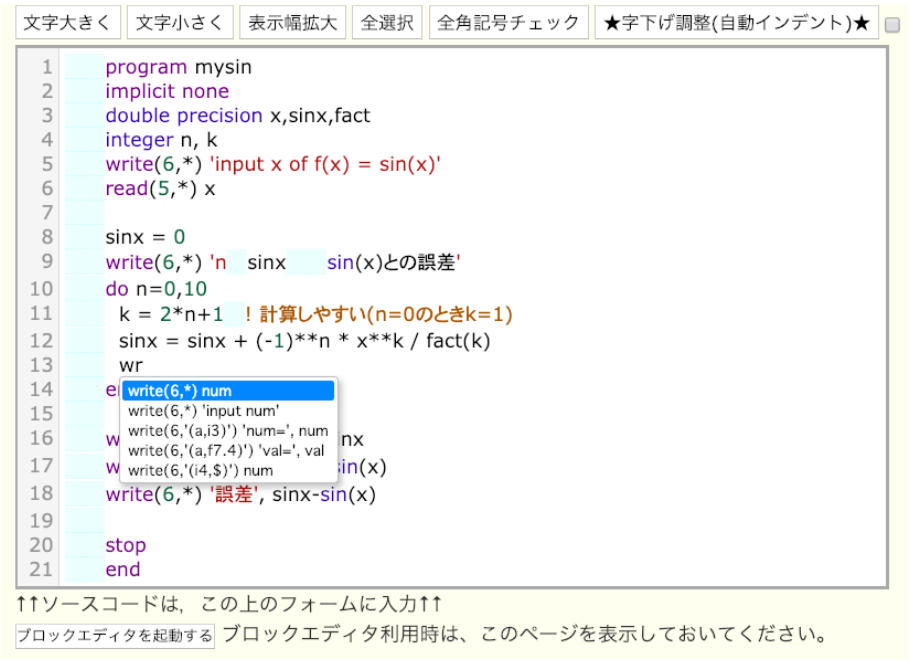

図 3.1: テキストエディタのインタフェース

の機能の使用方法としては，[Tab] キーを押下することで図 3.1 のようなの候補が表示され，[↓] キーで入力したい候補を選択して [Enter] キーで決定して入力することができる．

3.3 ブロックエディタ

3.3.1 ブロックエディタのインタフェース

ブロックエディタのインタフェースを図 3.2 に示す．画面左側の灰色のエリアをクリックすると，図 3.2 のようなブロックパレットと呼ばれる該当するブロックの一覧が表示される．学習者がブロックを選択し，ドラッグすると，プログラム編集領域に配置することができる．ブロックの配置を行うと，画面右のソースコードが自動的に更新される．エディタ上部の「演習 Web 上エディタに上書き転送＆保存」ボタンを押すことでブロックエディタで作成したプログラムを先ほどのテキストエディタに転送することができる．

