プログラミング教育のための目的文作成手順の提案と目的文作成支援環境の予備設計

酒井 大我　増原 英彦　叢 悠悠

関数の入出力を自然言語で説明する目的文は，プログラミングによる問題解決を学習する際に，特に再帰的な思考が必要な問題を整理して考えるのに役立つ．しかし，これまでのプログラミング教育で目的文の書き方を明示的に教えることはほとんどなく，学習者も良い目的文を書けるようになっているとは言い難い．本研究では，目的文の書き方を教育する為に，良い目的文が満たすべき条件を考察し，目的文を作成する具体的な手順とその手順を支援するブロック型環境を提案する．提案の一部機能のみを持つ環境を作成し，それを用いた学生がどのような目的文を作成するかを予備的に調査した．その結果を基に目的文作成支援環境の設計を議論する．

1　はじめに

プログラミング教育分野では，学習効果を高める試みとして様々な取り組みが行われている．具体的には，教育用に言語を開発・改良する [1,8] ことや教育用にプログラミング環境・演習ソフトウェア・プログラム可視化ソフトウェアなどのツールを開発する [3,6] ことやコーディング・プログラム設計・プログラミングによる問題解決の学習方法を提案する [2,5] ことのような取り組みがある．

デザインレシピ [2] は，プログラミング初学者を対象としたプログラミングによる問題解決の一連のステップであり，Felleisen らの教科書 *How to Design Programs* の中で提案された．デザインレシピは，この 6 つのステップから成る．

1. データ定義，データ例の作成
2. 目的文の作文，入出力の型の決定
3. 入出力例の作成
4. テンプレートの作成
5. コーディング
6. テスト

我々は，デザインレシピにおける目的文の重要性に注目し，学習者がより良い目的文を作文でき，より効果的に利用できることを目標とする．目的文とは，関数の役割を自然言語で説明した文章である．目的文には，ステップ 2 で学習者によって作文され，ステップ 5 で学習者によって利用されるという 2 つの側面がある．

本稿では，目的文作文の側面について，良い目的文を導く手順とそれを実現する作文環境を提案する．手順は，(1) 単語選択，(2) 雛形を使った単語の組み合わせ，(3) 引数名の付加から成る．作文環境は，ブロック型デザインレシピ環境 Mio [7] を拡張したもので，ブロックを用いて各手順に沿って目的文を作文させる．

目的文の利用は本稿の対象外であるが，提案・実現する作文手順・環境は今後の目的文を利用する環境の実現する際にも有用であると見込まれる．学習者は，ステップ 4 で計算に必要な関数呼出を書き出し，ステップ 5 で各関数呼出の式の意味を目的文を用いて考える．本稿で提案する環境によって目的文が構文木

Proposal for procedure of writing purpose statements in programming education and Preliminary design of composition environment for purpose statement.

Taiga Sakai　Hidehiko Masuhara　Youyou Cong, 東京工業大学情報理工学院数理・計算科学系, Dept. of Mathematical and Computing Science, School of Computing, Tokyo Institute of Technology.

として得られる為，次のようなことを実現できる．一つ目に，目的文の機械的な変形によって，関数呼出の式の意味を説明する文章を合成できる．二つ目に，そのような文章を学習者に提示することで，ステップ 5 で行う考察を助ける．

本論文では，第 2 節で良い目的文が満たす性質と学習者によって作文される目的文の問題点について述べる．第 3 節では良い目的文の作文手順を述べ，第 4 節でその手順に沿って目的文を作文させるブロック型環境について述べる．そして，第 5 節では提案の一部機能を持つ環境を用いて学生が作文する目的文を調査するユーザー実験について述べる．第 6 節では実験結果を述べ，第 7 節で実験結果から得られた知見を用いて目的文作文環境の設計の改善について議論する．最後に，第 8 節を本論文のまとめとする．

2 良い目的文の定義・利用と学生が書く目的文の問題点

2.1 良い目的文

良い目的文は，以下の 4 つの性質を満たす目的文であると定義する．

性質 1 関数の入力を明示しており，それは可変の値である (定数ではない)
性質 2 関数の出力を明示している
性質 3 関数の入出力を，プログラミングで使う言葉 (データ型を指す整数などの言葉) ではなく，問題領域の言葉を使って書いている
性質 4 関数の入力を引数名と対応づけている

