บทที่ 7 การสร้างโปรแกรมขนาดใหญ่

(Constructing Larger Programs)

ในบทที่แล้วได้กล่าวถึงการทำงานของโปรแกรมที่ประกอบด้วยฟังก์ชันหลายชนิด ฟังก์ชันเหล่านี้อาจจะได้มาจากคลังมาตราฐาน หรือเป็นฟังก์ชันที่นักเขียนโปรแกรมสร้างขึ้นเอง โปรแกรมเหล่านี้เก็บฟังก์ชันต่าง ๆ ไว้ในแฟ้มเดียวกัน การติดต่อระหว่างโปรแกรมหลักและฟังก์ชันสามารถทำได้โดยการส่งผ่านข้อมูลทางพารามิเตอร์ อย่างไรก็ตามเมื่อโปรแกรมมีขนาดใหญ่ขึ้นจนไม่สามารถเก็บฟังก์ชันจำนวนมากไว้ในแฟ้มเดียวกันได้ หรือมีโปรแกรมหลายโปรแกรมที่เรียกใช้ฟังก์ชันเดียวกัน นักเขียนโปรแกรมจึงสร้างโปรแกรมที่มีการเรียกใช้ฟังก์ชันจากแฟ้มข้อมูลอื่น โดยมีการแปลรหัสของแฟ้มข้อมูลแต่ละแฟ้มโดยวิธีการแปลโปรแกรมแบบแยกส่วน แล้วนำรหัสที่แปลแล้วมารวมกัน (link) เพื่อสร้าง โปรแกรมกระทำการ (executable program) ฟังก์ชันที่เก็บไว้ในแฟ้มเหล่านี้สามารถสื่อสารกับฟังก์ชันได้โดยการเรียกใช้ตัวแปรครอบคุม (global variable) และการส่งผ่านพารามิเตอร์ ในฟังก์ชันเหล่านี้จะมีการกำหนดคุณสมบัติเก็บตัวแปร หรือสตอเรจคลาส (storage class)

7.1 ตัวแปรเฉพาะถิ่น

ตัวแปรเฉพาะถิ่น (local variable) คือ ตัวแปรที่กำหนดไว้ในฟังก์ชัน ตัวแปรเหล่านี้จะเป็นที่รู้จักเฉพาะในฟังก์ชันที่ตัวแปรอาศัยอยู่เท่านั้น

สำหรับตัวระบุ (identifier) ที่อยู่ในโปรแกรมนั้นจะมีสตอเรจคลาสซึ่งจะเป็นตัวกำหนดคุณสมบัติของตัวแปร เช่น การมองเห็นโดยส่วนอื่น อายุการทำงาน และตำแหน่งในหน่วยความจำ

7.2 ตัวแปรครอบคลุม

ตัวแปรครอบคลุมปืนตัวแปรที่สร้างขึ้นจากการประกาศตัวแปรแบบภายนอกไว้ที่ส่วนต้นของโปรแกรม การกำหนดฟังก์ชันจะต้องมีการประกาศตัวแปรครอบคลุมและฟังก์ชันนี้จะมีสตอเรจคลาสเป็น Extern โดยปริยาย และสามารถเก็บมูลค่าไว้ในหน่วยความจำได้ตลอดการทำงานของโปรแกรม ตัวแปรครอบคุมและฟังก์ชันจะถูกเรียกใช้ได้โดยฟังก์ชันอื่น ๆ ในโปรแกรมที่มีตำแหน่งถัดจากฟังก์ชันนั้น