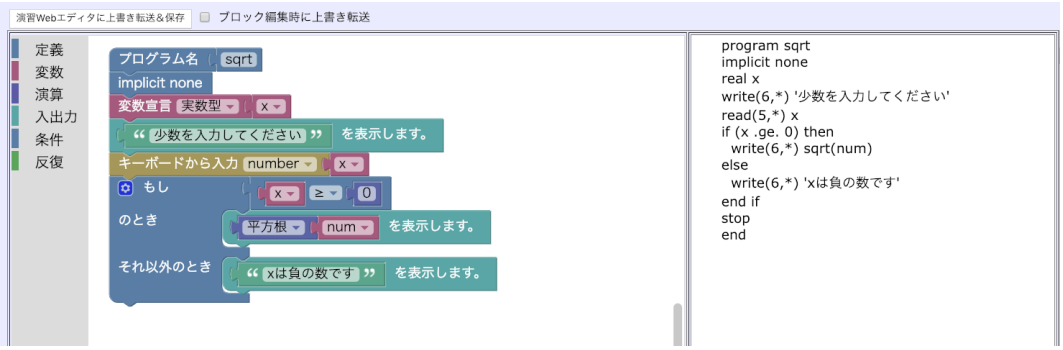

図 3.2: ブロックエディタのインタフェース

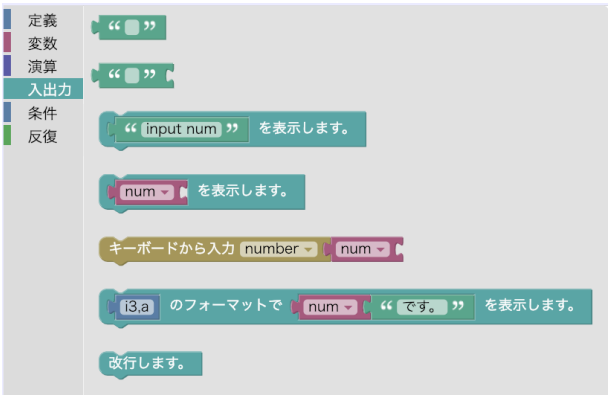

図 3.3: ブロックエディタのパレット

3.3.2 ブロックエディタの設計

Google Blockly を基に開発されている．Google Blockly は Javascript や Python や PHP などのソースコードを出力する機能を備えているが，Fortran には対応していない．我々が利用するブロックエディタは Python の出力として表示されるソースコードを Fortran のソースコードに変更することで Fortran に対応させている．Fortran のみに存在する関数もあるため，それぞれブロックを追加することで対応している．

3.3.3　ブロックエディタの改善

我々は今回以下の 2 点についてブロックエディタの改善を行った．1 点は新規のブロックを追加した．演習の中で使用する関数がすべて追加されていなかったため，図 3.4 のように適宜必要だと思われる関数のブロックを JavaScript を使用してブロックを作成し追加した．もう 1 点は表示されるソースコードの修正を行った．インデントが統一されていなかった点や出力されるべきソースコードが Python のソースコードになっていた点を修正した．

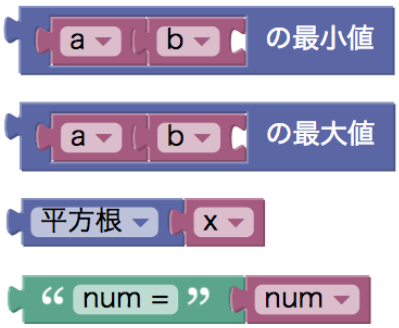

図 3.4: 新規に追加したブロックの例

3.3.4　ブロックエディタの編集ログ

ブロックエディタでイベントが発生した，つまり学生が何らかの操作をした際に，そのログを JSON 形式でサーバに送信して保存した．そのデータのテーブルのカラムを表 3.1 に示す．uid と assid，lecid，repid の 4 つの ID は学生や演習を識別するための ID である．json は JSON 形式のブロックの操作の情報である．src，blk は操作時点のソースコードとブロック情報である．dt は操作したときの日時である．

表 3.1: ブロックエディタの編集ログのテーブルのカラム

uid	assid	lecid	repid	json	src	blk	dt
学生 ID	演習 ID	講義 ID	レポート ID	イベント情報	ソースコード	ブロック情報	日時

表 3.1 のテーブル内に JSON 形式で保存したブロックの操作情報の構造を表 3.2 に示す．type ではイベントの種類，つまりブロックの操作の種類がわかるようになっており，create がブロックの作成，move がブロックの移動，deleate がブロックの消去のようになっている．workspaceId と blockId の 2 つの ID はワークスペースと操作したブロックを識別するための ID で，group は操作したブロックのグループの情報である．またブロックの操作は，すべてが一つの type で行われているわけではなく，例として，あるブロックを別のブロックの中に入れたとき，move，deleate，move という順に操作したことになる．

表 3.2: ブロックエディタの編集ログのテーブルのカラム

type	workspaceId	blockId	group
イベントの種類	ワークスペース ID	操作したブロックの ID	操作したブロックのグループ

第4章　実験

本章では実験環境，実験方法を説明する．

4.1　実験環境

本論文の実験は筆者所属大学・学部で 2018 年度後期の講義で実施された．実験環境の概要を表 4.1 に示す．この講義の受講生は前期でプログラミングの基礎について学習しているが，ほとんどの学生が Fortran の学習経験はない．また．この講義には 3 つのクラスがあり，合計で 264 名の学生が受講した．講義は 90 分で前半で教員による説明があり，後半は演習で自由に質問しながら行うことができる．

表 4.1: 実験の実施環境

対象学部・学年	工学部 2 年
対象講義	情報処理応用
クラス数	3 クラス（A クラス:90 名 B クラス:79 名 C クラス:95 名）
前提条件	前期に別の言語でプログラミングの基礎について学習
実験時間・形式	90 分・前半は教員による説明，後半は演習
指導体制	教員 1 名

