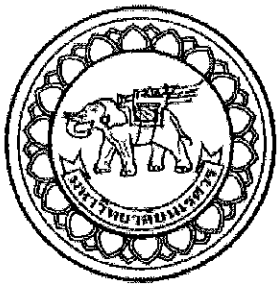


โปรแกรมตรวจสอบมาตรฐานการเขียนโปรแกรม ในภาษา C++

C++ Coding Standard Detector

นายวรชาติ พิรุณรักษ์ รหัสนิสิต 48364890

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2551

ใบรับรองโครงงานวิศวกรรมศาสตร์

หัวข้อโครงงาน	โปรแกรมตรวจสอบมาตรฐานการเขียนโปรแกรมในภาษา C++
ผู้ดำเนินโครงงาน	นายวรชาติ พิรุณรักษ์ รหัส 48364890
อาจารย์ที่ปรึกษา	ดร.พนมขวัญ ริยะมงคล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2551

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร.พนมขวัญ ริยะมงคล)

.....กรรมการ

(ดร.ไพศาล มุณีสว่าง)

.....กรรมการ

(ดร.สุรเดช จิตประไพกุลศาล)

.....กรรมการ

(นายเศรษฐา ตั้งค้าวานิช)

หัวข้อโครงงาน	โปรแกรมตรวจสอบมาตรฐานการเขียนโปรแกรม ในภาษา C++
ผู้ดำเนินโครงงาน	นายวรชาติ พิรุณรักษ์ รหัส 48364890
อาจารย์ที่ปรึกษา	ดร.พนมขวัญ ริยะมงคล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2551

บทคัดย่อ

โครงงานนี้เป็นการศึกษาและพัฒนาโปรแกรมเกี่ยวกับรูปแบบการเขียนโปรแกรม ของภาษา C++ ซึ่งเป็นโปรแกรมที่สามารถตรวจสอบความถูกต้องของโปรแกรมเป้าหมายว่าถูกต้องตามมาตรฐาน ที่ผู้ใช้ต้องการ โดยโปรแกรมสามารถตรวจสอบรูปแบบการเขียนโปรแกรมโดยมีมาตรฐานทั้งหมด 3 รูปแบบ คือ ANSI, GNU และ K&R

ผลที่ได้จากการทำโครงงานนี้ คือ ได้โปรแกรมที่สามารถตรวจสอบมาตรฐานการเขียนโปรแกรม โดยสามารถตรวจสอบได้ทั้งหมด 3 รูปแบบ พร้อมทั้งแสดงจุดผิดพลาด ตามบรรทัดที่เกิดข้อผิดพลาด และเหตุผลถึงจุดผิดพลาดดังกล่าว

Project Title	C++ Coding Standard Detector
Name	Mr. Worachart Pirunruk ID. 48364890
Project Advisor	Panomkhawn Riyamongkol, Ph.D.
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic Year	2008

ABSTRACT

This project studied and developed a program for checking coding standards in C++. This program can verify a target source code and detect fault from the standard that the user chooses. In addition, this program can verify source code in 3 main standard; ANSI, GNU and K&R.

The result of this project is the program that can detect faults from a target source code in 3 main standards and can also display line number and detail of faults in the source code.

กิตติกรรมประกาศ

ขอขอบคุณท่านอาจารย์สุรเดช จิตประไพกุลศาล ที่ให้การช่วยเหลือ คอยให้คำแนะนำ แหล่งศึกษาค้นคว้า อีกทั้งวิธีการเขียนโปรแกรม อาจารย์พนมขวัญ ริยะมงคล อาจารย์ที่ปรึกษา และอาจารย์กรรมการทุกท่าน

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

เนื่องด้วยศักยภาพของคอมพิวเตอร์ในปัจจุบันนั้นมีความล้ำสมัยเป็นอย่างมาก และโปรแกรมคอมพิวเตอร์ถูกพัฒนาให้ใช้งานสะดวก และช่วยในการทำงานมากยิ่งขึ้นจึงทำให้โปรแกรมคอมพิวเตอร์ในปัจจุบัน มีความซับซ้อนมากขึ้น สวนใหญ่ได้รับการพัฒนาโดยโปรแกรมเมอร์หลายคน หลายกลุ่ม แบ่งหน้าที่ความรับผิดชอบในแต่ละส่วนงานของโปรแกรม ด้วยเหตุนี้จึงก่อให้เกิดปัญหาเรื่องวิธีการเขียนโปรแกรม มาตรฐานการเขียนโปรแกรม(Coding Standard) ของแต่ละบุคคลไม่สอดคล้องกันจึงก่อให้เกิดปัญหา ดังนี้

- โค้ดที่มีความสัมพันธ์สอดคล้องกัน ยากจะสื่อความหมายให้แต่ละบุคคลเข้าใจตรงกัน
- สิ้นเปลืองเวลาทำความเข้าใจในชิ้นงานส่วนอื่น ซึ่งมีวิธีการเขียนโค้ดที่แตกต่างกัน
- ยากที่จะมานำพัฒนา เนื่องจากไม่เข้าใจโค้ดของผู้เขียนคนก่อน หรือใช้เวลานาน

ดังนั้นผู้จัดทำการโครงงานได้เล็งเห็นถึงความสำคัญของ มาตรฐานการเขียนโปรแกรมจึงได้จัดทำโครงงานนี้ขึ้น เพื่อความสะดวกและง่ายต่อการตรวจสอบความถูกต้องมาตรฐานการเขียนโปรแกรมคาดว่าการศึกษาโครงงานนี้จะก่อให้เกิดผลประโยชน์ต่อทั้งตัวผู้ที่มาศึกษา และตัวโปรแกรมเมอร์เอง

1.2 วัตถุประสงค์ของโครงงาน

- สร้างโปรแกรมประยุกต์(Application) ที่สามารถตรวจสอบซอร์สโค้ด (Source Code) เป้าหมาย ว่าถูกต้องตรงมาตรฐานการเขียนโปรแกรมตามที่ผู้ใช้ต้องการหรือไม่
- สามารถแสดงได้ว่าเกิดข้อผิดพลาด ณ จุดใดในซอรสโค้ด

1.3 ขอบเขตของโครงงาน

- ได้โปรแกรมประยุกต์ที่สามารถตรวจสอบมาตรฐานการเขียนโปรแกรมได้
- สามารถตรวจสอบโปรแกรมเป้าหมาย ว่าถูกต้องตามมาตรฐานการเขียนโปรแกรมหรือไม่ โดยสามารถตรวจสอบการเว้นวรรค, การย่อหน้า และตำแหน่งของปีกาได้
- สามารถแสดงจุดผิดพลาด หรือบรรทัดที่เกิดข้อผิดพลาดได้ เมื่อทำการตรวจสอบ

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษารูปแบบของ Coding Standard ในภาษา C++
2. ศึกษาโปรแกรม Astyle และโปรแกรม Doxygen
3. ศึกษาวิธีการเขียนโปรแกรมประยุกต์ที่ทำงานบนวินโดส์
4. พัฒนาโปรแกรม พร้อมทั้งทดสอบการใช้งาน
5. สรุปผลการทำงานและเขียนคู่มือการใช้งาน

โครงงานเริ่มต้นเดือน มิถุนายน 2551 สิ้นสุดเดือนกุมภาพันธ์ 2552 รวมเป็นเวลา ทั้งสิ้น 9 เดือน ดังนี้

1.5 แผนการดำเนินงาน

ตารางที่ 1.1 ระยะเวลาและแผนการดำเนินงาน

ระยะเวลาและแผนการดำเนินงาน	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
1. ศึกษารูปแบบของ Coding Standard ในภาษา C++									
2. ศึกษาโปรแกรม Astyle และโปรแกรม Doxygen									
3. ศึกษาวิธีการเขียนโปรแกรมประยุกต์ที่ทำงานบนวินโดส์									
4. พัฒนาโปรแกรม พร้อมทั้งทดสอบการใช้งาน									
5. สรุปผลการทำงานและเขียนคู่มือการใช้งาน									

1.6 ผลที่คาดว่าจะได้รับ

- ได้ความรู้ความเข้าใจการเขียนโค้ดในภาษา C++ เพิ่มมากขึ้น
- ได้ความรู้เพิ่มเติมเรื่องการเขียนโปรแกรมประยุกต์บนระบบปฏิบัติการวินโดร์
- ได้ความรู้เพิ่มเติมเกี่ยวกับวิธีการเขียนโปรแกรมที่ตรงตามมาตรฐานการเขียนโปรแกรม

1.7 งบประมาณของโครงงาน

1.7.1 ค่าใช้จ่ายระหว่างจัดทำโครงงาน	600	บาท
- ค่าถ่ายเอกสาร		
- ค่าหนังสือ		
- ค่าวัสดุคอมพิวเตอร์ เช่น แผ่นซีดี		
1.7.2 ค่าใช้จ่ายการจัดทำรูปเล่มรายงาน	400	บาท
- ค่ากระดาษที่ใช้จัดทำรูปเล่ม		
- ค่าเข้ารูปเล่ม		
รวม	1,000	บาท

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในการพัฒนาโปรแกรมในโครงการนี้ จะนำโปรแกรมที่มีอยู่มาเป็นพื้นฐานในงานที่จะทำการพัฒนาขึ้น ซึ่งในการโครงการนี้ ได้มีสิ่งที่ได้ทำการศึกษา คือ รูปแบบของ Coding Standard ในภาษา C++ เพื่อช่วยให้มีความรู้ความเข้าใจเพิ่มมากขึ้นในการปฏิบัติโครงการนี้ ต่อมาจำเป็นที่ต้องศึกษาการทำงานของโปรแกรม Astyle ซึ่งเป็น Open Source ที่ช่วยในการตรวจสอบ Coding Standard ว่ามีข้อด้อยหรือข้อเด่นประการใดเพื่อนำไปปรับใช้ในโครงการนี้ อีกทั้งโปรแกรม Doxygen ซึ่งเป็นโปรแกรมในการช่วยจัดทำเอกสาร ซึ่งมาจากตัว Source Code ซึ่งสามารถจำแนก Class, function รวมทั้ง Header ออกมาแสดงในรูปแบบ HTML เพื่อช่วยให้ง่ายต่อการแสดงผล ต่อมาจำเป็นต้องศึกษาวิธีการสร้าง Windows Application เพื่อทำ Interface ให้ผู้ใช้สามารถใช้งานโปรแกรมได้อย่างสะดวกมากขึ้น

