- 第三周
  - 课堂内容（基本知识点）：
  - 课本补充：
- 第四周
  - 课堂
  - 作业
- 第五周
  - 课堂
  - 作业
  - 课本
- 第六周
  - 课堂
  - 作业
- 第七周
  - 课堂
- 第八周
  - 课堂
  - 课本
- 第九周
  - 课堂
  - 作业
  - 课本
- 第十周
  - 课堂
  - 作业
- 第十一周
  - 课堂
  - 作业
- 第十二周
  - 课堂
  - 作业
- 第十三周
  - 课堂
- 第十四周
  - 课堂
  - 课本
- 第十五周

- 课堂
- 作业
- 课本

第三周

课堂内容（基本知识点）：

- 注释（ //，/**/）
- 预处理指令（#.....）
- 主函数main：唯一，程序的“入口”
- 语句（statement）
- 代码风格！！！

演示程序：猜数字

- 随机数生成：srand与rand组合使用(stdlib.h)，srand用于初始化随机数生成器（**void srand(unsigned int seed)** ），参数为“种子”（seed)，rand根据种子来生成一个数（**int rand(void)**）；若rand前没有srand，视为srand(1)。为了生成随机数，需要“种子”随机，常选择利用时间来产生种子，即 srand(time(NULL)); 其中time函数（声明：time_t time(time_t *arg ）; <time.h>; 其中time_t是无符号整数类型，参数为指向无符号整型的指针)根据当前的时间返回一个无符号整数。可以发现这样的随机数实际上是伪随机。