講義のスケジュールを表 4.2 に示す．ブロックエディタは，第 2 週目に導入を行い，第 4 週目から操作ログの記録を開始した．後述するアンケートはおよそ第 8 週と第 12 週の時期に回答可能な状態にした．

4.2　実験方法

実験は，前述した講義の中ですべてのクラスに同様に行った．講義の前半は教員がその日に学習する内容についての説明を行う．後半は前半に学習した内容に関連する演習課題を各自で行う．演習課題は 1 回の講義につき 3 問程度出題され，演習中の教員に対

表 4.2: 講義のスケジュール

講義回	講義日		設定	講義内容
	A クラス	B, C クラス		
1	10 月 3 日	10 月 4 日		ガイダンス・基本文法
2	10 月 10 日	10 月 11 日	ブロックエディタの導入	条件分岐
3	10 月 17 日	10 月 18 日		繰り返し処理
4	10 月 24 日	10 月 25 日	編集ログ記録開始	制御構造の組み合わせ
5	10 月 31 日	11 月 1 日		ファイル処理
6	11 月 7 日	11 月 8 日		配列・行列
7	11 月 14 日	11 月 15 日		中間試験
8	12 月 5 日	11 月 29 日	プログラミングについてのアンケート	関数とサブルーチン
9	12 月 12 日	12 月 6 日		非線形方程式の解法
10	12 月 19 日	12 月 13 日		数値積分
11	1 月 9 日	12 月 20 日		常微分方程式の解法
12	1 月 16 日	1 月 10 日	エディタについてのアンケート	補完と回帰
13	1 月 23 日	1 月 17 日		連立一次方程式
14	1 月 30 日	1 月 24 日		練習問題
15	2 月 6 日	2 月 7 日		期末試験
16	2 月 13 日	2 月 14 日		まとめ

する質問や学生同士の相談は自由に行うことができる．また演習課題の提出期限は講義から一週間程度あり，提出期限内であればどの環境からでも自由に編集し提出することができる．後半の演習をブロックとテキストの Web エディタを使用してもらい，どちらのエディタの編集ログも記録した．

4.3 学生に対するアンケート

我々は学生に対して 2 つのアンケート調査を行った．それぞれのアンケートの内容はプログラミングについてと Web エディタについてである．プログラミングについてのアンケートは学生の経験と得意・不得意，プログラミングの講義の実態について調査するために行った．Web エディタについては学生に 2 つの Web エディタの比較と評価をしてもらうために行った．

4.3.1 プログラミングについてのアンケート

以下はプログラミングについてのアンケートの内容である．回答は質問 6 以外すべて 4 択で行ってもらった．

1. プログラミングは得意であるか
2. プログラミングの経験はあるか
3. 大学でのプログラミングの講義は理解できているか
4. タイピングや指示された作業についていけないことがあるか
5. コンパイルエラーを修正できないことはあるか
6. 今までのプログラミングの講義で，困ったことはあるか（自由記述）

4.3.2 Web エディタについてのアンケート

以下は Web エディタについてのアンケートの内容である．回答は全て 4 択で行ってもらった．

1. 演習や試験でテキストとブロックのどちらを使いたいか
2. 演習や試験でサンプルプログラムの一部を修正して回答するとき，テキストとブロックのどちらを使いたいか
3. ブロックはテキストよりも文法エラーを減らすのに役立つと思うか
4. ブロックはテキストよりも入力ミスを減らすのに役立つと思うか
5. ブロックはテキストよりもプログラムの構造を考えやすいと思うか
6. ブロックはテキストよりもプログラムの意味や動作を理解しやすいと思うか
7. ブロックはテキストよりもプログラムを完成させるまでの時間を短縮できると思うか

これらのアンケートは上記のスケジュールに示した時期にそれぞれ行われ，最低でも2週間の回答期間を設けた．それぞれのアンケートは合計で269件と258件の有効な回答を得た．実験の対象人数と異なるのは回答を行わなかった学生もいるためである．また，有効な回答の中には，全ての質問に回答しなかった学生も含まれる．