2.1 รูปแบบของ Coding Standard ในภาษา C++[1][2]

กล่าวคือ Standard นั้นมีความสำคัญ เนื่องด้วยสามารถช่วยให้ทุกๆคนที่ทำงานในแนวทางเดียวกันรู้สึกเหมือนว่า ได้เดินมาในเส้นทางเดียวกัน ไม่ได้ขึ้นอยู่กับรูปแบบของคนใดคนหนึ่ง

2.1.1 ข้อดีของ Coding Standard

- โปรแกรมเมอร์สามารถเข้าใจในตัว Code ได้ง่ายขึ้น และรู้ว่าจะเป็นไปในทางใด
- โปรแกรมเมอร์คนอื่นที่จะนำไปพัฒนาสามารถทำได้เร็วขึ้น
- ลดข้อผิดพลาดในการเขียน Code ลง

2.1.2 ข้อเสียของ Coding Standard

- บุคคลส่วนใหญ่ไม่ค่อยให้ความสำคัญต่อ Standard
- Standard ลดความคิดสร้างสรรค์
- Standard ถูกบังคับให้ใช้หลายโครงสร้าง

2.1.3 Name

ชื่อเป็นหัวใจหลักของโปรแกรม พยายามทำให้ชื่อที่ถูกตั้งขึ้นมีความสอดคล้องกับตัวโปรแกรม หรือตั้งชื่อให้กระชับ ซึ่งมีเฉพาะโปรแกรมเมอร์ที่เข้าใจระบบเท่านั้นที่จะสามารถตั้งชื่อให้กระชับได้ ถ้าชื่อออกมากระชับแล้วนั้นจะทำให้ทุกๆอย่างดูกระชับด้วยตัวของมันเอง, เข้าใจถึงความสัมพันธ์กัน และเข้าใจความหมายได้ชัดเจน

2.1.3.1 Class name

- พยายามตั้งชื่อให้สอดคล้องต่อการทำงานภายใน Class

- ในบางเวลา Suffixes อาจช่วยในการตั้งชื่อได้ เช่น ถ้าในระบบของท่านนั้นใช้ agent ดังนั้นอาจตั้งชื่อเป็น DownloadAgent แสดงถึงการทำงานที่แท้จริง

- ใช้อักษรตัวพิมพ์ใหญ่เพื่อแบ่งคำ เช่น LinkList

- อย่าใช้ underbars ('_')

ตัวอย่าง

Class NameOneTwoThree

Class Name

2.1.3.2 Method and Function names

- โดยส่วนมากแล้วทุกๆ Method และ Function มันเป็นการกระทำ ดังนั้นชื่อควรบ่งบอกให้ชัดเจนว่าทำอะไร เช่น CheckForError() รวมไปถึง ErrorCheck(), DumpDataToFile() รวมไปถึง DataFile() วิธีการนี้จะช่วยให้เข้าใจการทำงานของ Method หรือ Function ได้อย่างชัดเจน

- บางครั้ง Suffix อาจจะช่วยในการตั้งชื่อได้

  * Max - เพื่อหมายถึงค่าสูงสุด
  * Cnt - เพื่อบอกถึงค่าที่กำลังในอยู่ในปัจจุบัน หรือ Current
  * Key - ค่าหลัก หรือKey Value

  ตัวอย่าง RetryMax หมายถึง จำนวนครั้งของการทดลองสูงสุด

- บางครั้ง Prefix อาจจะช่วยในการตั้งชื่อได้

  * Is - เพื่อถามคำถามในบางสิ่ง หรือให้คนอื่นทราบว่ามันคือคำถาม
  * Get - รับค่า
  * Set - ตั้งค่า

  ตัวอย่าง IsHitRetryLimit

- ใช้รูปแบบการตั้งชื่อเหมือนกับ class name

ตัวอย่าง

```
class NameOneTwoThree
{
Public:
        int     DoIt();
        void    HandleError();
}
```

2.1.3.3 Class Attribute name

- ชื่อของ Attribute ควรขึ้นต้นด้วยตัวอักษร 'm'

- หลังจาก 'm' ใช้กฎเหมือน class name
- 'm' บ่งบอกถึง modified เหมือน 'p' คือ pointer

ตัวอย่าง

```
class NameOneTwoThree
    {
    Public:
        int     VarAbc();
        int     ErrorNumber();
    Private:
        int     mVarAbc;
        int     mErrorNumber;
        string* mpName;
    }
```

2.1.3.4 Variable Names on the Stack

- ใช้ตัวพิมพ์เล็กทั้งหมดในการตั้งชื่อ
- ใช้ Underbars ('_') ในการแบ่งคำ

ตัวอย่าง

```
int NameOneTwoThree::HandleError(int errorNumber)
    {
        int             error = OsErr();
        Time            time_of_error;
        ErrorProcessor  error_processor;
        Time*           p_out_of_time = 0;
    }
```

2.1.3.5 Pointer Variables

- pointer ควรที่จะขึ้นต้นด้วยตัว 'p' เป็นส่วนมาก
- วาง * ไว้ใกล้ชนิดของ pointer ไม่ใช่ที่ชื่อตัวแปร

ตัวอย่าง

```
String*  pName = new String;
String* pName, name, address;  // pName เท่านั้นที่เป็น pointer
```

2.1.3.6 Reference Variables and Functions Returning References

- Reference ควรขึ้นต้นด้วยตัว 'r'

ตัวอย่าง

```
Class test
{
Public:
        void            DoSomething(StatusInfo&  rStatus);
        StatusInfo&     rStatus();
        const StatusInfo&   Status() const;
private:
        StatusInfo&     mrStatus;
}
```

2.1.3.7 Global Variables

- Global Variables ควรขึ้นต้นด้วยตัว 'g'

ตัวอย่าง

```
Logger      gLog;
Logger*     gpLog;
```

2.1.3.8 Global Constants

- Global Constants ควรถูกแบ่งคำด้วยตัว '_'

ตัวอย่าง

```
const  int  A_GLOBAL_CONSTANT = 5;
```

2.1.3.9 #Define and Macro Names

- ใส่ #define และ macro เป็นตัวพิมพ์ใหญ่ทั้งหมด และแบ่งคำด้วย '_'

ตัวอย่าง

```
#define MAX(a,b) blah
#define IS_ERR(err) blah
```

2.1.3.10 C Function Names

- ในภาษา C++ แต่ละโปรเจคควรที่จะมี C Function เล็กน้อย
- สำหรับ C Functions ใช้ตัวพิมพ์เล็กทั้งหมด และแบ่งคำด้วยตัว '_'
- ทำเช่นนี้จะช่วยให้ C functions แตกต่างอย่างมากกับชื่อใน C++ อื่นๆ

ตัวอย่าง

```
int some_bloody_function()
```

```
{
}
```

2.1.3.11 C++ File Extensions

- ใช้ .h เพื่อบ่งบอกว่าเป็น Header File และ .cc ใน source file
- Header Files จะประกอบไปด้วย: .h, .hh, .hpp
- Source Files จะประกอบไปด้วย: .c, .cc, .cpp

2.1.4 Documentation

Comment ควรที่จะสามารถบอกโปรแกรมเมอร์ได้ว่า Code ที่อยู่ถัดจากนี้ทำงานอย่างไร และควรที่จะตอบคำถามโปรแกรมเมอร์คนอื่นที่มาศึกษาได้ เช่น ทำงานอย่างไร, ทำอะไร หรือทำไปทำไม

ถ้าท่านคิดว่า Code ของท่านจะไม่มีโปรแกรมเมอร์คนไหนสงสัยเลย แสดงว่าท่านได้โกหกตัวเองแล้ว ไม่เคยมีระบบใหญ่ที่สมบูรณ์และสามารถอธิบายความหมายของ Code ได้ด้วยตัวมันเอง ดังนั้นสิ่งที่จะช่วยให้โปรแกรมเมอร์ที่มาศึกษา Code ต่อจะเข้าใจ มีดังนี้

- พยายามคิดที่จะตอบคำถามของโปรแกรมเมอร์คนอื่น ให้ได้ว่าโปรแกรมเมอร์เหล่านั้น ต้องการที่จะทราบอะไรเกี่ยวกับโปรแกรมของเรา
- ชื่อตัวแปร
- ชื่อClass
- การจำแนก Class
- การจำแนก Method
- ชื่อไฟล์

2.1.4.1 Include Statement Documentation

Include ควรที่จะเขียน Comment ด้วยเพื่อบอกว่าทำไมเราจึง Include

ตัวอย่าง

```
#ifndef XX_h
#define XX_h

// SYSTEM INCLUDES
//
#include                    // standard IO interface
#include                    // HASA string interface
```

#include // USES auto_ptr

2.1.4.2 Block Comment

ตัวอย่าง

```
{
    // Block1 (Comment ที่เป็นประโยชน์สำหรับ Block 1)
    ... Some Code
    {
        // Block2 (Comment ที่เป็นประโยชน์สำหรับ Block2)
        ... Some Code
    } // End Block2
} // End block1
```

การกระทำเช่นนี้จะช่วยให้จับคู่ Block ในการเขียนโค้ดได้ง่ายขึ้นหากคุณไม่ได้ใช้ Compiler ที่มีความฉลาดมากนัก

2.1.5 Classes

Class ก็คือ แม่แบบของ Object ในการใช้งาน Object เราจะต้องประกาศก่อนว่า Object นั้นอยู่ใน Class ใด

Object ที่อยู่ใน Class เดียวกันจะมี Properties และ Method เหมือนๆกัน แต่จะมีค่าใน Properties เหมือนกันหรือไม่ก็ได้ เช่น