ตัวอย่างที่ 7.1 โปรแกรมที่แสดงการใช้ตัวแปรครอบคลุมและตัวแปรเฉพาะถิ่น

```
/*
* Octal dump (using globals).
*/
#include <stdio.h>]

#include <ctype.h>
#define MAXCHARS 8          /* Number of chars on an output line */
int     chars[MAXCHARS];    /* chars for current output line */
int     ingroup;            /* chars in group */
long    cnt;                /* chars in input */
main()
{
        void print_group(void);         /* no parameters now */
        for (; (chars[ingroup] = getchar()) != EOF; cnt++)
                if (++ingroup == MAXCHARS)
                {
                print_group();
                ingroup = 0;
                } /* End if */
        if (ingroup)
                print_group();
        return 0;
} /* End main */
void print_group(void)
{
        int i;
        void print_space(int c);
        printf("%06ld ", cnt - ingroup + 1);
        for (i = 0; i < ingroup; i++)               /* print octal codes */
                printf(" %03o", chars[i]);
        printf("\n");
        for (i = 0; i< ingroup; i++)
        {
                putchar(' ');                       /* space */
                if (isspace(chars[i]))
                        printf_space(chars[i]); /* whitespace */
                else if (isprint (chars[i]))
                        printf(" %c", chars[i]); /* printable */
                else
                        printf("  ");                       /* other chars */
        } /* End for */
        putchar('\n');
} /* End void print_group */
```

```
/*
* print_space (doesn't use globals).
*/
void print_space(int c)
{
        int label;
        if (c == ' ')
                printf(" ");
        else
        {
                switch(c)
                {
                        case '\b' : label = 'b'; break;
                        case '\f' : label = 'f'; break;
                        case '\n' : label = 'n'; break;
                        case '\r' : label = 'r'; break;
                        case '\t' : label = 't'; break;
                        case '\v' : label = 'v'; break;
                        default : label = '?'; break;
                } /* End switch */
                printf("\\%c",label);
        } /* End else */
} /* End void print_space */
```

ผลลัพธ์ที่ได้

```
Enter 8 charecter : abcdefgh
ASCII CODE of a =97  and  octal = 141
ASCII CODE of b =98  and  octal = 142
ASCII CODE of c =99  and  octal = 143
ASCII CODE of d =100  and  octal = 144
ASCII CODE of e =101  and  octal = 145
ASCII CODE of f =102  and  octal = 146
ASCII CODE of g =103  and  octal = 147
ASCII CODE of h =104  and  octal = 150
```

โปรแกรมนี้อ่านข้อมูลนำเข้าแล้วเก็บไว้ในแถวลำดับ chars เพื่อนับจำนวนตัวอักขระ และพิมพ์ข้อมูลในรูปแบบของเลขฐานแปดออกทางจอภาพ

7.3 สตอเรจคลาส

สตอเรจคลาส มี 4 ชนิด คือ auto register static และ extern เมื่อนักเขียนโปรแกรมประกาศตัวแปรในโปรแกรม แล้วสามารถกำหนดสตอเรจคลาสโดยใช้ชื่อคลาสก่อนชนิดของข้อมูล ในกรณีที่นักเขียนโปรแกรมไม่ระบุสตอเรจคลาส คอมไพเลอร์จะถือว่าตัวแปรนั้นเป็นคลาสแบบ auto โดยปริยาย

7.3.1 สตอเรจคลาสแบบ auto

ตัวแปรที่มีสตอเรจคลาสแบบ auto หมายถึง ตัวแปรถูกสร้างขึ้นเมื่อฟังก์ชันที่กำหนดตัวแปรถูกเรียกใช้งาน และถูกทำลายเมื่อฟังก์ชันหยุดทำงาน โดยทั่วไปเรามักจะกำหนดตัวแปรเฉพาะถิ่นของฟังก์ชันให้มีคุณสมบัติของการเก็บข้อมูลแบบ auto ในการประกาศตัวแปรที่ต้องการจัดข้อมูลอัตโนมัติจะใช้คำสั่ง auto

ตัวอย่างที่ 7.2 การประกาศสตอเรจคลาสแบบ auto สำหรับตัวแปรเฉพาะถิ่นในฟังก์ชัน

```
{
    auto int x, y;
    …
}
```

ซึ่งสามารถเขียนได้อีกวิธีหนึ่งคือ

```
{
    int x, y;
    …
}
```

การประกาศ auto int x, y หมายถึงตัวแปร x และ y เป็นตัวแปรเฉพาะถิ่นแบบ auto ดังนั้นตัวแปร x และ y จะเป็นที่รู้จักเฉพาะในฟังก์ชันที่กำหนดตัวแปรทั้ง 2 เท่านั้น ถ้าไม่มีการกำหนดคุณสมบัติของการจัดเก็บเป็นแบบอื่น ตัวแปรเฉพาะถิ่นทุกตัวจะมีคุณสมบัติของการเก็บแบบ auto โดยปริยาย จึงไม่จำเป็นต้องเขียนคำสั่ง auto ไว้ข้างหน้า