第5章　実験結果と考察

本章では前章で述べた実験の結果を示し，考察を行う．

5.1　プログラミングについてのアンケート

プログラミングについてのアンケートの結果を図5.1に示す．結果として，プログラミングを不得意と感じている人が全てのグループに80%近くいることが判明した．プログラミングの経験はおおよそ70%の人が経験がないと回答した．一方で大学のプログラミングの講義の内容を理解しているかという質問に対してはほぼ理解していると回答した人は5%から13%と極めて低い結果となった．タイピングやエラー回避も苦手である学生がすべてのクラスで70%以上を占めた．

プログラミングについてのアンケートの回答と成績の相関を表5.1に示す．まずクラスごとの比較をすると，BクラスとCクラスでは質問2以外の全ての質問で有意水準5%で有意な値が得られた．その中でもBクラスの質問1を除くと，全ての質問で有意水準1%で有意な差が認められた．Aクラスでは質問1,2のみで有意水準5%で有意な値が得られた．次に質問ごとに比較する．プログラミングの得意・不得意とプログラミングの経験を問うた質問1, 3で当然ながら全てのクラスで正の相関が得られた．また，全体的に質問4, 5でも大小はあるものの相関関係があることがわかった．しかし質問2のプログラミングの経験に関しては相関は見られなかった．これは前述の通り，おおよその学生がプログラミングの経験がないことが影響していることが考えられる．

またこれまでのプログラミングの講義で困ったことに関しての自由記述の代表的な回答を表5.2に示す.「基本が理解できていない」や「講義についていけない」などの回答が多く，プログラミングの講義に置いて，多くの学生が困っていることがわかった．実際に講義の中で，演習のプログラムをどのように書いてよいかがわからないといった学生も見られた．

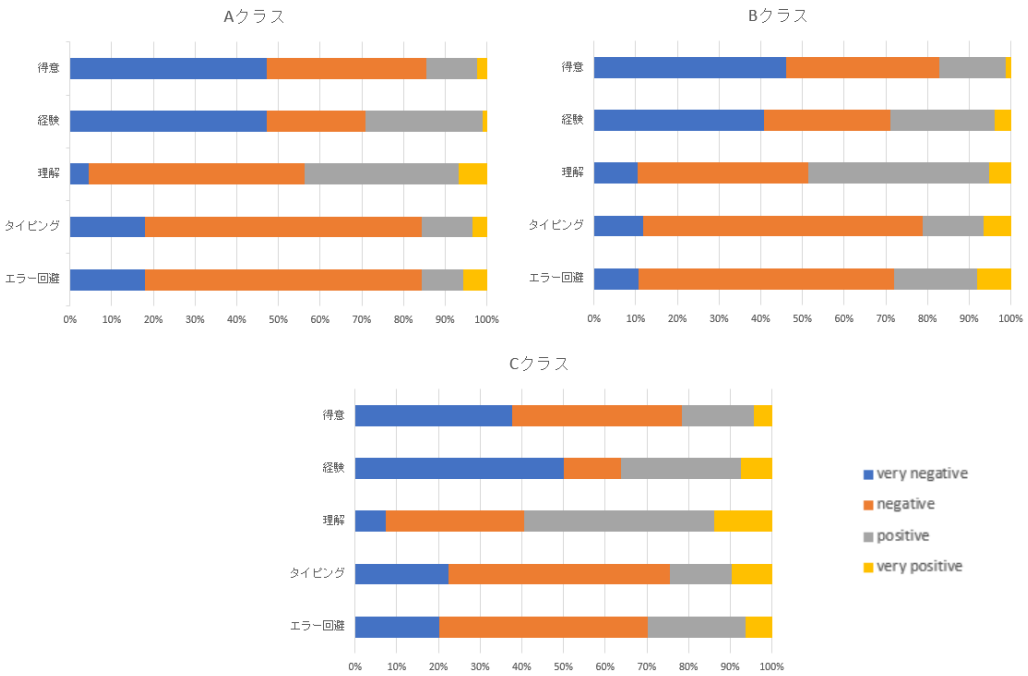

図 5.1: プログラミングについてのアンケートの結果

5.2 Web エディタについてのアンケート

Web エディタについてのアンケートの結果を図 5.2 に示す．このアンケートはブロックとテキストの 2 つの Web エディタの学生にとっての評価を比較するために行った．質問 1, 2 のテキストエディタとブロックエディタのどちらを使用したいかという問に対して，プログラムの新規作成とサンプル修正の両者において，テキストを使いたいという学生が 80%を上回った．すべてのクラスでサンプルプログラムの修正のほうがよりテキストを使いたいと答えており，やはりブロックエディタのデメリットとして小さな編集のしにくさが挙げられると考えられる．質問 3 から質問 7 までのブロックエディタとテキストエディタの比較に関しての回答はおおよそは近い分布を示したものの，それぞれのクラスで結果に差が生じた．