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(){
    srand(time(NULL));
    int a = rand() % 100 + 1;
    printf("%d",a);
    return 0;
}
//为了生成一定范围内的随机数，常利用取模来控制范围。如生成1-100的随机数
```

课本补充：

程序运行需要经过的步骤：

1. 预处理：c源文件被交给预处理器，执行#开头的预处理指令
2. 编译：预处理后的程序进入编译器，翻译为机器指令

3. 链接：链接器把由编译器产生的目标代码和其它附加代码（如库函数）整合到一起，产生可执行程序

GCC：最流行的C编译器之一，随LInux发行。常用命令：gcc a.c b.c -o out，表示编译、链接a.c和b.c两个源文件（注意：不需要包含头文件，如b.h），生成名为out的可执行程序；然后再在当前目录下输入.\out即可运行out程序

MinGW（Minimalist GNU on Windows）：在Windows平台上模拟了Linux下GCC的开发环境

集成开发环境（IDE），提供一个能够进行代码编辑、编译、链接、执行、调试等一系列工作的平台，如clion, devcpp……。（它也是个软件，建立在命令行之上）

第四周

课堂

1.数据类型：int, double, char, 字符串（字符数组）;

2.输入输出：printf, scanf; scanf需要&（因为参数是指针）（字符串不需要&，本身就是指针）

3.对常量可加const保护；

4.转换说明与转义序列

- 转换说明：格式为“%m.pX”(或“%-m.pX”)，m、p为可选项，为整型常量；X为字母。如%.2f为保留两位小数。具体规则略。
- 转义序列：如"\n", \可以让程序包含一些特殊字符。

5.运算符与优先级

作业

- 运算时的类型。如int / int还是int（故 9 / 4 == 2），可以 9 * 1.0 / 4 来解决。
- <math.h>库函数：pow指数计算，pow(3，4) == 3 * 3 * 3 * 3 == 81; sqrt开平方，sqrt(9) == 3；cbrt()开立方，cbrt(27) == 3；abs计算整数的绝对值，fabs计算浮点数的绝对值。注意它们参数的范围！！！

第五周

课堂

- 选择（if，if……else if……else，switch，switch注意每个分支后跟break）、循环（for，while，do……while），数组
- 选择语句尽可能简化、合并相同情况，不要多层嵌套。
- i++与++i：均可实现i的自增，但是前者的值为i，后者为i+1。
- 对于数组，常常定义宏来表示数组大小，如#define LEN 10，便于后期修改和维护。
- 三目运算符：表达式1？表达式2：表达式3，即如果1成立则执行2，否则执行3。
- 数组注意不要越界，可以适当开大几个元素来防止出现越界问题
- 慎用可变长数组

作业

- 注意数据范围
- 桶计数法：用来计数x个范围为1 ~ n的数字中，每个数字出现了多少次。所有数据里最大值是多少（如n），就开一个多大的数组（或n+1,更方便一些），出现了某个数字i，那就让a[i]++表示i这个数字出现了一次。计数完后来个for循环从a[1]到a[n]来读取记录的出现次数。
- 字符串的读取（重要！！）：scanf("%s",a)会跳过开头的空白符，从第一个非空白符开始读入，直到遇到空白符（空格、回车、制表符等）就停止读入。getchar读入单个字符，不管是空格还是回车。gets读入字符串，直到遇到回车停止，并把回车放在远处（所以往往需要一个getchar再把它读掉）。
- 区分等号(==)和赋值（=）！！！

课本

- 布尔类型，布尔变量。
- 逗号运算符，最后一个表达式的值为整个逗号表达式的值。
- break与continue
- goto(好用，但不建议用，尤其是不要多用，会大大降低代码可读性与可维护性)，需要标签（label）指示跳转到哪里。

第六周

课堂

- 简单算法：二分查找，数字位数判断，素数判断
- 选择排序（重要！）对于n个数，一共需要比较1+2+3+……+（n-1）=n * (n - 1) / 2次。二重循环，对第i个数字和第i+1、i+2……、第n个数分别比较，如果第i个数字比后面的大就交换，每一轮都能把当前剩余的最小的放到当前的最前面。（注意，用index指示min，不用每次比较都交换，只需要完成一轮之后交换开头和index指示的对象即可）(区分冒泡排序，冒泡是比较并交换相邻的)
- scanf的返回值：成功读入则返回1；若是输入类型与转换说明的类型不匹配，那返回0；若根本没有输入，返回EOF（一般为-1）。如while(scanf(...) != EOF){……}用来反复读取并处理，直到结束，可以实现边读取边处理。

作业

- 逆序操作
- 交换两个变量：直接换（中间变量temp），函数换（必须借助指针），异或交换（a^=b^=a^=b）
- 异或的一个应用：Bool型变量a，可用a ^= 1 (即a = a ^ 1)来对a 进行取反

第七周

课堂

- clock()：当前时间（<time.h>），如 clock_t time = clock();
- 冒泡排序（bubble-sort）：比较前后两个，如果前面的大于后面的就交换。注意：第一轮是比较第0和第1个、第1和第2个、……第n-2和第n-1个，这样第n-1个为最大元素；第二轮就是从0到n-2，排好的是n-2个……
