1031 計 算 機 程 式 設 計 期 中 考

姓名:_____學號:_____ 103/11/10 (一)

Time：**14:30 – 16:20**

考試規則：1. **請闔上課本**，**不可**參考任何文件包括小考、作業、實習、或是其他參考資料
2. 你可以在題目卷上直接回答，但是請在答案卷上註明題號
3. 你覺得需要的話，可以使用沒有教過的語法，但是僅限於 **C 語言**
4. **不可**使用電腦、平板、智慧手機、及工程型計算機
5. 請不要左顧右盼！請勿討論！請勿交換任何資料！對於題目說明有任何疑問請舉手發問
6. 如果你提早交卷，請迅速安靜離開教室
7. 違反上述考試規則視為不誠實的行為，交由學校依學務規章處理
8. 請在題目卷及答案卷上**寫下姓名及學號**，交卷時請繳交**題目卷**及**答案卷**
9. 本份考卷共有 **143** 分的題目，因為*程式設計*這種實際使用的東西很多地方很瑣碎，不像其它抽象化以後的學科，在這份考卷裡並不要求你各種寫法都很熟練，也不要求你在有限的時間裡把熟悉或是不熟悉的題目通通寫完，所以給你各種形式的題目和比較多的分數，但是你得到的成績**超過 100 的部份是不算在學期成績裡的**

1. [6] 請問 C 語法中變數與函數的命名規則為何？由容易閱讀、容易修改的程式角度來看，變數與函數的命名規則為何？

Sol:

基本上可以使用英數字 a~z, A~Z, 0~9,或是底線_
不可以使用其他的標點符號或是空白
變數與函數名稱的起始字元不可以是數字 0~9
不可以使用保留字(關鍵字)

變數名稱應該代表其存放資料的意義 (例如 numberOfStudents, numberOfData, …)
函數名稱應該要代表它所抽象化的程序步驟 (例如: calculateInterest, remove_data, read_file, …)

2. [6] 請簡述 C 語言中主要有哪些種類的流程控制敘述？

Sol:

循序式 (sequential): 所有的控制結構都是由上往下一個一個執行的
選擇式 (selection): 例如 if 敘述, switch 敘述
重覆式 (repetition): 例如 while 敘述, for 敘述，以及 do while 敘述

3. [6] 請簡單說明為什麼說使用 C 語言寫程式時是由上而下(top-down)的設計方法？

Sol:

我們在設計一個程式來處理資料時，常常可以把處理方法分成幾個段落，或是幾個階段，一部份一部份來做，如果某一階段的動作很複雜，通常還會再進一部分成幾個細部的動作，如此在每一個函式裡我們只處理同樣細節層次的步驟，一層一層地由整體功能逐步切割到細部功能的設計方法稱為由上而下的程式設計方法。

這是運用任何“程序化程式語言”(C, Pascal, Fortran, Basic, … 時一般的程式設計方法，由於程序化的程式語言基本上就是目前計算機的控制語言比較高階的版本，所以只要是計算機做得到的功能其實都可以用程序化的程式語言完成，差別只在於有沒有效率，是不是比較容易偵錯，是不是很容易看得懂，設計起來是不是很簡便而已，例如我們現在知道圖形化的視窗界面系統應該用物件導向的方式來設計會比較容易，所以我們會用物件導向的程式語言來設計，例如 (SmallTalk, C++, Objective-C, Java, 等等語言)，可是在開始設計視窗界面系統時卻都還是使用 C 語言來設計的，例如 X-Window, MS Windows 3.1/95/98/ME 等等。

4. [5] 請問在寫 C 程式時，如果設計時發現需要一個變數 day 來存放 1 到 366 之間的整數數值代表今天是由 1 月 1 日算過來第幾天，可以有哪些型態的選擇？如果程式往下執行時另外需要計

算此日期的月份，在算出來以後直接記錄在這個 day 變數裡，請問這樣的作法的好處與壞處?