Web エディタについてのアンケートの回答と成績の相関を表 5.3 に示す．まず今回のアンケートの回答で有意差が認められる値が得られたのは A クラスの質問 1 と質問 6 のみだった．その上でそれぞれの質問について検討する．「ブロックを使用したい」,「ブロッ

表 5.1: プログラミングについてのアンケートの回答と成績の相関 (太字は有意差あり)

	A クラス			B クラス			C クラス		
	r	t 値	p 値	r	t 値	p 値	r	t 値	p 値
質問 1	**0.219**	t(88) = 2.103	0.038	**0.269**	t(79) = 2.483	0.015	**0.262**	t(96) = 2.658	0.009
質問 2	0.072	t(87) = 0.679	0.499	0.203	t(79) = 1.846	0.069	0.081	t(96) = 0.793	0.429
質問 3	0.251	t(87) = 2.422	0.017	**0.307**	t(79) = 2.868	0.005	**0.452**	t(96) = 4.959	0.000
質問 4	0.149	t(87) = 1.402	0.449	**0.404**	t(79) = 3.929	0.000	**0.427**	t(96) = 4.621	0.000
質問 5	0.148	t(87) = 1.393	0.835	**0.430**	t(78) = 4.201	0.000	**0.337**	t(96) = 3.504	0.001

表 5.2: 自由記述欄の代表的な回答例

	コメント
例 1	基本が理解できていないため，講義についていけない.
例 2	なぜエラーが出たのか，またそのエラーの原因が何なのかが分からない.
例 3	一度理解に躓いたらそれ以降で取り返しがつかない.
例 4	講義のスピードが早く，内容の理解がついていかない.
例 5	教員の説明，またその説明に出る専門用語が分からない.

クがテキストに比べて良い」という回答ほど回答としての値が大きくなるように設定した．学生が大きな相関があると言い切れるものはないが，全体的に見て「ブロックを使用したい」,「ブロックがテキストに比べて良い」と回答した学生は成績が低いという傾向があることがわかった．テキストでプログラムを容易に書くことができる学生ほど成績が高くなることは推測されるので，これは当然の結果と言える．しかしながら同時にブロックエディタが成績が低い学生の支援していることも明らかになった．ここでの結果のみでは成績が低い学生にどのように影響しているかはわからないものの，ブロックエディタはそのような学生にとって多く必要とされていた.

表 5.3: Web エディタについてのアンケートの回答と成績の相関 (太字は有意差あり)

	A クラス			B クラス			C クラス		
	r	t 値	p 値	r	t 値	p 値	r	t 値	p 値
質問 1	**-0.243**	t(83) = 2.24	0.025	-0.120	t(76) = 1.051	0.296	-0.029	t(92) = 0.275	0.784
質問 2	-0.166	t(83) = 1.533	0.129	-0.195	t(76) = 1.733	0.087	0.002	t(92) = 0.023	0.982
質問 3	-0.031	t(83) = 0.283	0.778	-0.018	t(76) = 0.162	0.871	0.062	t(92) = 0.602	0.549
質問 4	0.084	t(82) = 0.760	0.449	-0.117	t(76) = 1.018	0.312	0.0367	t(92) = 0.356	0.723
質問 5	-0.023	t(82) = 0.209	0.835	-0.080	t(76) = 0.698	0.487	-0.119	t(92) = 1.115	0.251
質問 6	**-0.222**	t(83) = 2.070	0.042	-0.159	t(76) = 1.406	0.164	-0.100	t(92) = 0.968	0.336
質問 7	-0.187	t(83) = 1.730	0.087	0.075	t(76) = 0.654	0.515	-0.018	t(91) = 0.167	0.867