問題 wages 企業は労働記録の表を管理している．労働記録は従業員名，時給，週の労働時間の情報を含んでいる．このとき，各従業員の週給を計算する関数 `wages` を定義せよ．

問題 `wages` に対する良い目的文の例として,「労働記録の表 `table` から各従業員の週給を計算する」がある．

この目的文は，上で定義した 4 つの性質を満たす．

性質 1

関数の入力として,「労働記録の表」を明示している．また,「労働記録の表」は給与払いをする上で可変の値である．

性質 1 を満たさない目的文の例として,「労働記録の表 `table` と残業手当 `overtime` から各従業員の週給を計算する」が挙げられる．これは，関数入力「残業手当」が定数であるという点において問題である．

性質 2

関数の出力として,「各従業員の週給」を明示している．

性質 2 を満たさない目的文の例として,「労働記録の表 `table` から各従業員の週給を計算する関数を作成する」が挙げられる．これは，関数出力の解釈に「各従業員の週給」と「各従業員の週給を計算する関数」の 2 つがあるという点において問題である．

性質 3

「労働記録の表」や「各従業員の週給」という言葉は，給与払いの問題領域の言葉である．

性質 3 を満たさない目的文の例として,「労働記録型のリスト `table` から整数型のリストを計算する」が挙げられる．これは，関数入力「労働記録型のリスト」・関数出力「整数型のリスト」がプログラミングで使う言葉であるという点において問題である．

性質 4

関数の入力「労働記録の表」は，引数名 `table` と対応づいている．

性質 4 を満たさない目的文の例として,「労働記録の表から各従業員の週給を計算する」が挙げられる．これは，関数入力「労働記録の表」が引数名と対応していないという点において問題である．

2.2 目的文の利用

How to Design Programs は，学習者が関数定義 (デザインレシピのステップ 5) に目的文を利用することを提案している [2]．特に，目的文の利用により関数呼出の式の意味を学習者が考察できると述べている．以下は，*How to Design Programs* で挙げられて

いる目的文の利用例である．具体的に，文字列リストに含まれる文字列の数を求める問題を考える．

デザインレシピのステップ 4 が完了すると，エディタは以下のような状態になる．

```
; List-of-strings -> Number
; 文字列リスト alos に含まれる文字列の数を
; 決定する †1
(define (how-many alos)
 (cond
  [(empty? alos)...]
   [else
   (...(first alos)...
    ...(how-many (rest alos))...)]))
```

学習者は，ステップ 5 でこのテンプレートを用いて関数を定義する．

まず初めに，学習者はテンプレートに書かれている各式の意味を考察する．このうち，関数呼出の式の意味は，関数入力の箇所を実際の引数に置き換えた目的文から得られる．以下は，how-many 関数のテンプレートに書かれている各式の意味である．

(first alos)

入力の先頭要素

(rest alos)

入力から先頭要素を取り除いたリスト

(how-many (rest alos))

入力から先頭要素を取り除いたリストに含まれる文字列の数

そして，この考察結果を基に関数を定義する．how-many 関数が返す値は「alos に含まれる文字列の数」であるので,「入力から先頭要素を取り除いたリストに含まれる文字列の数」に 1 を加えた式を作ることで，関数定義を以下のように完成できる．

```
; List-of-strings -> Number
; 文字列リスト alos に含まれる文字列の数を
; 決定する
(define (how-many alos)
 (cond
  [(empty? alos) 0]
   [else
    (+ (how-many (rest alos)) 1)]))
```

†1 原著は英語である為，著者が日本語に訳した．

2.3 学生が書く目的文の問題点

一部の学習者にとって，良い目的文を書くことは容易ではない．実際に，2022 年度の計算機科学概論 †2 で収集した学生の目的文の中には，良い性質を満たさないものが見られた．計算機科学概論は，デザインレシピに基づいた関数型プログラミングを学ぶ授業であり，目的文を用いたプログラミングを学習する．具体的に，税抜き価格からイートインの値段を求める関数の目的文に以下のようなものが見られた．

例 1: イートインの値段を返す関数を作成する

目的文からは，関数の出力情報が「イートインの値段」か「イートインの値段を返す関数」のどちらであるかが明確ではない．また，関数の入力情報も含まない．よって，性質 1，2 を満たさない．この目的文は，eatin の関数呼出の式を「イートインの値段」,「イートインの値段を返す関数」の 2 通りの意味で考察できるという点で問題がある．