7.3.2 สตอเรจคลาสแบบ register

ตัวแปรที่มีการประกาศให้เป็นตัวแปรแบบ register จะมีคุณลักษณะของการจัดเก็บเหมือนแบบ auto เพียงแต่ข้อมูลจะถูกเก็บไว้ในริจิสเตอร์ที่มีความเร็วสูง ตัวแปรที่มีการกำหนดแบบ resgister มักจะเป็นตัวแปรที่มีการทำงานบ่อยครั้งเช่น ตัวนับรอบการวนซ้ำ

ตัวอย่างเช่น

register int counter = 1

หมายถึง ตัวแปรชนิดจำนวนเต็มชื่อ counter ถูกกำหนดให้เก็บค่าไว้ในรีจิสเตอร์ของเครื่องคอมพิวเตอร์ และให้ตัวแปร counter นี้มีค่าเริ่มต้นเป็น 1

7.3.3 สตอเรจคลาสแบบ static

คำสั่ง static ใช้ประกาศตัวระบุ เช่น ตัวแปรและฟังก์ชันให้มีการเก็บในหน่วยความจำแบบสถิต (static storage allocation) ตัวระบุแบบนี้จะคงสภาพอยู่ในหน่วยความจำตั้งแต่โปรแกรมเริ่มทำงานโดยมีการจองเนื้อที่ และกำหนดค่าเริ่มต้นของตัวแปร ตัวระบุแบบ static มี 2 ชนิด คือ ตัวระบุแบบภายนอก (external identifier) เช่นตัวแปรครอบคุม ชื่อฟังก์ชัน และตัวแปรเฉพาะถิ่น

ตัวแปรเฉพาะถิ่นแบบ static อยู่ในหน่วยความจำตราบเท่าที่โปรแกรมทำงาน การกำหนดเนื้อที่ของตัวแปรและการกำหนดค่าเริ่มต้นของตัวแปรชนิดนี้เกิดขึ้นในกระบวนการแปลโปรแกรม นอกจากนี้คำสั่ง static นั้นมีการกำหนดค่าเริ่มต้นของตัวแปรไว้เป็นศูนย์โดยปริยาย

ตัวอย่างที่ 7.3 ฟังก์ชันพิมพ์จำนวนครั้งที่ฟังก์ชันนี้ถูกเรียกใช้

```
/*
*Keep a count of times function is called (working version).
*/
#include<stdio.h>
main()
{
        void testfunc(void);
        testfunc(); testfunc(); testfunc();
        return 0;
} /* End main */
void testfunc(void)
{
        static int cnt = 0;
        printf("testfunc call #%d\n", ++cnt);
}/* End void testfunc */
```

ผลลัพธ์ที่ได้

```
testfunc call #1
testfunc call #2
testfunc call #3
```

ในตัวอย่างนี้ มีการกำหนดการจัดเก็บตัวแปร cnt ซึ่งเป็นตัวแปรแบบ static ในฟังก์ชัน testfunc และตัวแปรนี้ได้รับการกำหนดค่าเริ่มต้นเป็นศูนย์ เมื่อโปรแกรมมีการเรียกใช้ฟังก์ชัน testfunc ในแต่ละครั้งจะมีการเพิ่มค่าให้ตัวแปร cnt และพิมพ์ผลลัพธ์ เนื่องจากโปรแกรมกำหนดให้ cnt เป็นตัวแปรแบบ static ดังนั้นค่าของ cnt จึงยังคงอยู่หลังจากที่เสร็จสิ้นการทำงานในแต่ละรอบ

7.3.4 สตอเรจคลาสแบบ extern

การประกาศให้ตัวระบุหรือตัวแปรเป็น extern นั่นหมายถึงการกำหนดให้ตัวระบุสามารถทำงานได้เหมือนตัวแปรแบบครอบคลุม ซึ่งโปรแกรมอื่น ๆ สามารถเรียกใช้ตัวแปรนั้นได้ การประกาศตัวแปร extern มีรูปแบบดังนี้

extern type ชื่อตัวแปร

เช่น extern int ingroup

การประกาศ extern นี้บอกคอมไพเลอร์ถึงชนิดของตัวแปรและเนื้อที่ที่ต้องการ อย่างไรก็ตามยังไม่มีการจองเนื้อที่สำหรับตัวแปรที่ประกาศเป็น extern แต่จะมีการจองเนื้อที่เมื่อมีการทำงาน เช่น คำสั่ง extern int ingroup บอกว่า ingroup มีการประกาศในแฟ้ม และคำสั่งนี้ทำให้ตัวแปร ingroup ถูกเรียกใช้ได้โดยฟังก์ชันใด ๆ และตัวแปรนี้ทำหน้าที่เหมือนตัวแปรครอบคลุม