รถยนต์A และ รถยนต์B ต่างก็อยู่ในคลาส รถยนต์

รถยนต์A อาจจะมีสีแดง แต่ รถยนต์B อาจจะมีสีดำก็ได้

2.1.5.1 Template

โดย Template คือ รูปแบบหรือแม่แบบ ซึ่งใช้อางอิงในการเขียนโปรแกรมให้ตรงตาม template ที่ตั้งไว้ ดังนั้นควรที่จะปฏิบัติตาม Template หากมีการสร้าง Class ใหม่

```
/**     A one-line description of the class
*
*       #include "XX.h" <BR>
*       -lib
*
*       A longer description.
*
```

```
*       @see    something
*/
#ifndef XX_h
#define XX_h
// SYSTEM INCLUDES
//
// PROJECT INCLUDES
//

// LOCAL INCLUDES
//

// FORWARD REFERENCES
//

class XX
{
Public:
// LIFECYCLE
        /** Default constructor
         *
        *@param from the value to assign to the object
        *
        * @return A reference to this object.
        */
        XX&   operator = (const XX& from);

// OPERATIONS
// ACCESS
// INQUIRY

protected:
```

```
private:
};

// INLINE METHODS
//

// EXTERNAL REFERENCES
//

#endif  // _XX_H_
```

2.1.5.2 LIFECYCLE

ถูกใช้สำหรับ Methods ที่ควบคุมกระบวนการของ object ชิ้นนั้น ส่วนมากรวมไปถึง constructors, destructors และ state Machine Methods

2.1.5.3 OPERATORS

วางทุก Operator ไว้ในส่วนนี้

2.1.5.4 OPERATIONS

วางคลาสที่ไม่ถูกเข้าถึง และ inquiry Methods ไว้ในส่วนนี้ โปรแกรมจะดูส่วนนี้เพื่อหาชิ้นส่วนของ Class's interface

2.1.5.5 ACCESS

วาง Attribute Accessors ไว้ในส่วนนี้

2.1.5.6 INQUIRY

ส่วนนี้ คือ IS* Methods ไม่ว่าคุณจะมีคำถามอะไรเกี่ยวกับobject นี้ สามารถสอบถามได้ใน IS method ตัวอย่าง IsOpen() จะบ่งบอกว่า object นี้เปิดอยู่หรือเปล่า

2.1.6 Prototype Source File

คือ ตัวอย่างของการจัดรูปแบบในการเขียนโปรแกรม

```
#include "XX.h"// Class implemented

//= = = = = = = = = = = = = = = = = = LIFECYCLE = = = = = = = = = = = = = = = = =
XX::XX()
{
```

```
} // XX

XX::XX(const XX&)
{
} // XX

XX::~XX()
{
} // ~XX

// = = = = = = = = = = = = = = = = OPERATOR = = = = = = = = = = = = = = = =
XX&
XX::operator = (const XX&);
{
        return *this;
} // =

// = = = = = = = = = = = = = = = = OPERATIONS = = = = = = = = = = = = = = = =
// = = = = = = = = = = = = = = = = ACCESS = = = = = = = = = = = = = = = = =
// = = = = = = = = = = = = = = = = INQUIRY = = = = = = = = = = = = = = = = =
//////////////////////////////////////// PROTECTED ////////////////////////////////////////
//////////////////////////////////////// PRIVATE /////////////////////////////////////////
```

2.1.7 Formatting

รูปแบบหรือลักษณะในการเขียนโปรแกรม

2.1.7.1 Brace Placement

วาง brace (ปีกกา) ไว้ใต้ keywords:

```
If (condition)                  while (condition)
{                               {
        …                               …
}                               }
```

หรือในอีกกรณี

```
If ( very_long_condition && second_very_long_condition)
{
        …
}
Else if (…)
{
        …
}
```

2.1.7.2 Add Comments to Closing Braces

ใส่ Comment ไว้หลังปีกกาปิด เพื่อช่วยในการอ่าน Code เพื่อให้ทราบว่าใน braces นี้ทำอะไร

```
While (1)
{
    If (valid)
    {
    } // if valid
    Else
    {
    } // not valid
} // end forever
```

2.1.7.3 Indentation/Tabs/Space Policy

- การเว้นวรรค ใช้ 3, 4 หรือ 8 ช่องว่างสำหรับแต่ละขั้น
- Tab ควรถูกตั้งไว้ที่ 8 ตัวอักษร
- เว้นวรรคเข้าไปเท่าไหร่ก็ได้ แต่อย่ามากจนเกินไป โดยส่วนใหญ่แล้ว ในแต่ละ Class หรือ function จะมีไม่เกิน 4 หรือ 5 ขั้น

ตัวอย่าง

```
void func()
{
        if (something bad)
        {
                if ( another thing bad )
                {
```

```
while (more input)
{
}
}
}
}
```

2.1.7.4 If then Else Formatting

Layout

```
if (condition)                    // Comment
{
}
else if (condition)               // Comment
{
}
else                              // Comment
{
}
```

2.1.7.5 Condition Format

- ควรที่จะใส่ค่าคงที่ไว้ทางด้านซ้ายมือของเครื่องหมายเท่ากับหรือไม่เท่ากับเสมอ เช่น

```
if ( 6 = = errorNum) …
```

2.1.7.6 Switch Formatting

- ค่า default ควรที่จะเป็นตัวบ่งบอกถึงข้อผิดผลาด
- ถ้าต้องการจะกำหนดตัวแปรให้ใส่ code เป็น Block

ตัวอย่าง

```
switch (…)
{
    case 1:
        …
    // FALL THOUGH

    case 2:
```

```
{
        int v;
        …
}
break;
default:
}
```

2.2 โปรแกรม Astyle และโปรแกรม Doxygen[4][5]

2.2.1 โปรแกรม Astyle

Astyle หรือ Artistic Style คือ source code ที่ทำหน้าที่จัด Tab, จัด format และทำให้รูปแบบการเขียนโค้ดในภาษา C, C++ และ Java สวยงามขึ้น ซึ่งมีวิธีการใช้งานอยู่ 2 ลักษณะ คือ ใช้งานผ่าน Command Line ใน MSDos หรือรวมเข้าไว้เป็นคลาส คลาสหนึ่งใน Code ที่กำลังพัฒนา

2.2.1.1 Documentation

- การจบบรรทัดจะมีรูปแบบเหมือนกันไฟล์ที่ทำการ Compile แต่ถ้ามีการจบบรรทัดที่หลากหลายจำทำการปรับให้มีรูปแบบที่ใช้มากที่สุดในไฟล์นั้นๆ
- ชนิดของไฟล์ที่สามารถใช้ใน Astyle ได้มีดังนี้ ".java" สำหรับไฟล์ที่เป็น Java, ".cs" สำหรับไฟล์ที่เป็น C# นอกจากนั้นจะเป็นของ C, C++ เช่น ".c", ".cc", ".cpp" เป็นต้น

2.2.1.2 Usage

วิธีการใช้งานผ่านทาง Command Line มีรูปแบบดังนี้

astyle [option] *SourceFile1 SourceFile2 SourceFile3* […]

หรือสามารถบันทึกเป็นอีกไฟล์ที่ชื่อแตกต่างกันได้

astyle [option] <*Original SourceFile*> NewNameSourceFile

2.2.1.3 Option File

ถ้าไม่มีการระบุ option Astyle จะแบ่งวรรคให้ วรรคละ 4 ตัวอักษร มีรูปแบบการเขียนอยู่ 2 วิธี ดังนี้

- Long option

  แต่ละ option จะถูกขึ้นต้นด้วย '--'

  --bracket=attach --indent=space=4

- Short option

แต่ละ option ขึ้นต้นด้วย '-'

-bps4 ซึ่งเหมือนกันการเขียน -b -p -s4

2.2.1.3.1 Style Option

รูปแบบแต่ละรูปแบบของ astyle option มีดังนี้

--style = ansi

รูปแบบ ANSI จะเว้นวรรค 4 ตัวอักษรต่อการย่อหน้า แต่จะไม่สนใจ Namespaces, Classes และ switches

```
namespace foospace
{
int Foo()
{
  if (isBar)
  {
    bar();
    return 1;
  }
  else
    return 0;
}
}
```

--style = gnu

รูปแบบ GNU จะเว้นจากแต่ละปีกกาด้วย 2 ตัวอักษร จะไม่สนใจ Namespaces, Classes และ switches

```
namespace foospace
  {
    int Foo()
    {
      if (isBar)
      {
        bar()
        return 1;
      }
```

```
        Else
         Return 0;
        }
    }
```

--style = kr

รูปแบบ Kerninghan & Ritchie แบ่งแต่ละวรรค 4 ตัวอักษรจะไม่สนใจ Namespaces, Classes และ switches

```
namespace foospace {
int foo() {
  if (isBar) {
    bar();
    return 1;
  } else
    return 0;
}
}
```

ที่ยกตัวอย่างมาด้านบนนั้นเป็นตัวอย่างเพียงคราวๆให้ได้เห็นวิธีการจัดรูปแบบในแต่ละรูปแบบ ซึ่ง Astyle ยังสามารถกำหนด Tab, กำหนดช่องว่างในแต่ละวงเล็บ และอื่นๆ

2.2.2 โปรแกรม Doxygen

Doxygen คือโปรแกรมระบบจัดการเอกสารสำหรับภาษา C++, C, Java, Objective-C, Python, IDL (Cobra and Microsoft favors), Fortran, VHDL, PHP และ C# ซึ่งสามารถช่วยจัดการได้ 3 รูปแบบ คือ

1. สามารถช่วยจัดทำเอกสารซึ่งเป็นแบบออนไลท์ใน HTML หรือเป็นแบบออฟไลท์ใน Latex โดยอ้างอิงจากไฟล์เอกสารต้นฉบับ รวมไปทั้งยังสามารถจัดให้อยู่ในรูป MS-word, PostScript, hyperlinked PDF, compressed HTML และUNIX man pages ไฟล์เอกสารที่จัดขึ้นจะถูกตัดลอกมาจากไฟล์ต้นฉบับโดยตรง ซึ่งทำให้ง่ายขึ้นในการเก็บไฟล์เอกสารควบคู่ไปกับตัว Source code

2. สามารถตั้งค่าให้ Doxygen คัดลอกโครงสร้างของ Code ออกมาจากต้นฉบับที่ไม่เป็นไฟล์เอกสาร วิธีการนี้จะช่วยได้มากในไฟล์ต้นฉบับที่มีขนาดใหญ่