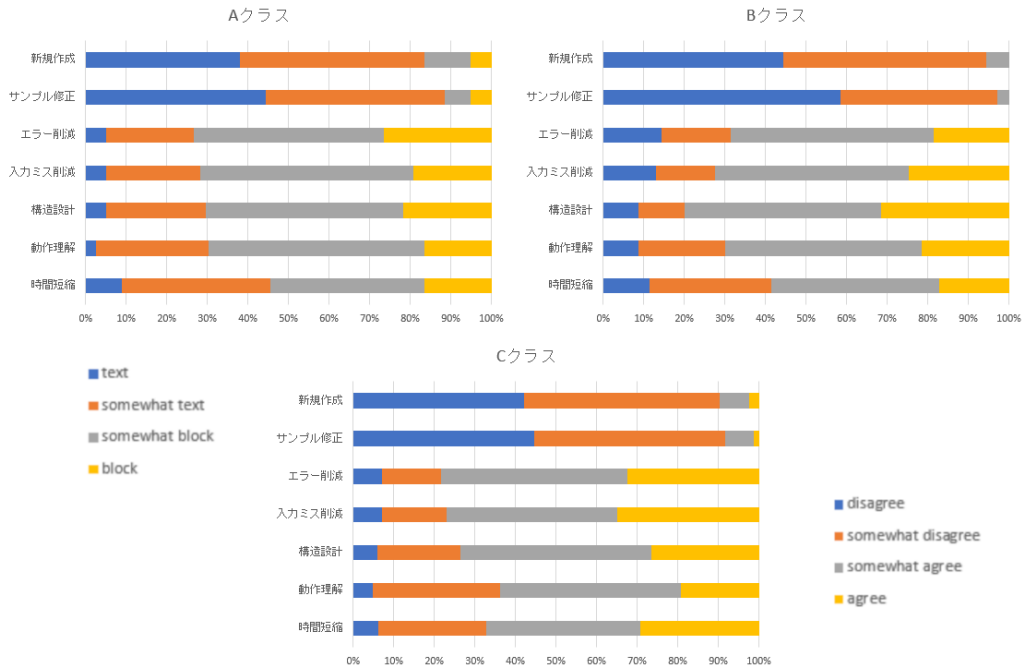

図 5.2: Web エディタについてのアンケートの結果

5.3 ブロックエディタ使用者の成績

ここでは，ブロックエディタを演習の中で使用していると見られる学生 63 名の成績を調査した．ログ記録後からブロックエディタを少なくとも 15 回操作した学生を対象とした．2 月 4 日時点で成績が 10 点未満の 3 名の学生はおおよそ講義に出席していないと判断して，この調査から除外した．成績と使用回数の相関を図 5.3 に示す．横軸が記録されたログの数で縦軸が成績成績は執筆の都合上，２月４日時点の成績を使用した．結果として，有意水準 1%で有意な値が得られたが相関係数は-0.046 となり，両者に相関がないことが判明した．($t(61) = 2.769$ $p = 0.007$)

5.4 タイピングスキルの影響

ここでは，ブロックエディタとテキストエディタが学生に良い影響を与えているかを検証するためにタイピングスキルと成績の相関を調べる．本来望ましいことではないが，

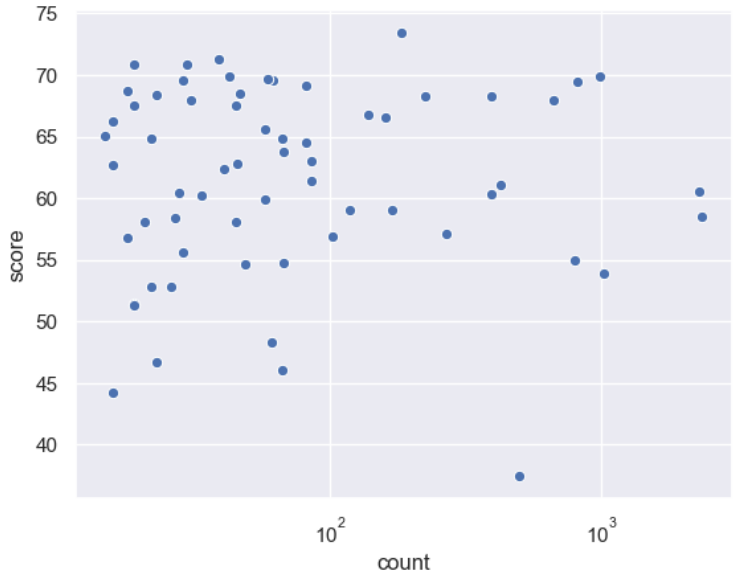

図 5.3: ブロックエディタと成績の相関

タイピングスキルとプログラミング講義の成績との相関が高くなってしまうことがある [8]．タイピングスキルと成績に正の相関が見られた場合，ブロックエディタや自動補完機能がタイピングスキルの低い学生の支援ができない，つまり影響が小さいと推測できる．ここでのタイピングスキルは，学生が自己申告したタイピングソフト yatt[9] のスコアを 1〜20 の数値で表したものである．