ตัวอย่างที่ 7.4 โปรแกรมที่มีการใช้คำสั่ง extern

```
#include <stdio.h>
void function1(int xx);
int main() {
        int num;
        extern int aa;
        printf("Enter number:");
        scanf("%d %d",&num,&aa);
        function1(num);
        return 0;
}

void function1(int xx)
{
        static int yy;
        extern int aa;
        yy=xx+aa;
        printf("yy= %d\n",yy);
}
int aa;
```

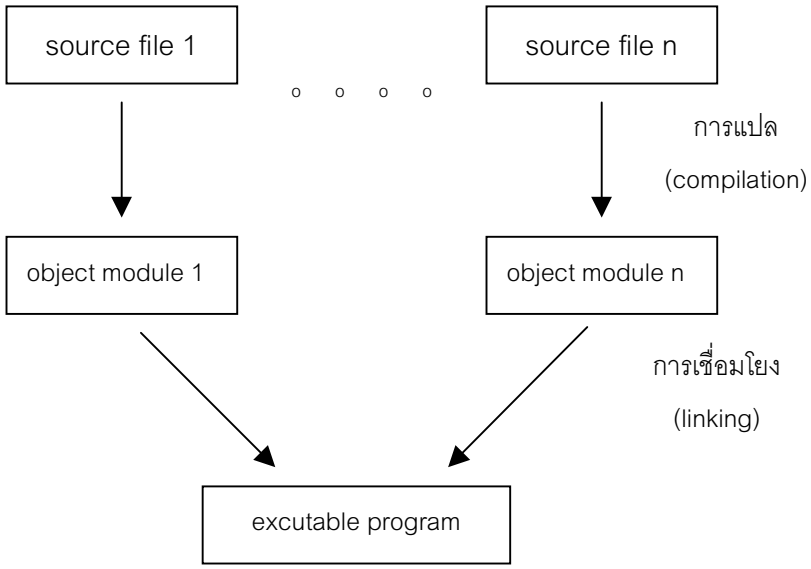
ผลลัพธ์ที่ได้

```
Enter number 1:15
Enter number 2:20
yy= 50
```

ในตัวอย่างที่ 7.4 function1 จะไม่คืนค่าเพราะมีคำสั่ง void ผลจากการทำงานของฟังก์ชันก็คือการพิมพ์ มูลค่าของ yy ออกทางจอภาพ ตัวแปร num2 ที่กำหนดให้เป็น extern จึงถือเสมือนเป็นตัวแปรแบบครอบคลุม สามารถทำงานได้ใน function1 และนำมาใช้ในการคำนวณหาค่า yy ได้

7.4 การสร้างโปรแกรมที่มีความซับซ้อน

การเขียนโปรแกรมภาษา C ที่ได้กล่าวมาแล้วนั้นมีการสร้างโปรแกรมรหัสต้นฉบับ (source code) จากแฟ้มเพียง 1 แฟ้มข้อมูล แต่ในกรณีที่ผู้ใช้มีการเรียกใช้ฟังก์ชันที่เขียนขึ้นจากโปรแกรมอื่น ผู้ใช้ต้องทำการคัดลอกฟังก์ชันเดิมไปไว้ในโปรแกรมใหม่ ในภาษา C นักเขียนโปรแกรมสามารถสร้างโปรแกรมโดยรวบรวมรหัสต้นฉบับจากหลายแฟ้มได้ โดยรหัสต้นฉบับเหล่านั้นจะต้องมีการแปลเป็นโมดูลจุดหมาย (object module) ที่เป็นภาษาเครื่อง ต่อมาจึงนำโมดูลจุดหมายมาเชื่อมโยงกัน โดยใช้โปรแกรม linker เพื่อสร้างเป็นแฟ้มที่กระทำการได้ (executable file)