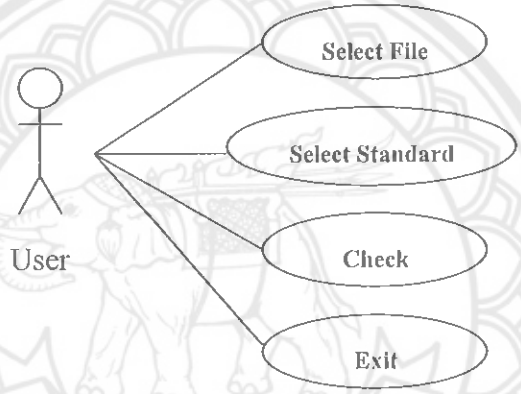
3. สามารถใช้ Doxygen ในการจัดทำเอกสารธรรมดาได้

บทที่ 3

การออกแบบ

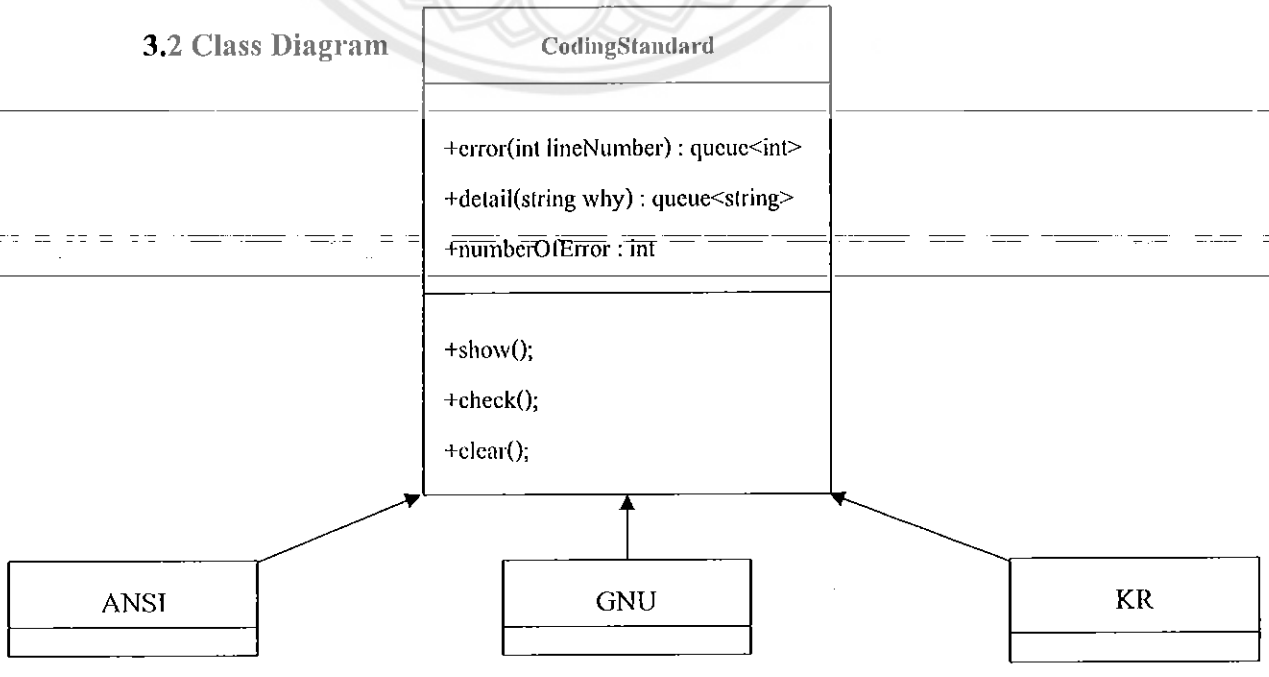
หลังจากที่ได้ศึกษารูปแบบของ Coding Standard โปรแกรมAstyle และ โปรแกรม Doxygen แล้วการออกแบบจึงต้องออกแบบเพื่อให้ผู้ใช้สามารถตรวจสอบรูปแบบของStandard หลักๆได้ทั้งหมด 3 รูปแบบ คือ ANSI, GNU และ K&R อีกทั้งต้องแสดงรูปแบบของแต่ละ Standard ออกมาให้ผู้ใช้ได้เห็น การออกแบบจึงมีขั้นตอนดังนี้

3.1 Use Case Diagram

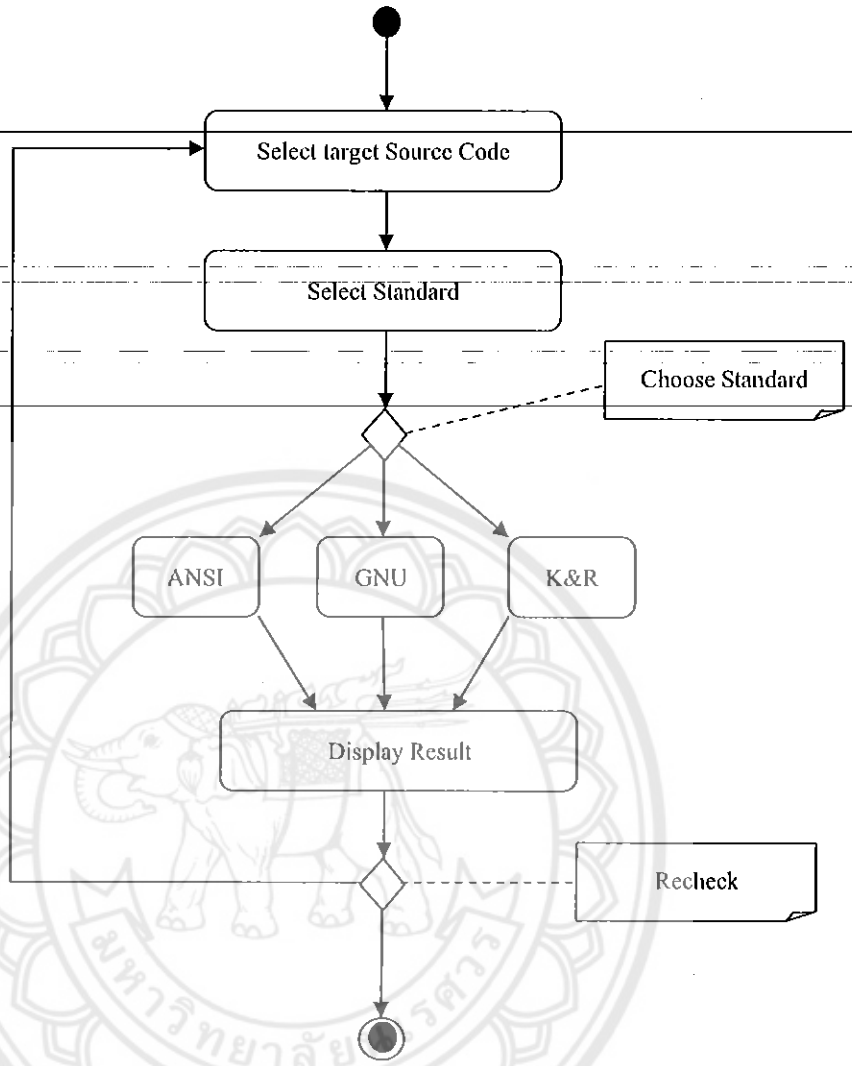


จากแผนภาพจะเห็นได้ว่าผู้ใช้สามารถทำการเลือกไฟล์ที่ต้องการตรวจสอบได้ ต่อจากนั้น สามารถเลือกรูปแบบมาตรฐาน และผู้ใช้สามารถตรวจสอบ หรือจะออกจากโปรแกรมได้