例 2: 整数を 1.1 倍して返す

「整数」は問題領域の言葉ではなく，性質 1 を満たさない．この目的文は，eatin の関数呼出を考える際に，関数の役割に適切した引数が分からないという点で問題がある．

良い目的文は，学習者による関数定義にとって重要であるにも関わらず，良い目的文の書き方やその教育法は確立されていない．以降の節では，良い目的文の書き方とそれを実現する為の環境を提案する．

3 学習者による良い目的文の作文手順

3.1 手順の提案

本研究では，学習者による良い目的文の作文手順を提案する．これは，学習者がプログラミング演習に取り組む際に用いるものである．提案手順は，次の 3

†2 東京工業大学情報理工学院数理・計算科学系で開講されている．

つの手順によって構成される．

手順 1：単語選択

問題解決に用いる問題領域の言葉を，問題文から単語として選択する．

手順 2：選択した単語と雛形の組み合わせ

選択した単語を目的文雛形の空欄に組み合わせる．

手順 3：関数入力と引数名の対応づけ

目的文の関数入力部分と，シグネチャで定義される引数名を目的文中で対応づける．

以下は，プログラミング問題 wages に対する良い目的文をペンと紙を用いて作成する具体的な手順である．

手順 1：単語選択

給与払いという問題解決に用いる言葉を，単語として問題文から選択する (図 1-(1))．

手順 2：選択した単語と雛形の組み合わせ

穴あきの文章群から，目的文の雛形として使う文章を選ぶ．今回は，「(入力) から (出力) を計算する」を目的文の雛形に用いる．選択した単語から入力として「労働記録の表」，出力として「各従業員の週給」を選ぶ (図 1-(2))．このとき，入力には可変の値であるものを選ぶ．

手順 3：関数入力と引数名の対応づけ

入力「労働記録の表」と引数名 `table` を対応づけることにより，目的文を完成させる (図 1-(3))．

3.2 良い目的文の性質を満たすことの確認

提案手順を用いて作文された目的文は，良い目的文の 4 つの性質を満たす．

- 手順 1 で問題文中から単語を選択することにより，性質 3(問題領域の言葉を使って書く) が満たされる．
- 手順 2 で目的文の雛型を用いることにより，性質 1(関数の入力を明示しており，それは可変の値である) と性質 2(関数の出力を明示している) が満たされる．
- 手順 3 で関数入力に引数名を対応づけることにより，性質 4(関数入力を引数名と対応づけている) が満たされる．

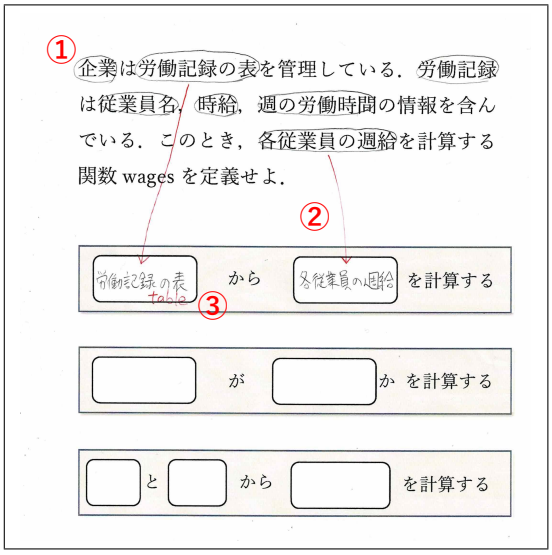

図 1　問題 wages に対する目的文作成手順

4 手順に沿った目的文作文のためのブロック型環境

4.1 環境の設計

本研究では，第 3 節の提案手順に沿って目的文を作文させるブロック型環境 (図 2) を設計した．

学習者は，第 3 節の提案手順に対応した 3 つの機能を順に用いることで目的文を作文できる．目的文「労働記録の表 `table` から各従業員の週給を計算する」を例に，本環境の機能について説明する．

機能 1：単語抽出

本環境は，学習者が選択した文字を基に単語を抽出する (図 2-1)．学習者は，必要な単語を選択する為に，問題文中から始点・終点の文字を選択する．例えば，単語「労働記録の表」を選択するには，始点「労」と終点「表」を選択する．