รูปที่ 7.1 กระบวนการแปลและเชื่อมโยงแฟ้มข้อมูลเพื่อสร้างโปรแกรมกระทำการ

ตัวอย่างที่ 7.5 โปรแกรมแสดงการเขียนข้อมูลที่รับเข้าและเลขหน้า

```
/*
* print lines with automatic page and line numbering
*/
#include <stdio.h>
#define MAXLEN      80          /* longest line */
#define PAGELEN     5           /* page length */
main()
{
        char input [MAXLEN + 1];        /* input line */
        void numline(char *line);
        int getline(char *line, int max);
        while(getline(input, MAXLEN) != -1)
                        numline(input);
        return 0;
} /* End main */
void numline(char *line)
{
        static int pageno =0;                   /* current page number */
        static unsigned long lineno = 0;        /* current line number */
        static int pageline = PAGELEN;          /* lines used on page */
        if (pageline == PAGELEN)                /* new page, write pageno */
        {
                printf("\n*****page Number: %d*****\n", ++pageno);
                pageline = 0;
        } /* End if */
        printf("%6ld %s\n", ++lineno, line);
        pageline++;
} /* void numline */
int getline(char *ptr, int max)
{
        register int c;
        register char *startptr = ptr;
        register char *endptr = startptr + max -1;
        while ((c = getchar ()) != '\n' && c != EOF)
                        if (ptr < endptr)
                                *ptr++ = c;
                *ptr = '\0' ;
        return (c == EOF) ? -1 : ptr - startptr;
} /* int getline  */
```

```
void putline(char *startptr)
{
        register char *ptr = startptr;
        for (; *ptr != '\0'; ptr++)
                putchar(*ptr);
        putchar('\n');
} /* End void putline */
```

ผลลัพธ์ที่ได้

```
***** page Number: 1 *****
     1 Naree
Nipon
     2 Nipon
ponpoj
     3 ponpoj
Jindaporn
     4 Jindaporn
Rattiya
     5 Rattiya
Kaemmawadee

***** page Number: 2 *****
     6 Kaemmawadee
Juejai
     7 Juejai
Wimommas
     8 Wimommas
Tassanee
     9 Tassanee
Pisit
    10 Pisit
```

โปรแกรมนี้ประกอบด้วยฟังก์ชัน 3 ฟังก์ชันคือ

(1) getline เป็นฟังก์ชันสำหรับอ่านข้อมูลนำเข้าทีละ 1 บรรทัด

(2) numline เป็นฟังก์ชันเขียนผลลัพธ์ที่ละ 1 บรรทัด

(3) main เป็นฟังก์ชันที่เรียกใช้ getline และ numline

สำหรับฟังก์ชัน getline อ่านข้อมูลนำเข้าเก็บไว้ในแถวลำดับชื่อ input ส่วน numline เขียน output 1 บรรทัด นับจำนวนบรรทัดและจำนวนหน้าของข้อมูล และเขียนหมายเลขหน้าที่ส่วนบนของ output ในฟังก์ชัน numline นี้ใช้ตัวแปร static ได้แก่ pageno และ lineno เพื่อเก็บค่าของจำนวนหน้าและจำนวนบรรทัด

ตัวอย่างที่ 7.6 โปรแกรมแสดงการสร้างแฟ้มกระทำการ ที่เก็บไว้ในแฟ้มต่างกัน จำนวน 3 แฟ้ม ได้แก่

- pageno.c
- inout.h
- inout.c

```
/*      *pageno.c
        * Print lines with automatic page and line numbering.
*/
#include <stdio.h>
#include "inout.h"
#define MAXLEN      80              /* longest line */
#define PAGELEN     5               /* page length */
main ()
{
        char input [MAXLEN + 1];    /* input line */
        void numline(char *line);
        while(getline(input, MAXLEN) != -1)
                numline(input);
        return 0;
} /* End main */
void numline(char *line)
{
        static int pageno = 0;                  /* current page number */
        static long lineno = 0;                 /* current line number */
        static int pageline = PAGELEN;          /* lines used on page */
        if (pageline == PAGELEN)                /* new page, write pageno */
        {
                printf("\n*****Page Number: %d*****\n", ++pageno);
                pageline = 0;
        } /* End if */
        printf("%6ld %s\n", ++lineno, line);
        pageline++;
} /* End void numline */
```

```
/*
* inout.h -- Prototypes for inout.c:
*       getline - read an input line into a string.
*       putline - new function to write a string.
*/
int getline(char *line, int max);
void putline(char *line);
```

```
/*
* inout.c -- Twc useful I/O functions:
*         getline - read an input line into a string.
*         putline - write a string as output line.
*/
#include <stdio.h>
#include "inout.h"
int getline(char *startptr, int max)
{
        register int c;
        register char *ptr = startptr;
        register char *endptr = startptr + max - 1;
        while ((c = getchar()) != '\n' && c != EOF)
                if (ptr < endptr)
                        *ptr++ = c;
        *ptr = '\0' ;
        return (c ==EOF) ? -1 : ptr - startptr;
} /* End int getline */
void putline(char *startptr)
{
        register char *ptr = startptr;
        for (; *ptr != '\0'; ptr++)
                putchar(*ptr);
        putchar('\n');
} /* End void putline */
```