Sol:

short 或 int 或 long (或是 long long, _int64)

好處: 省掉另外定義一個變數，少使用一些記憶體的空間

壞處: 變數的名稱在設計、閱讀和修改程式時很重要的依據，通常就直接代表存放在變數裡面資料的意義，如此會使得以後閱讀程式的人容易誤會整段程式的意義

5. [5] 請問下列程式片段執行時螢幕上會列印出什麼資料?

```
int main() {
        char a;
        for (a='P'; a<'Z'; a+=2)
                printf("%c", a+'z'-'Z');
        system("pause");
        return 0;
}
```

Sol: prtvx

6. [10] 請將下列 switch 敘述以 if/else 敘述改寫:

```
switch (i) {
case 0:
case 1:
        n = 10;
        break;
case 2:
        n = 100;
        break;
default:
        n = 1;
}
```

Sol:

```
if ((i==0)||(i==1))
        n = 10;
else if (i==2)
        n = 100;
else
        n = 1;
```

7. [8] 請問在 C 語言中, #define SIZE 300 和 const int SIZE=300; 個別使用的特性為何? 有什麼差別或是各有什麼好處壞處?

Sol:

#define SIZE 300 是一個前處理器的指令，用途是在編譯程式之前把程式裡面所有出現 SIZE 的字串取代為 300，必須注意 300 之後不可以有分號 ';'，有的時候會發生取代錯誤的現象

const int SIZE=300; 是定義一個整數變數，只是它的內容只能在定義的時候設定一次，以後不可以更改; 最主要的差別如下:

1. const int SIZE=300; 是一個變數，有型態，有記憶體位址
2. 在偵錯器(debugger)中可以看到其數值
3. 比較舊的編譯器不接受 const int SIZE=300; int x[SIZE]; 這樣子的敘述，只能用 #define

8. [20] 請指出下列程式中各個識別字在程式的什麼地方是可以使用的以及各個變數的生命週期

```
01 char bob;
```

```
02 int fun1(int);
03 int main() {
04      int fred, george;
05      int fun2();
06      if ((fred=fun2()) == 0) {
07          for (fred=0; fred<10; fred++) {
08              double george;
09              …
10          }
11          fun2();
12      }
13      fun1();
14 }
15 int fun1(char param) {
16      int fred;
17      ...
18 }
19 int fun2() {
20      static int carol;
21 …
22 }
```

變數

列號	名稱	可使用範圍, 生命週期
16	fred	16-18 每次 fun1()被呼叫時執行到 16 列產生, 執行到第 18 時列結束
01	bob	01-22 程式開始執行 main()的第一列 04 列前生成, main()結束-14 列後毀滅
04	fred	04-14 04 列定義時產生, 14 列函數結束時毀滅 正常情況下 main()函數只會執行一次,
04	george	04-07,11-14 與 04 fred 同
08	george	08-10 程式每執行迴圈內容到 08 列定義時產生 第 10 列迴圈內容執行一次完畢時毀滅
15	param	15-18 fun1 函數被呼叫到時此變數會定義出來, 18 列函數執行完畢時毀滅
20	carol	20-22 程式開始執行 main()的第一列 04 列前生成, main()結束-14 列後毀滅

函數名稱

列號	名稱	可使用範圍
02,15	fun1	02-22
03	main	03-22
05,19	fun2	05-14, 19-22

9. [20] 課本上這個計算 x^y (例如:$1.5^{1234567}$)的程式, 其中 y 為一個整數, 執行過程中需要計算 y-1 次的乘法, 是一個很沒有效率的程式, 請以迴圈改寫它成為一個乘法次數正比於 log y 的程式?

```
double power(double x, int y) { // assume
y>=0
    int i;
    double pow = 1.0;
    for (i=1; i<=y; i++)
        pow *= x;
    return pow;
}
```

Sol:

```
double power(double x, int y) {
    double basepower = x, pow = 1.0;
    while (y>0) {
        if (y%2) pow *= basepower;
        y /= 2;
        basepower *= basepower;
    }
    return pow;
}
```

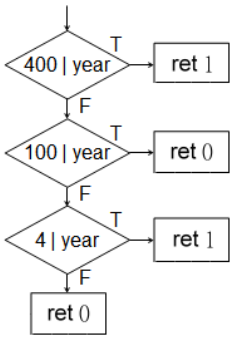
請注意這兩個程式沒有認真考量所計算答案的大小, 對於|x|<1 是沒有問題的, 但是|x|>1 的時候會很快就超過浮點數可以表達的範圍, 一種方法是很快運用對數計算出 y 的上限, 避免存放在變數 pow 或是 basepower 裡面的數值超過範圍, 另外一種方式則是每次計算前後檢查

