K-014

視覚障がい者のためのプログラム読み上げシステムの検討
Study on Program Reading Out System for Visually Impaired

平川 亮‡　　中藤良久‡
Ryo Hirakawa　Yoshihisa Nakatoh

1. はじめに

現代社会の課題の 1 つとして、視覚障がい者の社会進出が挙げられる[1]。中でも、プログラムの開発業務をこなせる労働者への需要は高く、一方、視覚障がい者自身のプログラマとしての就業機会への期待も高まっている[2]。そこで、視覚障がい者に対してプログラミング支援を行うことで視覚障がい者の社会進出に対する支援を行う。

視覚障がい者のプログラミングにおいて、最大の課題はエディタ上での記述内容(プログラムのソースコード)の把握である。そこで、現状では、記述内容をスクリーンリーダで読み上げることで対応している。しかし、スクリーンリーダを用いてソースコードを読み上げる従来の方法では、ソースコード中のどこを読み上げているのかを直感的に理解するのが困難である。そこで本研究では、従来のスクリーンリーダによる読み上げに比べ、視覚障がい者が理解しやすい読み上げ対応フローチャートによるプログラムの読み上げ方式を提案する。

2. 提案システムの概要

本稿では、プログラムの読み上げ方式としてソースコードを読み上げ対応フローチャートに変換し、そのフローチャートを読み上げる方式を提案する。なお、今回提案する手法は、最も一般的なプログラミング言語といえる C 言語のプログラムコードの読み上げを対象としている。

2.1 読み上げ対応フローチャート

本システムでは、従来のフローチャートではなく、読み上げに対応した形式のフローチャートを発案し、プログラムの読み上げに用いる。読み上げ対応フローチャートは以下の 3 つのアプローチによりユーザの理解を支援している。

(1)条件分岐の開始、終了箇所のブロック化
(2)繰り返し処理の統合化
(3)コメントによる構造化の利用

(1)は、条件分岐構造を含むプログラムのフローチャート化におけるアプローチである。本システムでは図1の(a)のような条件分岐構造を含むプログラムのソースコードが一般的には図1の(b)のようなフローチャートで表現されるのに対し、図 1 の(c)のように else などの条件分岐や分岐の終了箇所をブロック化し、矢印で画像的に表現されていた部分を処理の 1 つとして読み上げるようなフローチャートを提案した。

(2)は、繰り返し構造を含むプログラムのソースコードをフローチャート化する際におけるアプローチである。本システムでは、図 2 の(a)のようなソースコードを図 2 の(b)の図のような従来のフローチャートに対し、図 2 の(c)のようなフローチャートに変換した。この図 2(c)のフローチャートでは、図中の loop のブロックのように、繰り返し構造の始まりから終わりまでを一つの処理のごとくブロックを統合して読み上げる。この手法は繰り返し処理の部分をよりイメージしやすくするためのものである。

(3)は、図3の(a)のように“/*”，“*/”によるコメントで段階構造化されているプログラムに対するアプローチである。コメントはプログラムコードの構造化を図ることを目的として使用されるため、構造把握の支援を１番の目的としていない従来のフローチャートには利用されない。しかし、読み上げ対応フローチャートは、処理構の造把握を１番の目的としているため、図3の(b)のようにコメントにより処理を統合化したフローチャートを作成する機能も追加する。なお、統合されたブロックは繰り返しに関するアプローチと同様の方法で読み上げることができる。

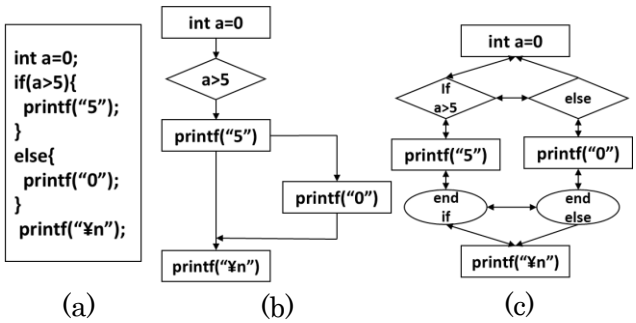

図 1 条件分岐構造のプログラム

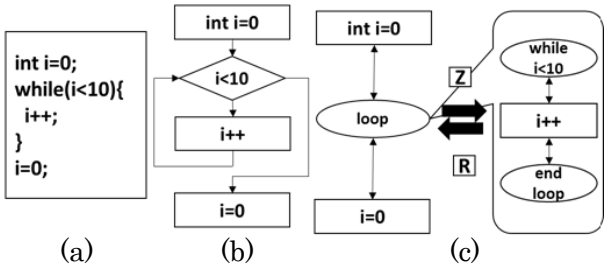

図 2 繰り返し構造のプログラム

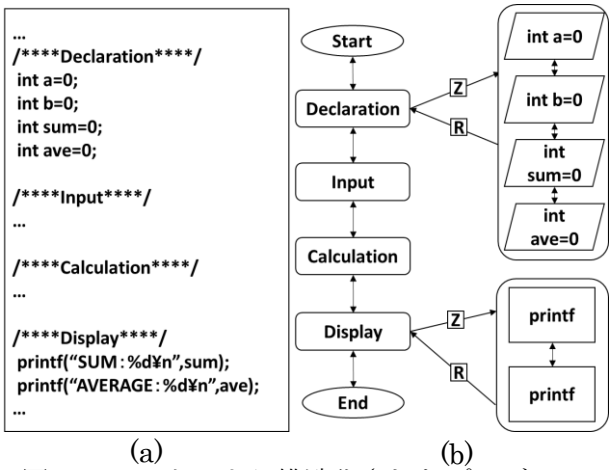

図 3 コメントにより構造化されたプログラム

‡ 九州工業大学　Kyusyu Institute of Technology

2.2 提案システムの読み上げ方式

従来法はソースコードを「行」単位で上下に移動しながら、各行の処理をスクリーンリーダでそのまま読み上げる方法である。一方、提案法は、プログラムを「処理のブロック」単位で上下左右に移動し、移動先のブロックの処理内容や条件式を読み上げる方法である。