3.2 Class Diagram

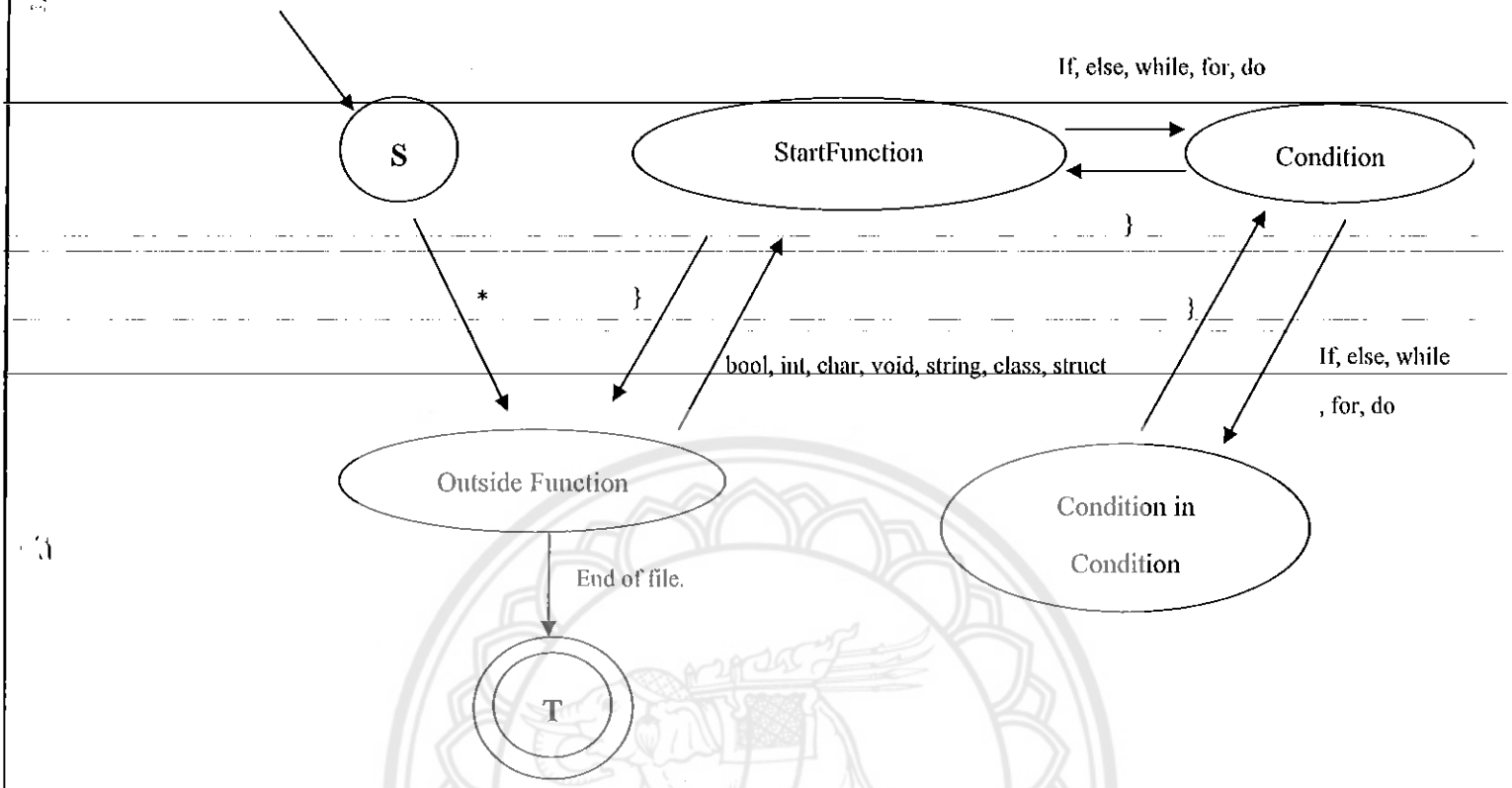


3.3 Activity Diagram



จากแผนภาพแสดงการทำงานของโปรแกรมโดยเริ่มต้นให้ผู้ใช้ ทำการเลือกไฟล์ซอร์สโค้ดที่ต้องการหลังจากนั้นเลือกรูปแบบของมาตรฐานที่ต้องการเช็ค โปรแกรมก็จะทำการตรวจสอบตามรูปแบบมาตรฐานที่ผู้ใช้เลือก แล้วหลังจากโปรแกรมตรวจสอบเสร็จสิ้นก็จะทำการแสดงผลลัพธ์ที่ได้จากการตรวจสอบ เมื่อเสร็จสิ้นขั้นตอนแล้วผู้ใช้ก็สามารถตรวจสอบไฟล์อื่นต่อไปได้โดยที่ไม่ต้องทำการปิดแล้วเปิดโปรแกรมขึ้นมาใหม่

3.4 Automata แสดงกระบวนการการตรวจสอบมาตรฐาน



จากแผนภาพแสดงการเปลี่ยนสถานะของการทำงานของโปรแกรม เริ่มต้นมาโปรแกรมจะมาอยู่สถานะSและรับข้อมูลเข้าทีละบรรทัด เมื่อโปรแกรมตรวจสอบเจอตัวอักษรไรก็ตามก็จะเข้ามาอยู่ที่สถานะ Outside Function โปรแกรมก็จะทำการตรวจสอบย่อหน้าของแต่ละบรรทัด และยังตรวจสอบด้วยอีกว่าถ้าเจอคำดังต่อไปนี้ bool, int, char, void, string, class, struct โปรแกรมก็จะทำการเปลี่ยนสถานะไปยัง สถานะ StartFunction แต่ถ้าไม่ใช่คำดังกล่าวโปรแกรมก็จะวนอยู่ในสถานะเดิม อีกทั้งโปรแกรมก็ทำการตรวจสอบความถูกต้องของย่อหน้าทุกๆบรรทัดไม่ว่าจะอยู่ในสถานะใดก็ตาม ต่อมาโปรแกรมจะทำการตรวจสอบ เมื่อพบคำดังต่อไปนี้ if, else, while, for, do โปรแกรมก็จะทำการเปลี่ยนสถานะอีกครั้ง ไปยังสถานะ condition ให้รู้ว่าตอนนี้โปรแกรมอยู่ภายในลูปดังกล่าว จากการที่ลูปนั้นสามารถซ้อนทับกันได้ โปรแกรมจึงตรวจสอบคำดังที่เข้ามาสถานะปัจจุบัน ถ้าตรวจสอบแล้วพบอีก โปรแกรมก็จะเปลี่ยนสถานะมาเป็นสถานะ Condition in Condition เพื่อให้ทราบว่าปัจจุบันโปรแกรมอยู่ในสถานการณ์ซ้อนกันของลูป แล้วต่อมาเมื่อโปรแกรมเจอปีกกาปิด หรือผ่านไป 2 บรรทัดจากบรรทัดที่ตรวจเจอคำดังกล่าว โปรแกรมก็จะทำการเปลี่ยนสถานะกลับไปยัง สถานะ Condition เช่นเดียวกันกับสถานะที่แล้วโปรแกรมก็จะกลับไปยังสถานะ StartFunction แล้วเมื่อสถานะนี้ตรวจสอบเจอปีกกาปิดก็จะทำการเปลี่ยนสถานะไปยัง Outside Function และเมื่อสถานะนี้เจอจุดสิ้นสุดของโปรแกรม โปรแกรมก็จะจบการทำงานการตรวจสอบรูปแบบมาตรฐาน

บทที่ 4

ผลการทดลอง

หลังจากได้ศึกษาและจัดทำโครงงานนี้ขึ้นจึงทำให้ได้โปรแกรมประยุกต์ในการตรวจสอบมาตรฐานการเขียนโปรแกรมขึ้น ซึ่งสามารถตรวจสอบมาตรฐานได้ทั้งหมด 3 รูปแบบคือ ANSI, GNU และ K&R เมื่อผู้ใช้ต้องการตรวจสอบมาตรฐาน ก็เพียงแค่เลือกมาตรฐานตามต้องการหลังจากนั้นเลือกไฟล์ซอร์สโค้ดที่ต้องการเช็ค โปรแกรมก็จะแสดงผลบรรทัดที่ไม่ถูกต้องตามมาตรฐานพร้อมทั้งบอกเหตุผลของข้อผิดพลาดดังกล่าว

4.1 แสดงตัวอย่างรูปแบบของ Standard ที่ถูกต้อง

4.1.1 ANSI

```
int Foo(bool isFoo)
{
  if (isFoo)
  {
    bar( );
    return 1;
  }
  else
    return 0;
}
```

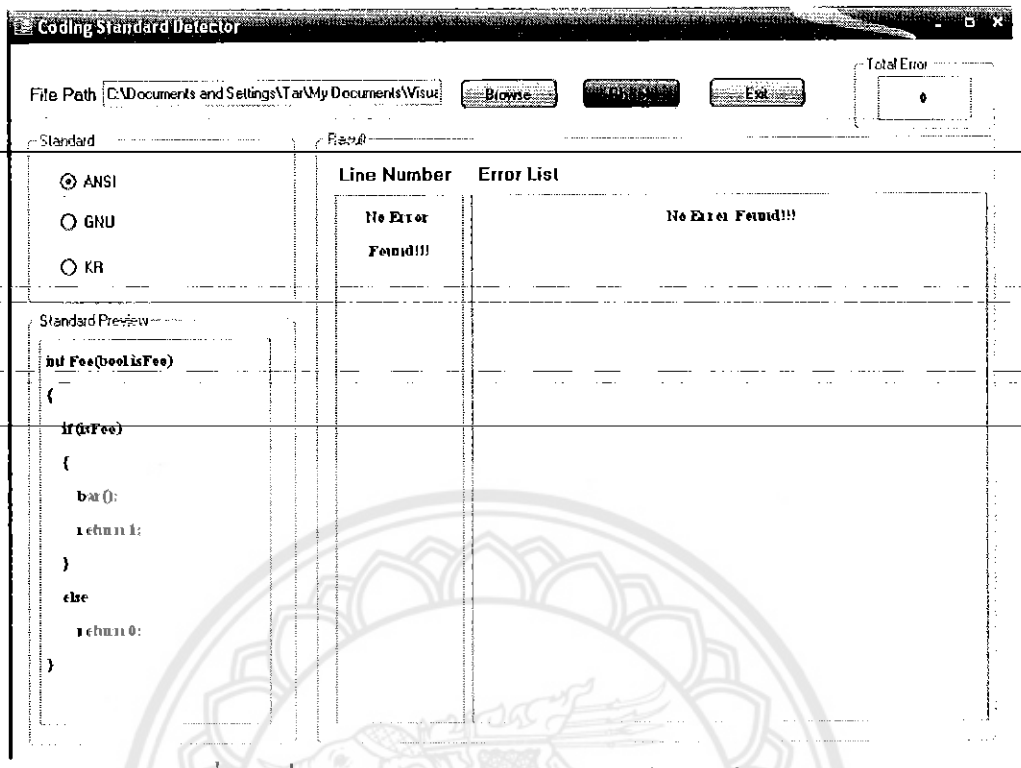
4.1.2 GNU

```
int Foo(bool isFoo)
  {
   if (isFoo)
    {
     bar( );
     return 1;
    }
   else
     return 0;
  }
```

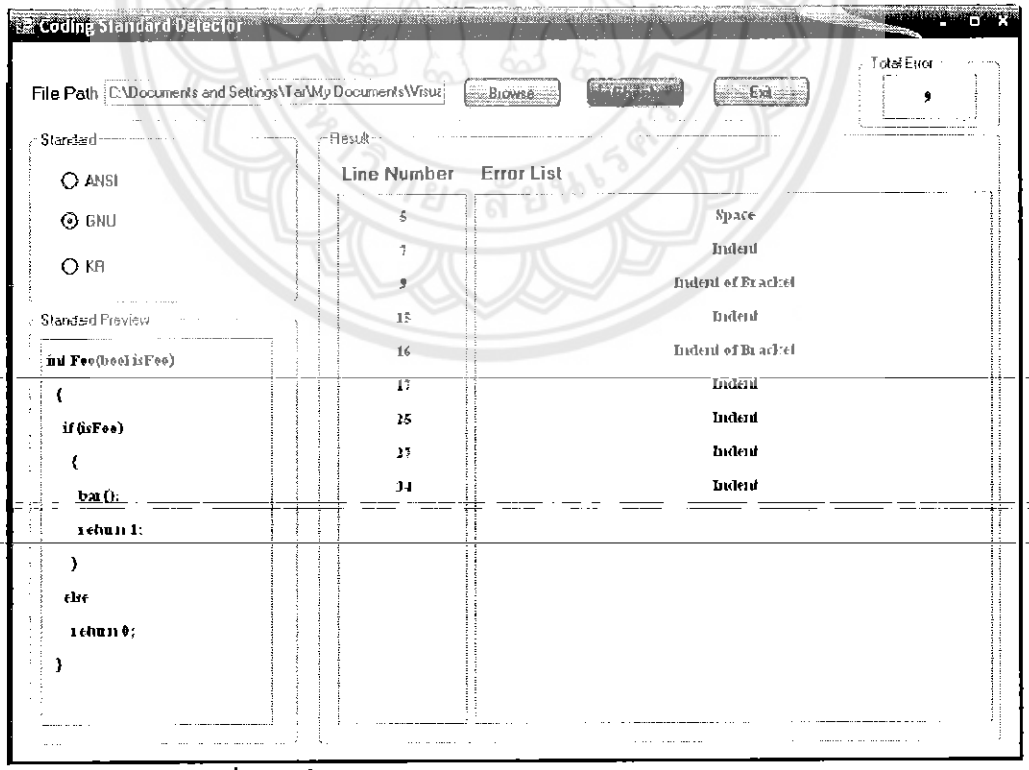
4.1.3 K&R

```
int Foo(bool isFoo) {
   if (isFoo) {
      bar( );
      return 1;
   } else
      return 0;
}
```