機能 2：単語と目的文雛形の組み合わせ

本環境は，目的文雛形ブロックに加えて，学習者の単語選択に応じた単語ブロックを提供する (図 2-2)．学習者は，これらのブロックを組み合わせることで作文する (図 2-3)．

機能 3：関数の入力情報と引数名の対応づけ

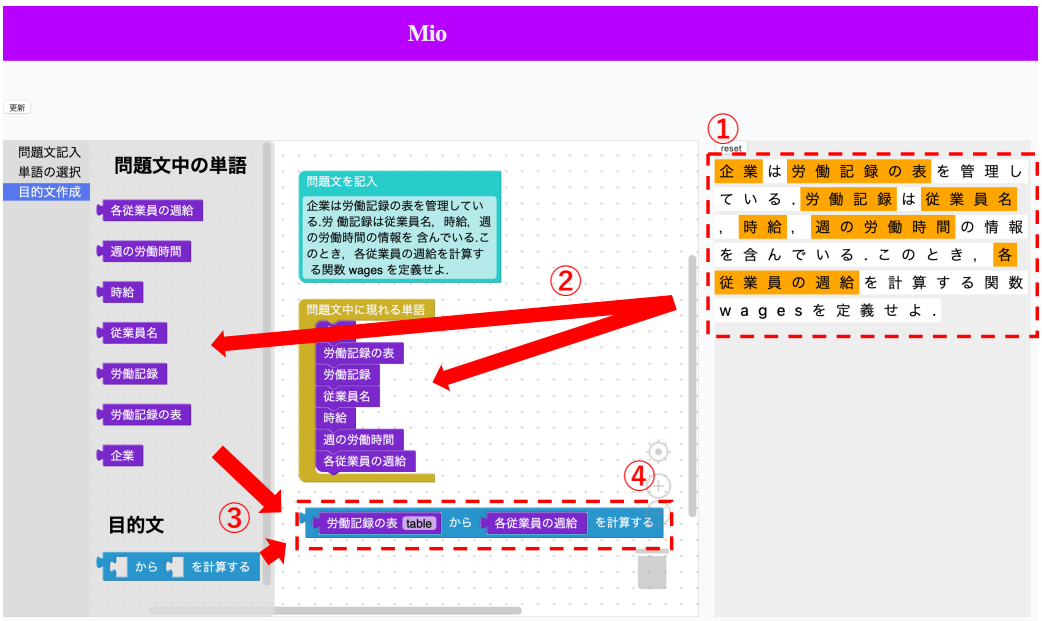

図 2　手順に沿った目的文作文のためのブロック型環境の概観

本環境は，関数入力に対応する単語ブロックにテキストボックスを追加する (図 2-4)．このとき，関数入力に対応する単語ブロックは，目的文雛形ブロックの空欄の内，関数入力に対応する空欄に組み合わされたブロックである．学習者は，シグネチャで定義した引数名 `table` をテキストボックスに記入することで，目的文を完成させる．

本環境では，学習者は各機能間の遷移を自由に行える．例えば，機能 2 を用いて作文している途中で，新たな単語を抽出する為に機能 1 に戻ることができる．

5　ユーザー実験

第 4 節で示したブロック型環境の設計について議論する為に，実際に学習者にブロック型環境を用いて目的文を作文してもらい，その目的文を調査する実験を行った．

実験の対象者は，2023 年度の「計算機科学概論」の履修生 30 名であり，そのほとんどが情報理工学院数理・計算科学系の学部 2 年生である。計算機科学概論では毎回のプログラミング課題で目的文を記述することが要求される．

対象者は，以下に示す 2 つの問題を与えられる．それぞれの問題に対して，第 4 節の設計を基に実装したブロック型環境を用いて目的文を作文する．問題 checkout は，Fisler らの研究 [4] で用いられた演習問題 Shopping Cart を参考にしたものである †3．

問題 area　受け取った図形の面積を求める関数 area を定義せよ．受け取る図形の例としては，半径が 2 の円や幅が 6，高さが 5 の長方形を考える．なお，円周率は 3.14 とする．

問題 checkout　あるオンライン衣料品店では，支払い時に割引が適用される．ショッピングカートは，購入する商品のリストになっている．各商品は，名前（”shoes”などの文字列），販売価格 (整数) の情報を持つ．次の 2 つの割引を適用した後のカートの総額を計算する関数 checkout を定義せよ．カートに少なくとも総額 100 ドル以上の靴が含まれている場合，すべての靴の価格を 20%割引する（商品の名前が ”shoes” と完全に一致するもののみ対象）．カートに少なくとも 2 つの帽子が含まれている場合，カートの