ผลลัพธ์ที่ได้

```
***** page Number: 1 *****
     1 Naree
Nipon
     2 Nipon
ponpoj
     3 ponpoj
Jindaporn
     4 Jindaporn
Rattiya
     5 Rattiya
Kaemmawadee

***** page Number: 2 *****
     6 Kaemmawadee
Juejai
     7 Juejai
Wimommas
     8 Wimommas
Tassanee
     9 Tassanee
Pisit
    10 Pisit
```

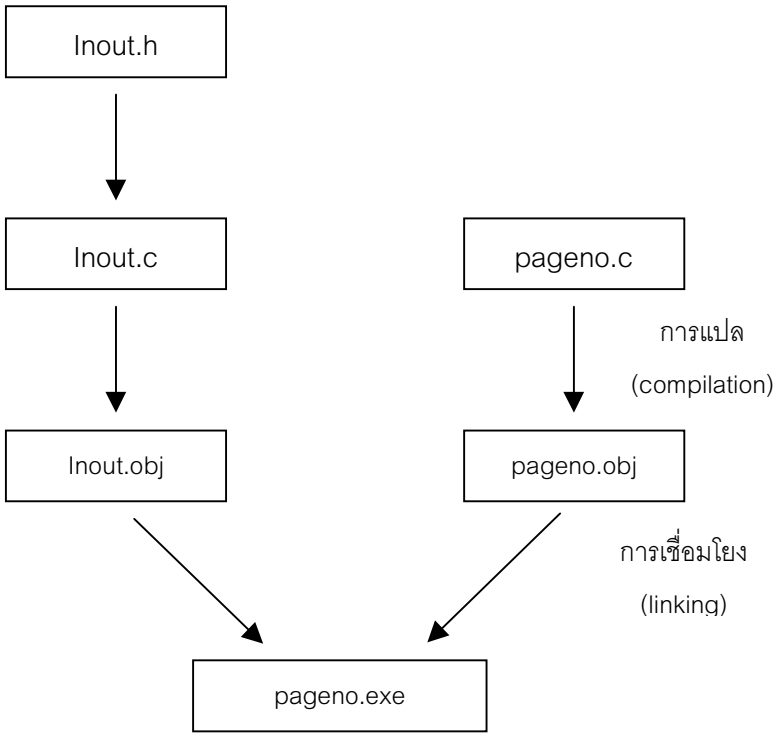
สำหรับโปรแกรมของตัวอย่างนี้มีโปรแกรมหลายโปรแกรมที่เรียกใช้ฟังก์ชัน getline และ putline ผู้ใช้อาจจะแยกโปรแกรมออกเป็นส่วนย่อย เพื่ออำนวยความสะดวกกับโปรแกรมอื่นที่เรียกใช้เฉพาะฟังก์ชันที่ต้องการ เราอาจจะแบ่ง pagenum.c เป็น 3 ส่วนคือ

(1) pageno.c ที่ประกอบด้วย ฟังก์ชัน main และ numline

(2) inout.h ทำหน้าที่เป็นแฟ้มเฮดเดอร์ (header) ประกอบด้วยฟังก์ชัน getline และ putline

(3) inout.c ประกอบด้วยฟังก์ชัน getline และ putline ซึ่งทำหน้าที่อ่านข้อมูลนำเข้าและเขียนสายอักขระ