ตัวอย่างของโปรแกรมประยุกต์



รูปที่ 4.1 เมื่อโปรแกรมทำการตรวจสอบแต่ไม่พบข้อผิดพลาด



รูปที่ 4.2 เมื่อโปรแกรมทำการตรวจสอบแต่พบข้อผิดพลาด

4.2 ผลการทดลอง

4.2.1 ANSI

4.2.1.1 ตรวจสอบรูปแบบการวางปีกกา

```
1 #include <iostream>
2 using namespace std;
3
4 void foo(bool isFoo)
5 {
6     cout<<"Hello \n";
7 }
8
9 void foo(bool isFoo){
10      cout<<"Hello \n";
11 }
12
13 char foo(bool isFoo)
14 {
15     return 'a';
16 }
17
18 int foo(bool isFoo)
19     {
20     return 0;
21 }
22
23 bool foo(bool isFoo)
24 {
25     return true;
26     }
27
```

รูปที่ 4.3 ซอร์สโค้ดที่ใช้ทดสอบรูปแบบที่ 4.2.1.1

ผลที่ได้รับ

Line Number	Error Detail
9	Bracket
19	Bracket
26	Indent

จากการทดสอบรูปแบบการวางปีกกาดัง จะเห็นได้ในบรรทัดที่ 5 และ 7 มีการวางปีกกาที่ถูกต้องตามมาตรฐาน แต่ในบรรทัดที่ 9 มีการวางปีกกาที่ผิดจากมาตรฐาน คือนำไปวางไว้หลัง function, บรรทัดที่ 19 วางปีกกาเปิดไม่ตรงย่อหน้า คือมีการวางเกินไปหนึ่งย่อหน้า และบรรทัดที่ 26 วางปีกกาปิดไม่ตรงย่อหน้า คือมีการวางเกินไปหนึ่งย่อหน้า

4.2.1.2 ตรวจสอบรูปแบบการย่อหน้า

```
1     #include <iostream>
2 using namespace std;
3
4 void foo(bool isFoo)
5 {
6 cout<<"Test \n";
7     cout<<"Test \n";
8         cout<<"Test \n";
9 }
10
11 char foo(bool isFoo)
12 {
13 cout<<"Test \n";
14     cout<<"Test \n";
15         return 'F';
16 }
17
18 int foo(bool isFoo)
19 {
20 cout<<"Test \n";
21     cout<<"Test \n";
22         return '0';
23 }
24
25 bool foo(bool isFoo)
26 {
27 cout<<"Test \n";
28     cout<<"Test \n";
29         return false;
30 }
31
```

รูปที่ 4.4 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.1.2

ผลที่ได้รับ

Line Number	Error Detail
1	Indent
6	Indent
8	Indent
13	Indent
15	Indent
20	Indent
22	Indent
27	Indent

29	Indent

จากการทดสอบรูปแบบการย่อหน้า จะเห็นได้ว่าย่อหน้าที่ถูกต้องเมื่ออยู่ในฟังก์ชั่น คือทำการย่อหน้าหนึ่งครั้ง แต่ในบรรทัดดังปรากฏในตารางผลที่ได้รับ มีการวางย่อหน้าที่ไม่ตรงตามที่มาตรฐานกำหนดไว้

4.2.1.3 ตรวจสอบรูปแบบการใช้ if...else

```
1 #include <iostream>
2 using namespace std;
3
4 void foo(bool isFoo)
5 {
6     int x = 0;
7     if (isFoo)
8     {
9         x += 1;
10        isFoo = false;
11    }
12    else
13    {
14        x -= 1;
15        isFoo = true;
16    }
17    if (isFoo)
18        x += 1;
19    else
20        x -= 1;
21 }
```

รูปที่ 4.5 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.1.3

```
23 char foo(bool isFoo)
24 {
25     int x = 0;
26 if (isFoo)
27     {
28         x += 1;
29         isFoo = false;
30     }
31 else
32     {
33         x -= 1;
34         isFoo = true;
35     }
36 if (isFoo)
37         x += 1;
38 else
39         x -= 1;
40     return 'F';
41 }
42
43 int foo(bool isFoo)
44 {
45     int x = 0;
46         if (isFoo)
47     {
48         x += 1;
49         isFoo = false;
50     }
51         else
52     {
53         x -= 1;
54         isFoo = true;
55     }
56         if (isFoo)
57         x += 1;
58         else
59         x -= 1;
60     return '0';
61 }
63 bool foo(bool isFoo)
64 {
65 int x = 0;
66     if (isFoo)
67     {
68             x += 1;
69         isFoo = false;
70     }
```

รูปที่ 4.6 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.1.3 (ต่อ)

```
71      else
72      {
73      x -= 1;
74          isFoo = true;
75      }
76      if (isFoo)
77              x += 1;
78      else
79      x -= 1;
80      return false;
81  }
```

รูปที่ 4.7 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.1.3 (ต่อ)

ผลที่ได้รับ

Line Number	Error Detail
26	Indent
31	Indent
36	Indent
38	Indent
46	Indent
51	Indent
56	Indent
58	Indent
65	Indent
68	Indent
73	Indent
77	Indent
79	Indent

จากการทดลองรูปแบบการใช้ if...else บรรทัดที่ 4 – 21 มีการใช้ if...else ที่ถูกต้องตามมาตรฐานกำหนด แต่ในบรรทัดที่ 23 เป็นต้นมา มีการทดลองการวาง if...else ที่แตกต่างออก ดังบรรทัดที่ 26 มีการวาง if ที่ผิดย่อหน้า คือไม่มีการย่อหน้า, บรรทัดที่ 31 มีการวาง else ผิด คือไม่มีการย่อหน้า, บรรทัดที่ 36 และ 38 จะเป็นเหมือนกับที่กล่าวไปข้างต้น, บรรทัดที่ 46, 51, 56, 58 มีการวาง if...else ผิดย่อหน้า คือทำการย่อหน้าเกินกว่าที่กำหนด และ บรรทัดที่ 68, 77, 73, 79 แสดงถึงการย่อหน้าผิดภายใน if...else คือมีทั้งย่อหน้าไม่ถึงที่กำหนด และย่อหน้าเกินกว่ากำหนด

4.2.1.4 ตรวจสอบรูปแบบการใช้ while

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 void foo(bool isFoo)
 5 {
 6     bool ok = true;
 7     while (ok)
 8     {
 9         ok = false;
10     }
11 }
12
13 char foo(bool isFoo)
14 {
15     bool ok = true;
16 while (ok)
17     {
18     ok = false;
19     }
20     return 'F';
21 }
22
23 int foo(bool isFoo)
24 {
25     bool ok = true;
26         while (ok)
27     {
28             ok = false;
29     }
30     return '0';
31 }
32
```

รูปที่ 4.8 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.1.4

ผลที่ได้รับ

Line Number	Error Detail
16	Indent
18	Indent
26	Indent
28	Indent

จากการทดลองตรวจสอบการใช้ while พบว่า บรรทัดที่ 16 และ 26 เป็นการวางตำแหน่งผิดย่อหน้า คือย่อหน้าไม่ถึงที่กำหนด และย่อหน้าเกินกำหนด ส่วนในบรรทัดที่ 18 และ 28 เป็นการแสดงการย่อหน้าผิดรูปแบบ ภายในลูป while คือมีการย่อหน้าไม่ถึงที่กำหนด และย่อหน้าเกินกำหนด

4.2.1.5 ตรวจสอบรูปแบบการใช้ for

```
1 #include <iostream>
2 using namespace std;
3
4 void foo(bool isFoo)
5 {
6     int a = 0;
7     for (int x = 0; x < 5; ++x)
8     {
9         a += 1;
10    }
11 }
12
13 char foo(bool isFoo)
14 {
15    int a = 0;
16 for (int x = 0; x < 5; ++x)
17 {
18    a += 1;
19 }
20    return 'F';
21 }
22
23 int foo(bool isFoo)
24 {
25    int a = 0;
26        for (int x = 0; x < 5; ++x)
27        {
28            a += 1;
29        }
30    return '0';
31 }
32
```

รูปที่ 4.9 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.1.5

ผลที่ได้รับ

Line Number	Error Detail
16	Indent
17	Indent of Bracket
18	Indent
19	Indent of Bracket
26	Indent
27	Indent of Bracket
28	Indent

29	Indent of Bracket

จากการทดลองตรวจสอบรูปแบบการใช้ for การใช้ for ในนั้นจะมีรูปแบบทำนองเดียวกับการใช้ while แต่ได้เพิ่มการตรวจสอบการเช็คตำแหน่งของปีกกาเข้ามาด้วย คือในบรรทัดที่ 17, 19, 27 และ 29 เป็นการวางปีกกาที่ไม่ถูกต้องตามที่กำหนด มีการย่อหน้าไม่ถึงที่กำหนดและเกินกว่ากำหนด

4.2.1.6 ตรวจสอบรูปแบบการซ้อนกันของ if..else