†3　ユーザー実験の為，演習問題を翻訳・一部改変した

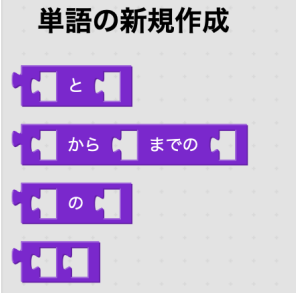

図 3　新規単語の作成の為のブロック

図 4　新規作成された単語ブロック

合計から 10 ドル割り引く (商品の名前が ”hat” と完全に一致するもののみ対象) .

なお，実験に用いたブロック型環境と第 4 節の設計の間には以下のような違いがある．

1. 機能 3 が実装されていない

機能 3 で追加するテキストボックスには，シグネチャで定義した引数名を記入する．本実験ではシグネチャを記述させないため，機能 3 を実装しなかった．

2. 新規単語の作成機能の追加

実験対象者が，単語を細かく分割するような単語選択をしたとしても，目的文を作文できるように追加のブロック (図 3) を用意した．例えば,「各従業員の週給」を「各従業員」と「週給」の 2 単語に分割する単語選択をしても,「... の...」というブロックを組み合わせることで図 4 のような単語を自然に作成できる．

3. 利用できる目的文雛型の制限

今回の実験では，目的文の雛形を「(入力) から (出力) を計算する」という単純な入出力関係を定義する文のみに限定した．

6　実験結果

実験対象者が作文した目的文を，目的文中に含まれる関数の入出力の種類に関して分類した (表 1，2)．なお，問題 checkout では関数の誤った役割を説明している目的文が複数見つかった為，それらは入出力を分類しなかった．

具体的な分類例は以下である．

例 1

図 5 - (a) の目的文は，入力「受け取る図形」と出力「図形の面積」を持つ.「受け取る図形」は図形の概念を表す単語,「図形の面積」は面積の概念を表す単語である．従って，この目的文は入力に図形系の単語，出力に面積系の単語を持つ目的文として分類する．

例 2

図 5 - (b) の目的文は，入力「2 つの割引と購入する商品のリスト」と出力「カートの総額」を持つ．この時，入力は接続ブロック「と」で接続されているので,「2 つの割引」と「購入する商品のリスト」の 2 つが入力となる.「2 つの割引」は割引の概念を表す単語,「購入する商品のリスト」は商品リストの概念を表す単語,「カートの総額」はカートの概念を表す単語である．従って，この目的文は入力に割引系の単語と商品リスト系の単語，出力にカート系の単語を持つ目的文として分類する．

関数の誤った役割を説明している目的文としては，以下のような例があった．

- 商品から割引の総額を計算する
- 購入する商品の名前と販売価格から割引を適用した後のカートの総額を計算する
- ショッピングカートの商品の総額から支払い時の総額を計算する
- カートに少なくとも総額 100 ドル以上の靴が含まれている場合の靴の価格の 20%割引する割引とカートに少なくとも 2 つの帽子が含まれている場合の 10 ドル割り引く割引からカートの総額を計算する
- 商品の総額から割引の総額を計算する
- 商品のリストから商品の販売価格 (整数) の総額を計算する
- リストから商品の総額を計算する

(a) 問題 area

(b) 問題 checkout

図 5　実験環境上で学生が作文した目的文

表 1　問題文 area の目的文の分類

入力	出力	個数
[図形系の単語] 受け取った図形，受け取る図形，図形	**[面積系の単語]** 面積，図形の面積，受け取った図形の面積，円の面積	22
[図形のパラメータ系の単語] 円の半径と長方形の幅と高さ，円の半径， 図形の長方形の幅 高さ 円の半径， 長方形の幅と高さ 円の半径		6
半径 2 の円		1
円と長方形		1