เมื่อแบ่งโปรแกรมออกเป็นโปรแกรมที่มีขนาดเล็กลงแล้ว ผู้ใช้สามารถแปล (compile) โปรแกรมแล้วเชื่อมโยงกับโปรแกรมอื่น ๆ ได้หลากหลาย การสร้างโปรแกรมเฮดเดอร์ inout.h ขึ้นเพื่อที่ผู้ใช้จะนำโปรแกรมนี้ไปใช้ได้เลยโดยไม่ต้องเขียนต้นแบบของฟังก์ชัน getline และ putline ในโปรแกรมที่สร้างขึ้นใหม่ทุกครั้ง สำหรับ #include ที่อ้างถึงแฟ้มเฮดเดอร์นั้นใช้คำสั่ง #include "inout.h" เขียนที่ส่วนต้นของโปรแกรม inout.c และ pageno.c



รูปที่ 7.2 กระบวนการสร้างโปรแกรม pageno.exe

7.5 การแปลโปรแกรมและเชื่อมโยงโมดูลในเทอร์โบ C

ในเทอร์โบ C ผู้ใช้สามารถแปลโปรแกรมเชื่อมโยงโมดูลเข้าด้วยกันด้วยคำสั่ง tcc ในบรรทัดคำสั่ง(line command)

รูปแบบ tcc fileA.c fileB.c

fileA ชื่อแฟ้มข้อมูลเขียนด้วยภาษา C

fileB ชื่อแฟ้มข้อมูลเขียนด้วยภาษา C

เช่น tcc pageno.c inout.c

ในกรณีนี้จะมีการแปลแฟ้มข้อมูล pageno.c และ inout.c ไปเป็นโมดูลจุดหมายแล้วจึงมีการนำโมดูลนี้มาเชื่อมกันเพื่อสร้างเป็นโปรแกรมกระทำการ โดยโปรแกรมนี้จะใช้ชื่อเดียวกับแฟ้มข้อมูลแรกสุดที่ติดกับคำสั่ง tcc ในกรณีนี้ โปรแกรมกระทำการที่ได้คือ pageno.exe

ถ้าผู้ใช้ต้องการสร้างโปรแกรมกระทำการ ที่มีชื่อแตกต่างไปจากโปรแกรมแรกที่ติดกับคำสั่ง tcc จะใช้ -e แล้วตามด้วยชื่อโปรแกรมดำเนินการที่ต้องการ

รูปแบบ tcc -efileE fileA.c fileB.c

fileE ชื่อโปรแกรมกระทำการ

fileA.c, fileB.c เป็นชื่อโปรแกรมที่เขียนด้วยภาษา C

เช่น tcc -epgno pageno.c inout.c

หลังจากที่มีการแปลโปรแกรม pageno.c และ inout.c เป็นโมดูลจุดหมาย โปรแกรมนี้จะใช้ชื่อเดียวกับโปรแกรมแรกสุด แล้วจะมีการเชื่อมโยงและสร้างโปรแกรมดำเนินการชื่อ pgno.exe ซึ่งสามารถเรียกใช้ได้ต่อไป

แบบฝึกหัด

1. จำนวนเฉพาะ (prime) คือจำนวนเต็มที่หารด้วย 1 และจำนวนนั้นเองลงตัวเท่านั้นเช่น 2, 3, 5, และ 7
   ก. จงเขียนโปรแกรมและฟังก์ชันที่สามารถทดสอบได้ว่าจำนวนเต็มที่กำหนดให้เป็นจำนวนเฉพาะ โดยใช้คำสั่ง extern ในโปรแกรม
   ข. ให้ใช้ฟังก์ชันที่เขียนขึ้นในข้อ ก. เพื่อแจกแจงและพิมพ์จำนวนเฉพาะทั้งหมดที่มีค่าระหว่าง 1 ถึง 10,000 ในจำนวนเต็ม 10,000 จำนวนที่กล่าวไปแล้ว ท่านคิดว่าควรจะทดสอบจำนวนเต็มกี่ตัวที่มีคุณสมบัติเป็นจำนวนเฉพาะฉ
   ค. ท่านอาจจะคิดว่า n/2 เป็นขอบเขตบนของตัวเลขที่จะนำมาทดสอบจำนวนเฉพาะ หรือจะทดสอบจำนวนเฉพาะที่มีค่ามากที่สุดเป็นรากที่สองของ n