10. [20] 請依照下列要求寫一個程式輸入民國年/月/日 (例如 103/11/10), 輸出該日期星期幾

i. 請寫一個函數 int isLeap(int year) 判斷是不是閏年，如果是的話回傳 1 否則回傳 0
請先繪製此函數判斷邏輯的流程圖，判斷邏輯如下:
閏年的西元年份是 4 的倍數但不是 100 的倍數，或是 400 的倍數

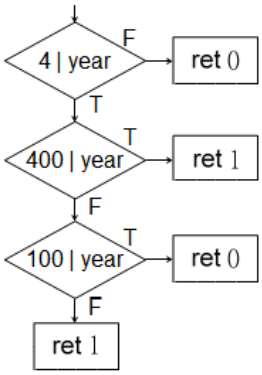
Sol: 請注意流程圖各個元件的畫法有一般的規範，元件內的寫法沒有特別的規範

```
int isLeap(int);
…
int isLeap(int year) {
    if (year % 400 == 0)
        return 1;
    else if (year % 100 == 0)
        return 0;
    else if (year % 4 == 0)
        return 1;
    else
        return 0;
}
```



或

```
int isLeap(int year) {
    if (year % 4 != 0)
        return 0;
    else if (year % 400 == 0)
        return 1;
    else if (year % 100 == 0)
        return 0;
    else
        return 1;
}
```



ii. 請寫一個函數 int weekdayOfJanFirst(int year) 計算每年的第一天是星期幾: (請藉由西元元年 1 月 1 日為星期一來計算)

Sol:

```
int weekdayOfJanFirst(int year) { // year >= 1
   return (year-1 +          // each common year has 365 days, 365%7=1
           (year-1)/4+(year-1)/400-(year-1)/100 + // each leap year has 366 days, 366%7=2
           1) % 7;           // 0001/01/01 is Monday
}
```

或

```
int weekdayOfJanFirst(int year) { // year >= 1
    int i, nLeapYears=0;
    for (i=1; i<year; i++)
        nLeapYears += isLeap(i);
   return (year-1 +           // each common year has 365 days, 365%7=1
           nLeapYears +    // each leap year has 366 days, 366%7=2
           1) % 7;           // 0001/01/01 is Monday, Sunday:0, Monday:1, Tuesday:2 …
}
```

iii 請寫一個函數 int dayOfYear(int year, int month, int day)計算輸入的月日是該年的第幾天?

Sol:

```
int dayOfYear(int year, int month, int day) {
    int i;;
```

```
        for (i=1; i<month; i++)
            if (i==1||i==3||i==5||i==7||i==8||i==10)
                day += 31;
            else if (i==2)
                day += 28 + isLeap(year);
            else // 4, 6, 9, 11
                day += 30;
        return day;
    }
```

或

```
    int dayOfYear(int year, int month, int day) {
        int leap = isLeap(year);
        switch (month) {
        case 1:
            return day;
        case 2:
            return day+31;
        case 3:
            return day+31+28+leap;
        case 4:
            return day+2*31+28+leap;
        case 5:
            return day+2*31+28+leap+30;
        case 6:
            return day+3*31+28+leap+30;
        case 7:
            return day+3*31+28+leap+2*30;
        case 8:
            return day+4*31+28+leap+2*30;
        case 9:
            return day+5*31+28+leap+2*30;
        case 10:
            return day+5*31+28+leap+3*30;
        case 11:
            return day+6*31+28+leap+3*30;
        case 12:
            return day+6*31+28+leap+4*30;
        }
    }
```

或

```
    int dayOfYear(int year, int month, int day) {
        int startDayOfMonth[] = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334};
        return day + startDayOfMonth[month]
                   + month>2?isLeap(year):0;
    }
```

iv 運用上述函數完成程式需求 (輸出時以數字 1~7 代表星期一到星期日)

