Lab01

1 Task 1: Environment

编译一个 Helloworld 程序，打印 Hello, world!\n

一些 GCC TIP

gcc <file.c> -o <output> 编译并且链接

gcc -E <file.c> -o <output.i> 预处理，不编译

gcc -S <file.i> 预处理转汇编

gcc -c <file.c> -o <output.o> 编译但是不链接

objdump -t <file.o> > output.t 输出符号表

objdump -S -d <file.o> > output.s 输出源代码和汇编的对照

gcc <file1.o> <file2.o> -o <output> 手动链接

```c
#include <stdio.h>
int main()
{
    printf("hello, world!\n");
    return 0;
}
```

---

2 Task 2: *A+B*

2.1 Most used type

Type	Format	Declaration
char	%c	char x=’a’;
int	%d or %i	int x=1;
(base 8)	%o	int x=03;
(base 16)	%x	int x=0x123;
unsigned int	%u	unsigned int x=2;
float	%f	float x=1.25
double	%lf	double x=1.25 (scanf)

2.2 Reading and Writing Integers

The number associated with the format in which you read and write

---

```c
int main(int argc, char** argv)
   {
      unsigned int u;
      printf ("Input a number\n") ;
      scanf("%u", &u);
      printf ("In base 10: %u\n", u);
      printf ("In base 10: %d\n", u);
      printf ("Input a number\n") ;
      scanf("%o",&u);
      printf ("In base 8: %o\n", u);
      printf ("In base 10: %d\n", u);
      printf ("Input a number\n") ;
      scanf("%x",&u);
      printf ("In base 16: %x\n", u);
      printf ("In base 10: %d\n", u);

      /** Normally exiting the program, return 0 **/
      return 0;
   }
```

---

then input 10
record the results.

2.3 Formatted Input/Output

scanf("%c %f", &type, &temp)

scanf is controlled by the conversion specification in the format string (in "…") starting from left to right.

When called, it tries to locate an item of the appropriate type (eg. %d, %f) in the input data, **skipping white-space characters**

- the space,
- horizontal and vertical tab,
- form-feed,
- and new-line character

Pay very attention to "%c"

White-space characters:

- one white-space character in the format string will match any number of white-space character in the input.

Other characters:

- when it encounters a non-white-space character in a format string, scanf compares it with the next input character.
- If the two characters match, scanf discards the input character and continues processing the format string.

- Else, scanf puts the offending character back into the input, then aborts without further processing.
- **test:**

  %d/%d will match _5/_96, but not _5_/_96

  %d_/%d will match _5_/_96

  Which _ represents the space

当在表达式里出现一个非空白字符，scanf 会去和下一个输入字符比较：
如果相同，往下继续；
如果不同，把读取字符放回输入流，放弃继续处理。

```c
int main(int argc, char** argv)
{
    //!
    int a, b;
    scanf("%d/%d",&a, &b);
    //! try to input "_5/_96" and "_5_/_96"
    //! record the redults
    printf("a and b are: %d, %d", a, b);

    scanf("%d /%d",&a, &b);
    //! try to input "_5/_96" and "_5_/_96"
    //! record the redults
    printf("a and b are: %d, %d", a, b);

    /** Normally exiting the program, return 0 **/
    return 0;
}
```

2.4 设计一段程序，从键盘读入 *A*、*B* 两个数。计算 *A*+*B* 的值并输出

描述：
A、*B* 均为整数
提示：
使用 scanf("%d %d", &A, &B);可以从键盘读出**用空格分割的**两个整数
使用 printf("%d", A+B);可以输出计算的结果
尝试普通案例之后，试一下输入 a = 2147483646，b = 1, 2, 3 的结果
演示：

```c
int main(int argc, char** argv)
{
```

```
        /** Define two integer variables **/
        int a, b;

        /** Use scanf to get values from stdin
         *
         * - scanf() read keyboard input
         * - "%d" is the pattern for integer input
         * - &a represent the address of variable a, scanf() modifies a via this
address
        **/
        printf("please input two int numbers:\n");
        scanf("%d %d", &a, &b);

        /** Use printf to put values to stdout
         *
         * - printf() prints
         * - "%d" is the pattern for integer output
         * - a + b is the value of answer, printf use it to format output
        **/
         // Try a = 2147483646, b = 1, 2, 3. See what happened.
        printf("a + b = %d\n", a + b);

        /** Normally exiting the program, return 0 **/
        return 0;
    }
```

---

3 Data Type in C

3.1 C has the following simple data types:

Data type	C code	Size in bytes	Range
Char or signed char	char	1	-128 to 127
Unsigned char	unsigned char	1	0 to 255
int or signed int	int	at least 2	-32768 to 32767
Unsigned int	unsigned int	at least 2	0 to 65535
Short int or signed short int	short	2	-32768 to 32767