タイピングスキルと成績の相関を表 5.4 に示す．A クラスと C クラスで有意水準 5%で有意な値が得られ，２つの値を比較すると今回の実験ではタイピングスキルと成績の相関は低いということが明らかになった．これはブロックエディタや自動補完機能が少なからずタイピングスキルの低い学生に対しての支援ができている可能性が考えられる．

表 5.4: タイピングスキルと成績の相関

クラス	r	t 値	p 値
A	-0.261	t(78) = 2.391	0.019
B	0.255	t(32) = 1.495	0.145
C	0.268	t(67) = 0.227	0.0262

第6章　結論

本章では，これまでの結果をまとめ，今後の課題を述べる．

6.1　まとめ

本研究はプログラミング講義の演習を通じて，ブロックインタフェースの長期的な適用と効果の検証を行った．プログラミングについてのアンケートでは多くの学生がプログラミングの理解や講義の難易度についてネガティブな考えを持っていることがわかった．Web エディタについてのアンケートでは，有意差は認められなかったものの成績の低い学生ほどブロックエディタに高い評価を与えていることがわかった．長期的に記録したブロックエディタの編集ログでは影響を理解することができなかった．今回の実験でブロックインタフェースの影響を明らかにしたとは言い難いが，ブロックインタフェースを必要とする学習者を特定することができた．

6.2　今後の課題

今後の課題として，今回の実験での問題点が 2 点挙げられる．1 点は学習者がどのような操作をしたかが完全に把握できなかったことである．ブロックの動作ひとつひとつを知ることは可能だが，学習者の操作を把握するためには 3.3.4 で説明したように，複数の編集ログから操作を推測する必要があるため容易ではない．しかしこれを把握することで学習者のブロックの操作の影響を知ることができる可能性がある．もう 1 点は学習時期における学習者のブロックの操作の調査ができなかったことである．これには編集ログの記録が導入に比べて遅れたことと長期的な使用者の数が少なかったことが起因している．これらの課題を解決することで，ブロックインタフェースの影響についての更なる知見を得ることができる可能性がある。

謝辞

卒業論文を完成するにあたり，ご指導ご教授くださりました三浦准教授に御礼申し上げます．また，輪講や中間発表においてご指導やご教授を下さりました情報セクションの先生方に御礼申しあげます．加えて，本論文のデータ収集実験や評価実験において，被験者としてご参加頂きました九州工業大学の学生にお礼を述べたいと思います．最後に，私の意思を尊重してくださり大学進学を応援して頂き，経済面や生活面において，ご支援をして頂いた家族に心から感謝申し上げます．

参考文献

[1] 未来の学びコンソーシアム. 小学校プログラミング教育必修化に向けて. https://miraino-manabi.jp/assets/data/info/miraino-manabi_leaflet_2018.pdf, 2018.

[2] MIT メディアラボ. Scratch. https://scratch.mit.edu/, 2006.

[3] Google. Blockly. https://developers.google.com/blockly/, 2013.

[4] 文部科学省. プログラミン. http://www.mext.go.jp/programin/, 2017.

[5] 三重野剛. Scratckly:プログラミング初学者のためのブロック型学習法. https://ist.mns.kyutech.ac.jp/miura/papers/thesis2017-mieno.pdf, 2018.

[6] Uri Wilensky David Weintrop. Between a block and a typeface: Designing and evaluating hybrid programming environments. In *Proceedings of the 2017 Conference on Interaction Design and Children*, June 2017.

[7] Caitlin Kelleher Josh Sheldon Franklyn Turbak David Bau, Jeff Gray. Learnable programming: Blocks and beyond. In *Communications of the ACM*, June 2017.

[8] 三浦元喜. 初学者向け processing プログラミング環境におけるコード補完機能の導入と実践. 情報処理学会情報教育シンポジウム, August 2018.

[9] Hiroshi Kimura. Yatt: Yet another typing trainer. http://www.mext.go.jp/programin/, (2018年7月5日確認).