表 2　問題文 checkout の目的文の分類

入力	出力	個数
[カート系の単語] ショッピングカート，カートの情報	**[総額系の単語]** カートの総額，総額，割引後の総額，割引を適用した後のカートの総額，次の二つの割引を適用した後のカートの総額	10
[商品リスト系の単語] 商品のリスト，購入する商品のリスト カートの商品のリスト		8
[商品リスト系の単語]　と　**[割引系の単語]** 2 つの割引，帽子と靴の割引，割引情報		4
[商品リストのパラメータ系の単語]　と　**[割引系の単語]** 購入する商品のリストの販売価格 (整数)，リストの商品の販売価格		2
関数の誤った役割を説明している目的文		8

- ショッピングカートから購入する商品の総額を計算する

7 考察

第 6 節のユーザー実験から得られる知見を基に，今後の展望として提案環境に追加するべき機能について考察する．

入出力例を記述する機能の追加

問題 checkout では，関数の役割に一致しない入出力を記述した目的文が見られた．これは，問題解決に関係する値が問題 area と比べて増えたことで，学習者にとって単語の中から入出力関係を見つけることが難しくなったことに起因すると考えられる．その為，入出力関係を入出力例などを用いて明示的に示す機能を併用し，学習者による入出力関係の発見を支援する必要があると考えられる．

定数値の単語を区別する機能の追加

問題 checkout では，割引を関数入力の一つとする目的文が見られた．このとき，割引は商品の精算をする上で定数である為，このような目的文は良い目的文ではない．これは，目的文雛型ブロックに単語ブロックを組み合わせる際に，どのような考えに基づいて使用する単語ブロックを選ぶべきかが学習者に伝わらなかったことに起因すると考えられる．従って，単語選択機能に定数値の単語をマーキングなどを用いて区別する機能を追加し，適切な単語ブロックの使用を支援する必要があると考えられる．

単語間の関係性を可視化する機能の追加

問題 area・問題 checkout の両方で，関数入力に「円の半径」等のパラメータを記述した目的文が見られた．このような目的文は，悪い目的文ではない．しかし，問題 area の目的文に対する観察の結果として，返り値の計算に必要な値をどのようなデータ型で関数に与えるのが良いのかを学習者が実験環境上で考察できていない可能性があった．従って，単語選択で得られた単語ブロックを組み合わせて単語間の関係性を可視化する機能を追加し，どのようなデータ型を定義できるのかを可視化させる必要があると考えられる．

8 まとめ

本研究では，学習者がより良い目的文を作文できるようにする為，良い目的文を作文する為の手順を提案した．まず初めに，良い目的文が満たす 4 つの性質を定義した．次に，良い目的文の作文手順として，具体的な 3 つの手順を提案した．そして，提案手順に沿った目的文作文を実現するブロック型環境を設計した．最後に，提案環境の一部機能を実装した環境を用いて予備実験をし，そこから得られた知見を基に必要な機能を考察した．

今後の展望としては，入出力例を記述する機能・定数の単語を区別する機能・単語間の関係性を可視化する機能の追加が挙げられる．その他の展望としては，提案環境で作成された目的文を関数定義に利用できる環境の実現がある．

参 考 文 献

[1] Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., and Miller, P.: Mini-languages: a way to learn programming principles, *Education and Information Technologies*, Vol. 2, No. 1(1997), pp. 65–83.

[2] Felleisen, M., Findler, R. B., Flatt, M., and Krishnamurthi, S.: *How to design programs: An introduction to computing and programming*, The MIT Press, 2001.

[3] Findler, R. B.: DrRacket: The Racket Programming Environment.

[4] Fisler, K., Krishnamurthi, S., and Siegmund, J.: Modernizing Plan-Composition Studies, *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, New York, NY, USA, Association for Computing Machinery, 2016, pp. 211–216.

[5] Foster, I.: *Designing and building parallel programs: concepts and tools for parallel software engineering*, Addison-Wesley Longman Publishing Co., Inc., 1995.

[6] Guo, P. J.: Online Python Tutor: Embeddable Web-Based Program Visualization for Cs Education, *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, New York, NY, USA, Association for Computing Machinery, 2013, pp. 579–584.

[7] Nose, J., Cong, Y., and Masuhara, H.: Mio: A Block-Based Environment for Program Design, *Pro-*

ceedings of the 2022 ACM SIGPLAN International Symposium on SPLASH-E, SPLASH-E 2022, New York, NY, USA, Association for Computing Machinery, 2022, pp. 62–69.

[8] Wirth, N.: The programming language pascal, *Acta Informatica*, Vol. 1, No. 1(1971), pp. 35–63.