Unsigned short int	unsigned short	2	0 to 65535
Long int or Signed long int	long	at least 4	-2147483648 to 2147483647
Unsigned long int	unsigned long	at least 4	0 to 4294967295
float	float	4	3.4E-38 to 3.4E+38
double	double	8	1.7E-308 to 1.7E+308
Long double	long double	10*	3.4E-4932 to 1.1E+4932

Check the size of the following data types.
char
unsigned char
int
unsigned int
short
unsigned short
long
unsigned long
float
double
long double

```
int main(int argc, char** argv)
   {
      //!
      int b = 3;
      //! print out the value of b;
      printf("the size of int is %lu\n", sizeof(b));

      /** Normally exiting the program, return 0 **/
      return 0;
   }
```

3.2 Unsigned Encodings

$$
\begin{aligned}
B2U_4([0001]) &= 0\cdot 2^3+0\cdot 2^2+0\cdot 2^1+1\cdot 2^0 = 0+0+0+1 = 1\\
B2U_4([0101]) &= 0\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0 = 0+4+0+1 = 5\\
B2U_4([1011]) &= 1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+1\cdot 2^0 = 8+0+2+1 = 11\\
B2U_4([1111]) &= 1\cdot 2^3+1\cdot 2^2+1\cdot 2^1+1\cdot 2^0 = 8+4+2+1 = 15
\end{aligned}
$$

Principle: Definition of unsigned encoding

For vector $\vec{x} = [x_{w-1}, x_{w-2}, \ldots, x_0]$

$$B2U_w(\vec{x}) \equiv \sum_{i=0}^{w-1} x_i 2^i$$

B2U: Binary to Unsigned. Thus, the function $B2U_w$ can be defined as a mapping

$B2U_w$: $\{0,1\}_w \rightarrow \{0, \ldots, 2^w - 1\}$.

3.3 Two's-Complement Encodings for signed numbers （补码）

$$
\begin{aligned}
B2T_4([0001]) &= -0\cdot 2^3+0\cdot 2^2+0\cdot 2^1+1\cdot 2^0 = 0+0+0+1 = 1\\
B2T_4([0101]) &= -0\cdot 2^3+1\cdot 2^2+0\cdot 2^1+1\cdot 2^0 = 0+4+0+1 = 5\\
B2T_4([1011]) &= -1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+1\cdot 2^0 = -8+0+2+1 = -5\\
B2T_4([1111]) &= -1\cdot 2^3+1\cdot 2^2+1\cdot 2^1+1\cdot 2^0 = -8+4+2+1 = -1
\end{aligned}
$$

Principle: Definition of two's-complement encoding

For vector $\vec{x} = [x_{w-1}, x_{w-2}, \ldots, x_0]$

$$B2T_w(\vec{x}) \equiv -x_{w-1}2^{w-1} + \sum_{i=0}^{w-2} x_i 2^i$$

B2T: Binary to Two's-Complement. Thus, the function $B2T_w$ can be defined as a mapping $B2T_w$: $\{0,1\}_w \rightarrow \{-2^{w-1}, \ldots, 2^{w-1} - 1\}$.

Note:
int8 最大值： 0111 1111 = 2^{w-1} -1
int8 最小值： 1000 0000= -2^{w-1}

Recall: typical ranges for C integral data types for 32-bit and 64-bit programs

C data type	Minimum	32-bit	Maximum		Minimum	64-bit	Maximum
[signed] char	−128		127		−128		127
unsigned char	0		255		0		255
short	−32,768		32,767		−32,768		32,767
unsigned short	0		65,535		0		65,535
int	−2,147,483,648		2,147,483,647		−2,147,483,648		2,147,483,647
unsigned	0		4,294,967,295		0		4,294,967,295
long	−2,147,483,648		2,147,483,647		−9,223,372,036,854,775,808		9,223,372,036,854,775,807
unsigned long	0		4,294,967,295		0		18,446,744,073,709,551,615
int32_t	−2,147,483,648		2,147,483,647		−2,147,483,648		2,147,483,647
uint32_t	0		4,294,967,295		0		4,294,967,295
int64_t	−9,223,372,036,854,775,808		9,223,372,036,854,775,807		−9,223,372,036,854,775,808		9,223,372,036,854,775,807
uint64_t	0		18,446,744,073,709,551,615		0		18,446,744,073,709,551,615