   จงให้เหตุผล และเขียนโปรแกรมทดสอบในทั้ง 2 กรณี

   ในกรณีข้อ ค. ให้สร้างโปรแกรมดำเนินการจากแฟ้มข้อมูลของรหัสต้นฉบับหลายแฟ้ม แล้วนำมารวมกัน
2. จงเขียนโปรแกรมที่สร้างแบบจำลองของการโยนเหรียญ ในการโยนเหรียญแต่ละครั้งโปรแกรมจะพิมพ์ 'Head' (หัว) หรือ 'Tail' (ก้อย) โดยให้โปรแกรมมีการโยนเหรียญ 100 ครั้ง นับจำนวนที่ออกหัวหรือก้อย พิมพ์ผลลัพธ์ โปรแกรมนี้จะเรียกใช้ฟังก์ชัน flip ซึ่ง return 0 เมื่อเหรียญหงายหน้าก้อยสำหรับค่า tail และ 1 เมื่อเหรียญหงายหน้าหัวสำหรับค่า head

<u>หมายเหตุ</u>

ให้ใช้ฟังก์ชัน rand() ในการหาเลขสุ่มเพื่อนำไปใช้สร้างแบบจำลอง (Simulate) ของการโยนเหรียญ ถ้าโปรแกรมสร้างแบบจำลองของการโยนเหรียญได้แม่นยำ เมื่อคำนวณจำนวนของหัวและก้อยจะได้จำนวนหน้าใกล้เคียงกันคือ ประมาณหน้าละ 50 ครั้ง

3. จงเขียนโปรแกรมสำหรับเล่นเกม 'ทายตัวเลข' โดยใช้ฟังก์ชันที่ประกอบด้วยตัวแปรครอบคลุมและตัวแปรเฉพาะถิ่น สามารถดำเนินงานได้ดังต่อไปนี้ :

   โปรแกรมเลือกจำนวนเต็มที่มีค่าระหว่าง 1 ถึง 1000 โดยวิธีสุ่มโปรแกรมพิมพ์ข้อความต่อไปนี้ที่จอภาพ

I have a number between 1 and 1000
Can you guess my number?
Please type your selected number.

ผู้ใช้จะพิมพ์จำนวนเต็มที่ทาย โปรแกรมจะตอบด้วยข้อความใดข้อความหนึ่งดังต่อไปนี้

1. Excellent! You are right!
   Would you like to play again (y or n) ?
2. The number is too low. Try again.
3. The number is too high. Try again.

เมื่อผู้เล่นทายถูกโปรแกรมจะพิมพ์ข้อความในข้อ 1

ถ้าผู้เล่นทายผิดโปรแกรมจะวนซ้ำจนกระทั่งผู้เล่นทายคำตอบได้ถูก โปรแกรมจะให้คำใบ้ โดยบอกผู้เล่นว่า 'The number is too low' หรือ 'The number is too high' ในกรณีที่ผู้เล่นใส่ตัวเลขที่มีค่าต่ำหรือค่าสูงกว่าเลขที่กำหนดให้ตามลำดับ

4. จงปรับปรุงโปรแกรมในข้อ 3. โดยโปรแกรมสามารถจะนับจำนวนครั้งที่ผู้เล่นทายคำตอบผิด ถ้าผู้เล่นทายผิดน้อยกว่า 10 ครั้ง โปรแกรมจะพิมพ์ข้อความ 'Either your know the secret or you are lucky!' ถ้าผู้เล่นทายผิด 10 ครั้ง โปรแกรมจะพิมพ์ข้อความ 'Ahah! You know the secret!' ถ้าผู้เล่นทายผิดมากกว่า 10 ครั้ง โปรแกรมจะพิมพ์ข้อความ 'You should be able to do better'

   โปรแกรมในข้อนี้เขียนโดยใช้ฟังก์ชัน และคำสั่ง extern

หมายเหตุ : ในกรณีนี้ผู้เล่นควรจะทายถูกเมื่อทายไม่เกิน 10 ครั้ง เพราะในแต่ละครั้งที่เขาทายนั้นเขาสามารถลดจำนวนตัวเลขที่จะทายได้ทีละครึ่ง ท่านสามารถหาคำตอบได้หรือไม่ว่าทำไม ในกรณีจำนวนเต็ม 1 ถึง 1000 นั้นผู้เล่นสามารถทายผิดไม่เกิน 10 ครั้ง