提案法を実装した支援システムは、(1)ユーザのキーボードからの入力待機、(2)入力内容に従ったフローチャート内のブロックの移動、(3)移動先のブロックの処理・条件式の読み上げ、の 3 つを繰り返すといった手順で動作するようになっている。例えば、図 1 の(c)のようなフローチャートの場合、最初に“int a=0”と読み上げる。次にユーザの入力を待ち、キーボードの“↓キー”が入力されると、“if a>5”のブロックに移動し、“if a>5”と読み上げる。その後再び入力を待ち、“→キー”を入力されると、“else”と読み上げるというような動作になっている。また、図 2(c)や、図 3(b)のようにいくつかの処理を 1 つのブロックとして統合しているフローチャートでは、そのブロックを選択しているときに“z”を入力することで図中の吹き出し部の中のような統合箇所を詳細化したフローへ移動する。そして、“r”を入力すると元の統合されたブロックに戻るといったように動作する。

3. 評価実験

提案した読み上げシステムの有用性を検証するため、従来法と提案法によるプログラムの読み上げを聞きながら、プログラム全体の中で問題と思われる箇所へ移動してもらう形式の実験を行った。なお、この実験はプログラムの読み上げを聞くのはコード記述時とデバッグ時であることから、デバッグ時における有用性を検証する目的で行った実験である。

3.1 実験条件

実験では、表 1 のように 3 種類の構造パターンを持つプログラムを難易度 3 段階に分けて用意する。また、被験者は晴眼者の 10 名で、実験は PC 画面の状況が見えないように行い、実験の評価項目は操作開始から正しい処理の箇所へたどり着くまでに要した時間とする。

表 1 実験で読み上げたプログラムの分類

	Branch	Repeat	Mix
Level 1	Not nested structure (About 40steps)	Not nested structure (About 40steps)	Not nested structure (About 40steps)
Level 2	Nested structure (About 70steps)	Nested structure (About 70steps)	Nested structure (About 70steps)
Level 3	Nested structure +α (About 100steps)	Nested structure +α (About 100steps)	Nested structure +α (About 100steps)

3.2 実験結果

実験に使用したプログラムでの実験終了までに要した時間を被検者全体で平均化し、構造別にまとめたものと難易度別にまとめたものを図 5、図 6 に示す。図 5 より、提案手法により分岐構造では 60%、繰り返し構造では 40%、両者混合の構造では 45%程度、操作時間が減少するという結果が得られた。また、図 6 より、難易度 1、2、3 ではそれぞれ、40%、42%、53%ほどの操作時間の減少がみられた。この結果より、分岐構造を含むプログラムにおいて大きく改善され、また、プログラムが長く複雑になるほど、提案手法による改善度が大きくなることも確認することができた。

4. 結論

本研究では、提案法が従来法に比べ、操作時間を短縮できたことから、提案した読み上げ手法の有用性の確認できた。今後は、ソフトの操作性・読み上げ方式のさらなる改善やコード記述などの実際のソフトウェア開発作業においての支援を行い、よりよい開発環境の構築を進めていく。

参考文献

[1] 厚生労働省,「平成 27 年度・障碍者の職業紹介状況等」, 2016 年 5 月, http://www.mhlw.go.jp/file/04-Houdouhappyou-11704000-Shokugyouanteikyokukoureishougaikoyoutaisakubu-shougaishakoyoutaisakuka/0000084782_1.pdf

[2] 長岡英司,「重度視覚障碍者のソフトウェア開発技能の職業的有用性」, 職業リハビリテーション、Vol.16, pp44-51 (2003)

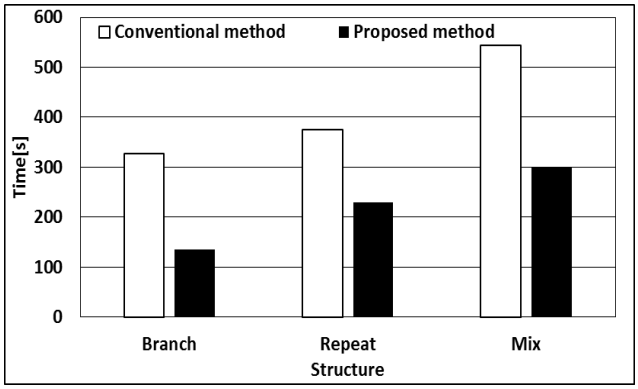

図 5 構造別実験結果

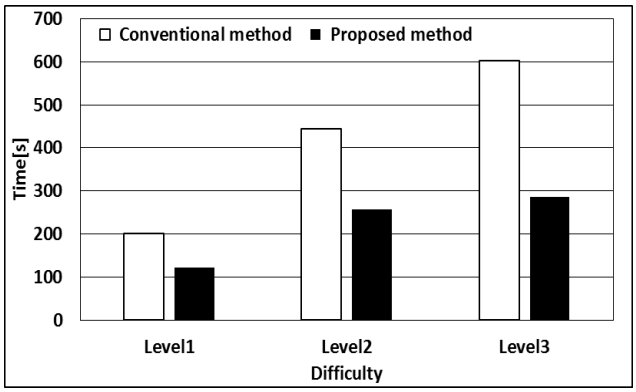

図 6 難易度別実験結果