```
1 int foo(bool isFoo)
2 {
3     int x = 0;
4 if (isFoo)
5     {
6     if (x > 0)
7     {
8                 x -= 1;
9             isFoo = false;
10        }
11            else
12            {
13            x += 1;
14        isFoo = true;
15        }
16    }
17        else
18    {
19    if (x > 0)
20        {
21                x -= 1;
22            isFoo = false;
23        }
24            else
25            {
26            x += 1;
27        isFoo = true;
28        }
29    }
30    return 0;
31 }
```

รูปที่ 4.10 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.1.6

ผลที่ได้รับ

Line Number	Error Detail
4	Indent
6	Indent

7	Indent of Bracket
8	Indent
11	Indent
12	Indent of Bracket
14	Indent
17	Indent
19	Indent
21	Indent
24	Indent
25	Indent of Bracket
26	Indent

จากการทดลองตรวจสอบการซ้อนกันของ if...else จะสังเกตได้ว่า บรรทัดที่ 4 และ 6 เป็นการวางผิดย่อหน้าของ if ต่อมาในบรรทัดที่ 7 และ 12 เป็นการวางปีกกาเปิดและปิด ของ if ที่อยู่ใน if อีกต่อหนึ่ง ส่วนในบรรทัดที่ 11 และ 17 เป็นการวางผิดที่ของ else คือเกิดกว่าย่อหน้าที่กำหนด ต่อมาบรรทัดที่ 12 เป็นการแสดงทำนองเดียวกันกับ if คือตรวจสอบปีกกาในลูปใน และบรรทัดที่ 19, 21, 24 และ 26 เป็นการตรวจสอบย่อหน้าในลูปของ if...else ที่ซ้อนทับกัน

4.2.1.7 ตรวจสอบรูปแบบของ Source code โดยรวม

```
1 #include <studio.h>
2
3 int foo(bool isFoo)
4 {
5     if(isFoo){
6         if (Condition)
7     {
8         x = 1;
9         return -1;
10            }
11    return 1;
12        }
13        else
14    {
15            return 2;
16        }
17        if(isFoo)
18        return 3;
19    else
20            return 4;
21        while(isFoo)
22    {
23            x += 1;
24        !isFoo
25    }
26    for (x < 1)
27    {
28            x -= 1;
29    }
30        cout<< "=================================================\n";
31    cout<< "          Program C++ Coding Standard Detector    \n";
32 cout<< "=================================================\n";
33
34    return 0;
35 }
```

รูปที่ 4.11 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.1.7

ผลที่ได้รับ

Line Number	Error Detail
5	Space
5	Bracket
7	Indent of bracket
8	Indent
9	Indent
10	Indent of bracket
11	Indent
12	Indent of bracket

13	Indent
15	Indent
16	Indent of bracket
17	Space
17	Indent
20	Indent
21	Indent
23	Indent
28	Indent
30	Indent
32	Indent

จากการทดสอบแบบแยกรูปแบบที่ผ่านๆมา การทดสอบนี้จึงได้นำการทดสอบทุกอย่างมารวมกันเพื่อแสดงให้เห็นว่าโปรแกรมสามารถตรวจสอบมาตรฐานทั้งหมดของมาตรฐานแบบ ANSI ได้ ดังผลการทดลองที่เกิดขึ้น คือ ในบรรทัดที่ 5 สามารถตรวจสอบได้ทั้ง เว้นวรรค และตำแหน่งการวางปีกกาเปิดที่ไม่ถูกต้อง ต่อมาในบรรทัดที่ 7 มีการวางปีกกาของ if ที่ผิดพลาด และข้อผิดพลาดที่แสดงในตารางแสดงผล คือข้อผิดพลาดที่มาจากการวางผิดตำแหน่งย่อหน้า และตำแหน่งของปีกกาเปิดและปิด ที่ผิดย่อหน้าดังที่ได้แสดงการทดสอบโดยแยกแต่ละการทดสอบมาข้องต้นแล้ว

4.2.2 GNU

4.2.2.1 ตรวจสอบรูปแบบการวางปีกกา

```
1 void foo(bool isFoo){
2   }
3
4 char foo(bool isFoo)
5 {
6     return 'a';
7 }
8
9 int foo(bool isFoo)
10    {
11   return 0;
12    }
13
14 bool foo(bool isFoo)
15   {
16   return 0;}
```

รูปที่ 4.12 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.2.1

ผลที่ได้รับ

Line Number	Error Detail
1	Indent of Bracket
5	Indent of Bracket
7	Indent of Bracket
10	Indent of Bracket
11	Indent
12	Indent of Bracket
16	Indent of Bracket

ทำนองเดียวกันกับการตรวจสอบแบบ ANSI โดย GNU แถบมีรูปแบบไม่แตกต่างกับ ANSI มากเท่าไหร่ ต่างกันที่จำนวนช่องว่างในหนึ่งย่อหน้า ดังนั้นการทดลองจึงไม่แตกต่างกันมาก โดยบรรทัดที่ 1 วางตำแหน่งของปีกกาผิด คือปีกกาเปิดต้องอยู่คนละบรรทัดกับฟังก์ชันและทำการย่อหน้าจากชื่อฟังก์ชันไปหนึ่งย่อหน้า ในบรรทัดที่ 5, 7, 10, 12 และ 16 ก็ทำนองเดียวกัน คือมีการวางปีกกาผิดย่อหน้าตามที่มาตรฐานกำหนด ส่วนในบรรทัดที่ 11 คือการวางโค้ดผิดย่อหน้า

4.2.2.2 ตรวจสอบรูปแบบการย่อหน้า

```
1 void foo(bool isFoo)
2   {
3 cout<<"Hello";
4   }
5
6 char foo(bool isFoo)
7   {
8   cout<<"Hello";
9     return 'a';
10  }
11
12 int foo(bool isFoo)
13  {
14      cout<<"Hello";
15    return 0;
16  }
17
18 bool foo(bool isFoo)
19  {
20    cout<<"Hello";
21    return 0;
22  }
```

รูปที่ 4.13 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.2.2

ผลที่ได้รับ

Line Number	Error Detail
3	Indent
8	Indent
14	Indent

จากการตรวจสอบการย่อหน้า จะเห็นได้ว่าในบรรทัดที่ 3 มีการวางผิดย่อหน้า คือต้องย่อหน้าไปอีก สองย่อหน้า ต่อมาในบรรทัดที่ 8 ทำนองเดียวกันกับบรรทัดที่ 3 คือวางผิดย่อหน้า โดยต้องทำการย่อหน้าไปอีก หนึ่งย่อหน้า และบรรทัดที่ 14 มีการย่อหน้าเกินไปหนึ่งย่อหน้า

4.2.2.3 ตรวจสอบรูปแบบการใช้ if...else

```
1 void foo(bool isFoo)
2   {
3     int x = 0;
4   if (isFoo)
5     {
6       x++;
7         x++;
8     }
9     else
10      {
11        x--;
12        x--;
13      }
14      if(isFoo)
15        {
16          x--;
17          x--;
18        }
19      else
20      {
21        x++;
22        x++;
23      }
24  if (isFoo)
25        x++;
26      else
27    x--;
28  }
```

รูปที่ 4.14 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.2.3

ผลที่ได้รับ

Line Number	Error Detail
2	Indent
3	Indent
5	Indent of bracket
6	Indent
7	Indent
8	Indent of bracket
9	Indent
10	Indent of bracket
11	Indent
12	Indent

13	Indent of bracket
14	Indent
14	Space
15	Indent of bracket
16	Indent
17	Indent
18	Indent of bracket
19	Indent
20	Indent of bracket
21	Indent
22	Indent
23	Indent of bracket
25	Indent
26	Indent
28	Indent

จากการทดลองการใช้ if...else จะสังเกตได้ว่ารูปแบบการวาง if...else ของรูปแบบ GNU นั้นไม่แตกต่างจากการใช้ if...else ของ ANSI ผลการทดลองจึงออกมาดังตาราง โดยในบรรทัดที่ 5, 8, 10, 13, 15, 18, 20 และ 23 จะเป็นวางผิดที่ของปีกกาเปิดและปิดของ if...else โดยมีทั้งวางหลัง if หรือ else วางผิดย่อหน้า ต่อมาในบรรทัดที่ 14 คือการเว้นวรรคผิดของ if คือต้องการเว้นวรรคหนึ่งครั้ง และบรรทัดที่ 6, 7, 9, 11, 12, 14, 16, 17, 19, 21, 22, 25, 26 และ 28 เป็นการแสดงการวางผิดย่อหน้าภาใน if...else โดยมีทั้งการวางไม่ถึงย่อหน้าที่กำหนด และวางเกินกว่าย่อหน้าที่มาตรฐานกำหนด

4.2.2.4 ตรวจสอบรูปแบบการใช้ while

```
1 void foo(bool isFoo)
2   {
3     bool ok = true;
4     while (ok){
5       ok = false;
6       }
7     while(ok)
8     {
9         ok = false;
10    }
11    while (ok)
12        {
13          ok = false
14        }
15  }
```

รูปที่ 4.15 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.2.4

ผลที่ได้รับ

Line Number	Error Detail
4	Bracket
5	Indent
7	Space
8	Indent of Bracket
11	Indent
12	Indent of Bracket
13	Indent

จากการทดลองการใช้ while จะเห็นได้ว่าบรรทัดทัดที่ 4 เป็นการวางปีกกาผิด คือไปอยู่หลัง while ที่ถูกต้องคือต้องอยู่ในบรรทัดใหม่ แล้วย่อหน้าเข้าไปอีกหนึ่งย่อหน้าจากย่อหน้าของ while ต่อมาในบรรทัดที่ 5, 11 และ 13 คือการวางผิดย่อหน้าในลูป while คือวางไม่ถึงย่อหน้าตามที่มาตรฐานกำหนด และในบรรทัดที่ 8 และ 12 คือการวางตำแหน่งของปีกกาเปิดและปิดผิดย่อหน้า

4.2.2.5 ตรวจสอบรูปแบบการใช้ for