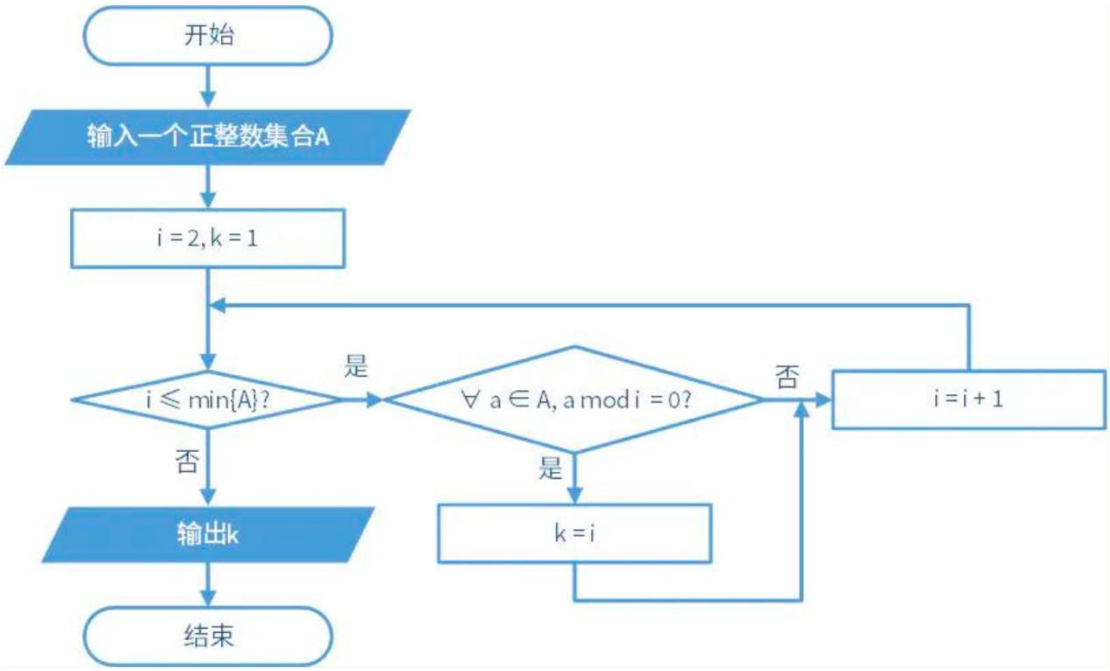
```c
/** @note Include stdio.h to use printf and scanf**/
    #include <stdio.h>
    // ######################################
    // 获得 int 型最大最小值 v2：直接使用 stdint.h 中的宏
    #include <stdint.h>

    int get_max_int_v2()
    {
        return INT32_MAX;
    }

    int get_min_int_v2()
    {
        return INT32_MIN;
    }

    int main(int argc, char** argv)
    {
        //TODO : print out the value;

        //TODO : print out the value;

        /** Normally exiting the program, return 0 **/
        return 0;
    }
```

---

4 最大公约数 GCD (A, B)

4.1 短除法

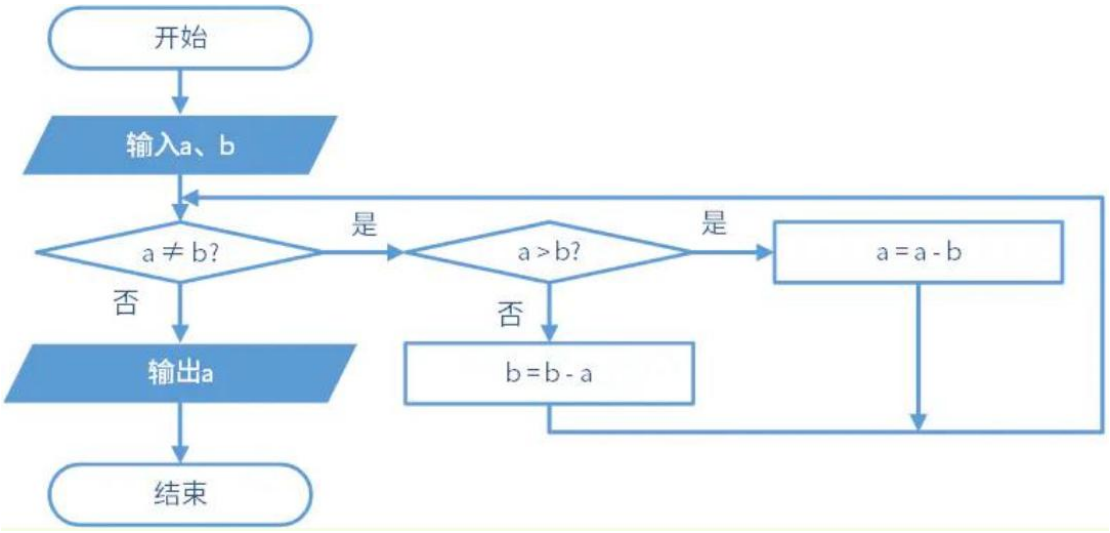
根据下述流程框图，实现 GCD (A, B)



```
int gcd1(int a, int b) {
       int i = 2;
       int k = 1;

       while (i <= a && i <= b) {
           // TODO
       }
       return k;
   }
```