Sol:

```
    int main(void) {
        int weekday, year, month, day;
        printf("\n 請輸入日期  民國年/月/日: ");
```

```
        scanf("%d/%d/%d",&year,&month,&day);
        if ((month<1)||(month>12)) {
            printf("月份輸入錯誤!\n");
            return 1;
        }
        else if ((day<1) ||
                 (( (month==1)||(month==3)||(month==5)||
                    (month==7)||(month==8)||(month==10)||
                    (month==12))&&(day>31)) ||
                 (( (month==4)||(month==6)||(month==9)||
                    (month==11))&&(day>30)) ||
                    ((month==2)&&isLeap(year)&&(day>29)) ||
                    ((month==2)&&!isLeap(year)&&(day>28))) {
            printf("日期輸入錯誤!\n");
            return 1;
        }
        weekday = (weekdayOfJanFirst(year+1911) +
                    dayOfYear(year+1911, month, day)-1) % 7;
        if (weekday == 0) weekday = 7;
        printf("民國%d 年%d 月%d 日為星期%d\n", year, month, day, weekday);

        system("pause");
        return 0;
    }
```

或是

```
    …
    weekday = (weekdayOfJanFirst(year+1911) +
                dayOfYear(year+1911, month, day)-1-1) % 7 + 1;
    printf("民國%d 年%d 月%d 日為星期%d\n", year, month, day, weekday);
    …
```

11. [10] 請舉例說明下列函式的功能?

```
int function(int param1, int param2) {
    return param1 - (param2 * (param1/param2));
}
```

請問哪一個預設的算術運算子可以直接由 param1 及 param2 兩個整數變數的內容計算這個結果 (請運用它改寫這個函式, 並且重新命名此函式以及變數)

Sol:

例如 param1 = 123, param2 = 10, 函數計算結果為 3

```
int remainder(int dividend, int divisor) {
    return dividend % divisor;
}
```

12. a) [3] 請問下列程式片段列印出來的結果是什麼

```
    …
    void f(int x) {
        x = x + 1;
    }
    int main(void) {
        int x=5;
```

```
    f(x);
    printf("x=%d", x);
    …
}
```

Sol: 5

b) [3] 如果希望修改上面的程式裡的 void f(int) 函數，以及其呼叫敘述 f(x)，使得上面的 printf("%d", x); 敘述能夠印出 x=6，該如何修改?

Sol:

```
void f(int *x) {
    *x = *x + 1; // (*x)++; or *x += 1;
}
…
f(&x)
```

13.
```
void func(int a) {
    int b=0;
    while (1) {
        b = b * 100 + a % 100;
        a /= 100;
        if (a == 0) break;
    }
    printf("%d\n", b);
}
```

a) [3] 請問當以 func(1234567); 呼叫時，在螢幕上列印的數字為何?

Sol: 67452301

b) [3] 請問以 func(1234567); 呼叫時, while 迴圈內部執行了幾次?

Sol: 4

c) [3] 請以 for 迴圈語法改寫此函數，請不要使用 break 敘述?

Sol:
```
void func(int a) {
    int b;
    for (b=0; a!=0; a/=100)
        b = b * 100 + a % 100;
    printf("%d\n", b);
}
```

d) [3] 請問當以 func(-1234); 呼叫時在螢幕上列印的數字為何?

Sol: -3412

14. 請以小括號標註下列運算式實際運算時的順序，例如 a + b * c / d 請標示 a + ((b * c) / d)

a) [5] d = a + 5 && d < 100 || d == 100

Sol:

```
d = ( ( (a + 5) && (d < 100) ) || (d == 100))
```

b) [2] (double) a / 5 + 2

Sol:

(((double)a)/5) + 2

c) [2] a + - d ++;

Sol:

a + (-(d++)); ++ 這個運算子只能作用在變數上

相關運算子的優先順序與結合性如下

優先順序	運算子	類別	結合性
	…		
2	!、+（正號）、-（負號）	一元運算子	由右至左
	…		
2	++、--、(型態)	遞增/遞減、型態轉換運算子	由右至左
3	*、/、%	算數運算子	由左至右
4	+、-	算數運算子	由左至右
6	>、>=、<、<=	關係運算子	由左至右
7	==、!=	關係運算子	由左至右
	…		
11	&&	邏輯運算子	由左至右
12	||	邏輯運算子	由左至右
	…		
14	=、*=、/=、%=	設定運算子	由右至左