```
1 void foo(bool isFoo)
2   {
3     int x = 0;
4     for (int i = 0; i < 5; ++i){
5       x++;
6       }
7     for(int i = 0; i < 5; ++i)
8     {
9         x--;
10      }
11    for (int i = 0; i < 5; ++i)
12      {
13          x = 10;
14          }
15  }
```

รูปที่ 4.16 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.2.5

ผลที่ได้รับ

Line Number	Error Detail
4	Bracket
5	Indent
7	Space
8	Indent of Bracket
13	Indent
14	Indent of Bracket

จากการทดลองการใช้ for ก็ไม่แตกต่างจากการใช้ while โดยในบรรทัดที่ 4 คือมีการวางปีกกาเปิดผิดที่ ต่อมาบรรทัดที่ 5 มีการย่อหน้าผิดตามที่มาตรฐานกำหนด คือย่อหน้าขาดไปหนึ่งย่อหน้า ในบรรทัดที่ 7 แสดงถึงการเว้นวรรคที่ผิดระหว่าง while กับ condition คือหลัง while ต้องเว้นวรรคก่อนหนึ่งเว้นวรรค ในบรรทัดที่ 8 และ 14 เป็นการวางผิดย่อหน้าของปีกกาเปิดและปิด และบรรทัดที่ 13 แสดงการย่อหน้าผิดในลูป while

4.2.2.6 ตรวจสอบรูปแบบการซ้อนกันของ if...else

```
1 void foo(bool isFoo)
2   {
3     if(isFoo)
4       {
5         if (isFoo)
6           {
7               cout<<"Hello";
8             cout<<"World";
9           }
10        else
11          {
12              cout<<"Test";
13            cout<<"Test";
14          }
15      }
16    else
17      {
18        if(isFoo)
19          {
20                cout<<"Test";
21            cout<<"Test";
22          }
23        else
24          {
25          cout<<"Test";
26            cout<<"Test";
27          }
28      }
29  }
```

รูปที่ 4.17 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.2.6

ผลที่ได้รับ

Line Number	Error Detail
3	Space
7	Indent
8	Indent
12	Indent
18	Space
20	Indent
25	Indent

จากการทดลองการซ้อนกันของ if...else ได้ผลทดลองดังตาราง จากบรรทัดที่ 3 และ 18 มีการเว้นวรรคที่ผิด ระหว่าง if หรือ else กับ condition ต่อมาในบรรทัดที่ 7, 8, 12, 20 และ 25 เป็นการย่อหน้าที่ผิดทั้งใน if ที่ยังไม่ได้ซ้อน และเมื่ออยู่ใน if ที่อยู่ใน if อีกทีหนึ่ง

4.2.2.7 ตรวจสอบรูปแบบของ Source code โดยรวม

```
1 int foo(bool isFoo)
2  {
3    if (isFoo)
4      {
5      if(Condition)
6        {
7              x = 1;
8          return 0;
9          }
10     }
11   else
12     {
13       return 1;
14     }
15     while(Condition)
16       {
17         x += 2;
18       return 2;
19     }
20   for (Condition)
21     {
22       x += 3
23       return 3;
24     }
25 if (isFoo)
26     return 0;
27     else
28     return 1;
29   if (isFoo)
30     {
31       if (Condition)
32         return 0;
33     }
34     cout<< "=================================================\n";
35   cout<< "          Program C++ Coding Standard Dectector   \n";
36   cout<< "=================================================\n";
37 }
```

รูปที่ 4.18 source code ที่ใช้ทดสอบรูปแบบที่ 4.2.2.7

ผลที่ได้รับ

Line Number	Error Detail
5	Space
7	Indent
9	Indent of Bracket

15	Indent
16	Indent of Bracket
17	Indent
25	Indent
27	Indent
34	Indent

จากการทดลองโดยแยกแต่ละรูปแบบแล้ว การทดลองด้านบนเป็นการทดลองของซอร์สโค้ดปกติ เพื่อแสดงให้เห็นว่าโปรแกรมสามารถตรวจสอบมาตรฐานแบบ GNU ได้ โดยในบรรทัดที่ 5 มีการเว้นวรรคผิดของ if ต่อมาบรรทัดที่ 7 การวางผิดย่อหน้า ในบรรทัดที่ 9 มีการวางปีกกาปิดของ if ลูปในผิดย่อหน้า ต่อมาในบรรทัดที่ 15, 17, 25, 27 และ 34 มีการย่อหน้าที่ผิด และบรรทัดที่ 9 และ 16 มีการวางตำแหน่งปีกกาผิดย่อหน้า

4.2.3 K&R

เนื่องจากรูปแบบ K&R มีรูปแบบที่แตกต่างจากรูปแบบที่ผ่านมา จึงไม่สามารทดสอบตามที่ทดสอบมาได้ จึงจะทดสอบแบบ source Code โดยรวม

```
1 int foo(bool isFoo)
2 {
3   if(isFoo){
4     if (isCondition){
5         x += 1;
6     return 1;
7   }
8     }
9   else {
10      x += 2;
11    return 0;
12  }
13
14  while (isFoo) {
15      x += 3;
16  isFoo = false;
17  }
18  for (a < 1) {
19      a -= 1;
20    }
21  cout<< "=================================================\n";
22    cout<< "          Program Coiding Standard Detecter     \n";
23 cout<< "=================================================\n";
24 }
```

รูปที่ 4.19 source code ที่ใช้ทดสอบรูปแบบ K&R

ผลที่ได้รับ

Line Number	Error Detail
2	Bracket
3	Space
6	Indent
7	Indent
8	Indent
20	Indent
23	Indent

เนื่องรูปแบบของ K&R นั้นมีรูปแบบของมาตรฐานที่แตกต่างออกไปกับแบบ ANSI และ GNU จึงไม่สามารถทำการทดสอบที่ทำการทดสอบกับทั้งสองรูปแบบก่อหน้านี้ได้ ทางผู้จัดทำจึงได้ทำการทดสอบโดยรวมทั้งหมดของรูปแบบ K&R โดยได้ผลลัพธ์ดังแสดงในตารางผลที่ได้รับ คือ ในบรรทัดที่ 2 มีข้อผิดพลาดคือการวางปีกกาเปิดผิดตำแหน่ง รูปแบบ K&R นั้นปีกกาเปิดนั้น

ต้องวางอยู่หลังฟังก์ชัน ต่อมาในบรรทัดที่ 3 ผิดช่องว่างโดยที่ระหว่าง if กับ condition ต้องมีช่องว่างหนึ่งช่อง ส่วนในบรรทัดที่ 6, 7, 8, 20 และ 23 เป็นการแสดงการย่อหน้าที่ผิด คือมีทั้งย่อหน้าไม่ถึงตามที่รูปแบบกำหนด และย่อหน้าเกินกว่ารูปแบบกำหนด

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

ในปัจจุบันโปรแกรมคอมพิวเตอร์มีขนาดใหญ่ และซับซ้อนมากขึ้น จึงทำให้เกิดปัญหาเรื่องมาตรฐานการเขียนโปรแกรมของโปรแกรมเมอร์แต่ละคน

ผู้จัดทำโครงงานได้พัฒนาโปรแกรมสำหรับภาษาC++ เพื่อช่วยตรวจสอบความถูกต้องของมาตรฐานการเขียนโปรแกรม โดยสามารถตรวจสอบการเว้นวรรค, การย่อหน้าและตำแหน่งของปีกาได้ตามมาตรฐาน ดังต่อไปนี้ คือ ANSI, GNU และ K&R

ซึ่งสามารถบ่งบอกถึงบรรทัดและรายละเอียดของข้อผิดพลาดได้อย่างถูกต้อง

5.2 ปัญหาที่พบ

1. เนื่องด้วยตัวโปรแกรมแต่ละ Function มีขนาดใหญ่จึงทำให้ยากต่อการหาจุดพลาดแต่ละครั้งในการ debug โปรแกรม รวมทั้งตัวผู้จัดทำไม่มีความชำนาญในเรื่องมาตรฐานการเขียนโปรแกรมมาก่อนจึงทำให้เสียเวลาในการศึกษา

2. การเขียน Windows application โดยใช้ภาษา C++ ซึ่งมีความยากมากกว่า C# จึงทำให้ผู้จัดทำดำเนินการทำหน้าตา interface ออกมาได้ช้า

3. เนื่องด้วยเวลาจำกัดจึงไม่สามารถสร้าง function ที่ผู้ใช้สามารถกำหนดรูปแบบของมาตรฐานเองได้ทันเวลา

4. โปรแกรมที่ผู้จัดทำโครงงานได้พัฒนาขึ้น มีจุดบกพร่องที่ไม่สามารถตรวจสอบ switch…case ได้เนื่องด้วย condition ดังกล่าวมีรูปแบบที่แตกต่างออกไป

5. Source Code มีหลายบรรทัดมาก จึงยากในการแก้ไข

5.3 แนวทางการแก้ปัญหา และข้อเสนอแนะ

1. ศึกษารูปแบบการตรวจสอบจากโปรแกรม Astyle หรือ Doxygen ให้มากขึ้นเจาะลึกมากขึ้นกว่านี้

2. ทำให้โปรแกรมสามารถตรวจสอบมาตรฐานโดยที่ผู้ใช้ เป็นผู้กำหนดมาตรฐานเองได้

เอกสารอ้างอิง

[1] Scott Meyers. **Effective C++: 55 Specific Ways to Improve Your Programs and Designs.** 3rd Ed., 2005

[2] Todd Hoff. "C++ Coding Standard." [Online]. Available: http://www.possibility.com/Cpp/CppCodingStandard.html. 2008.

[3] Stephen Fry. "GNU Coding Standards." [Online]. Available: http://www.gnu.org/prep/standards/. 2008.

[4] Jim Pattee. "Artistic Style." [Online]. Available: http://astyle.sourceforge.net. 2008.

[5] Dimitri van Heesch. "Doxygen." [Online]. Available: http://www.doxygen.org. 2008.

[6] Stan Lippman. "System::String -> std::string." [Online]. Available: http://blogs.msdn.com/slippman/archive/2004/06/02/147090.aspx. 2008.

[7] Microsoft. "Convert from System String to std string." [Online]. Available: http://www.experts-xchange.com/Programming/Languages/.NET/Visual_CPP/Q_21755751.html. 2008

ประวัติผู้เขียนโครงงาน

ชื่อ นายวรชาติ พิรุณรักษ์

ภูมิลำเนา 111/1-2 ถนนสุรศักดิ์สงวน ต.ศรีราชา อ.ศรีราชา จ.ชลบุรี

ประวัติการศึกษา

- สำเร็จการศึกษาระดับมัธยมศึกษาตอนต้นจากโรงเรียนจุฬาภรณราช วิทยาลัย ชลบุรี
- สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนชลราษฎรอำรุง จังหวัด ชลบุรี
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร

E-mail: **p_worachart2002@hotmail.com**