4.2 更相减损术

根据下述流程框图，实现 GCD (A, B)

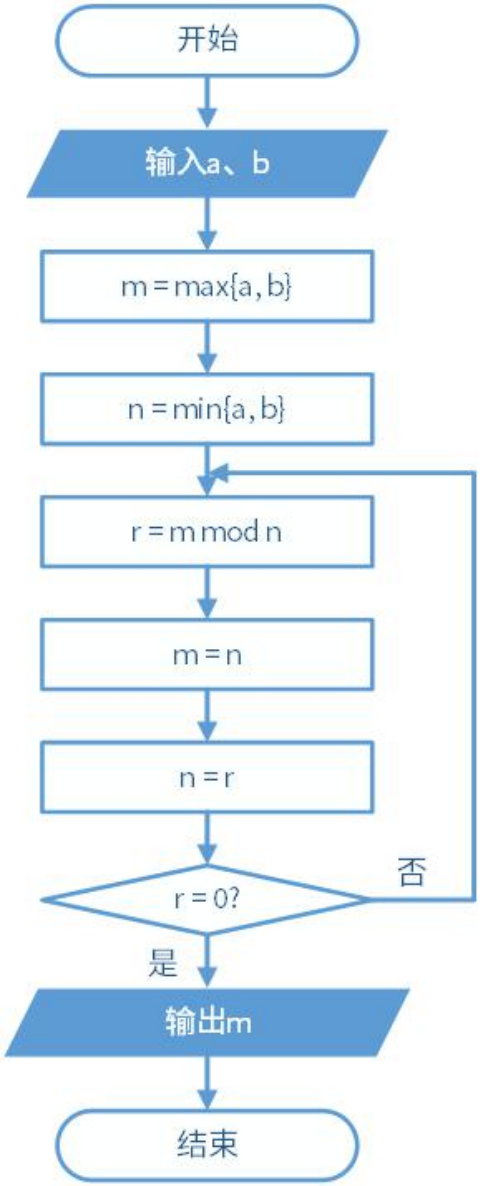


```
int gcd2(int a, int b) {
       while (a!=b){
            // TODO
        }
       return a;
   }
```

4.3 辗转相除法

根据下述流程，实现 GCD (A, B)

- 首先用两数中交大数除以较小数，求得数；
- 再用较小数和余数按上述操作进行相除；
- 直到余数为 0，此时的除数即为最大公约数。



```
int gcd3(int a, int b) {
       int m = 0;
       int n = 0;
       int r = a;

       if(a > b){
          // TODO
       }
       else {
          // TODO
       }

       while(r != 0){
```

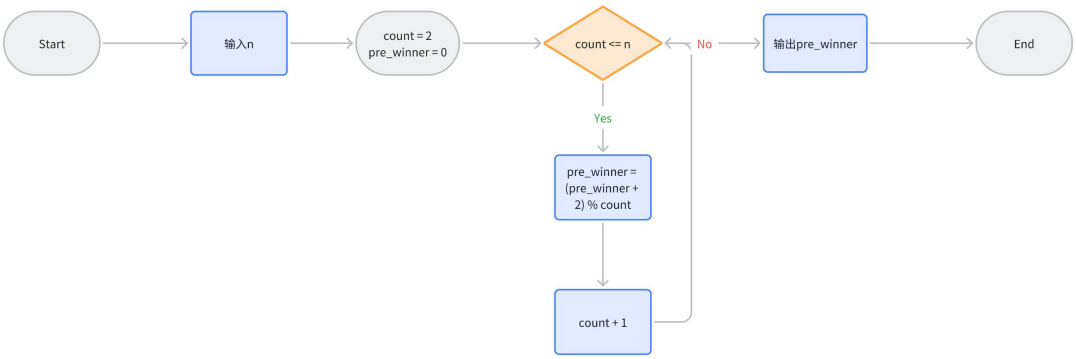
```
            // TODO
        }
        return m;
    }
```

---

5 Homework：Josephus 问题

有 *n* 个人排成一圈，按顺时针方向依次编号 0，1，2,…, *n-1*。从编号为 0 的人开始顺时针“一二”报数，报到 2 的人退出圈子。这样不断循环下去，圈子里的人将不断减少，最终一定会剩下一个人。编写一段程序，输入 *n*,输出最后剩下的人的编号。

例如，如果有 5 个人，那么依次是编号为 1，3，0，4 的退出，最后剩下编号为 2 的。



要求:

（1）提供源代码（按照 canvas 上 Homework submission format 形式）

（2）提供实验报告，包括 n=10~20 的结果统计表，以及对该流程图算法的理解。

（3）尝试使用数组/链表等进行实现，不需要按照流程图。（Bonus）