- 归并（merge）（归并排序的一个步骤），将两个排好序的数组归并成一个排好序的数组。初始l、r、index设为0，a[l]和b[r]比较，小的(假如是b[r])放入c[index]然后index++，r++，继续比较，直到某个数组用完，然后把另一个数组填入c。
- 边界情况（如数组的开头、结尾，二维数组的四周一圈）相邻元素个数和形式和别的不太一样，可以单独讨论，也可以边界扩展。如把n*n的棋盘扩展成(n+2)*(n+2)

第八周

课堂

- 函数的基本知识：返回类型，函数名，形式参数，函数体。函数不能返回数组，所以常常返回一个指针指向一个数组的开头来实现返回数组；可以返回结构体。另外，exit() 和return x的区别是，不论哪个函数调用exit都会使程序终止。（stdlib.h，exit(0)）注意函数是值传递，传进来的参数只是一个原数据的拷贝，所以相当于只能读而不能写，想改变原数据需要传进来原数据的指针。（计基，汇编语言）

课本

- 局部变量：在函数体内声明的变量，有自动存储期和块作用域，从声明到所在函数末尾为作用域，函数返回后该变量即收回。加static后就变为静态存储期，不再收回。
- 全局变量：在函数体外声明，静态存储期、文件作用域（从声明到所在文件末尾）
- 栈结构：push压栈，pop出栈，只能对栈顶数据操作。可以用一个index变量和一个数组、通过改变index来改变栈顶，实现pop、push等功能。

第九周

课堂

- 递归：函数调用它本身。
- 归并排序（mergesort）。借助递归完成，需要Merge和Mergesort两个函数。把原来的数组二分，然后把得到的两个数组继续二分，直到最后递归分成全是一个元素的数组，然后再合并，由于一个元素的数组是已经排好序的，所以直接用Merge函数进行合并即可，然后得到的新数组也是排好序的，继续合并，直到最后合并完成。代码见class文件夹
- main函数的两个参数：int argc, char *argv[]（指向数组的指针，可以用来实现二维数组）。argc用于计数传进来的命令行参数个数，argv储存命令行参数。argv[0],程序的名称；argv[1 ---- n-1]储存命令行参数；argv[n]:空指针NULL。没有其它命令时，argc = 1。
- 找最大公约数：辗转相除法，同样是递归实现，代码略。（gcd(a, b) = gcd(b, a % b)，a>b，最终结束条件是b==0）

作业

- C语言允许连续赋值。如a[x1][y1] = a[x2][y2] = 5。

- 2的多少次方的乘除，尽量用位运算而不是pow函数，省时且简洁。（例： 1 <<(n-1) 表示把二进制下的1左移n-1位， 也就是pow(2,n-1)。）
- 有浮点数一定注意。一是不要不小心把浮点数写成int； 二是浮点数的计算天然有误差，如果需要尽可能精确的话需要注意算法（IEEE 754标准）；三是浮点数的绝对值用fabs()函数（<math.h>）（不能和0比较然后负的话取反，原因为第二条）
- 递归一般比较耗时间，链式的线性递归还好说，如果一个递归函数里面>=2次（比如x次）调用自己来递归，那计算量会指数级暴增(约pow(x,n)级别)。所以递归一定尽可能用比较简单的算法来实现，而且如果需要多次调用这个递归函数一定要开一个变量存储它的结果，这样后续只需要用这个变量的值就可以，一定不要出现多少次调用多少次。（其它也是，比如多次需要pow(x,y), 就令a = pow(x,y), 后续再用就用a。）

课本

快速排序quicksort（重要！），见157页。

第十周

课堂

- 有符号与无符号，整型与浮点型，浮点型的float与double（IEEE 754），浮点数的精度损失。不要轻易混用有符号和无符号类型；混合时，有符号会被转化为无符号再参与运算（-1 --> UINT_MAX）；无符号常用于位运算等底层运算，尽量少用。
- typedef int NNN，现在NNN这种数据类型即为int类型，用来定义新的数据类型。

作业

- &按位与运算，转换为二进制之后再操作。如12 & 5 == 4。（|按位或运算，^按位异或运算，~按位非运算。）。
- 逻辑运算符。&&逻辑与，||逻辑或，！逻辑非。分真和假。0为假，非0为真。（4和1都会被视为1，真的话返回1。）（如：5&6==4，但是5&&6==1。区分）
- n & 1, 判断n的奇偶性，n为奇数则结果为1，偶数则结果为0。例如：if ((i & 1) && !(j & 1)) putchar(' '); 即：如果i为奇数并且j为偶数，则输出' '。

第十一周

课堂

- 变量的三个要素：类型、值和地址。变量在使用时可以用作左值或者右值（作为左值时，指代的是变量的空间/地址；作为右值时，指代的是这个空间里的数值）。指针是一个变量，这个变量的值是另外一个变量的地址。如int *型，表示这个变量是一个指针，它指向一个int类型的变量（注int *p和int *p均可，空格数不要求）
- 用&这个运算符（取地址），表示一个变量的地址。用*这个运算符（间接寻址），用在指针变量p上面（*p），表示p指向的变量
- 在表达式中，数组名实际上就是数组第一个元素的地址。数组作为函数的形参时，写 int *arr 和 int arr[] 和 int arr[10]是完全等价的, 均是数组第一个元素的指针
- malloc函数（<stdlib.h>），申请一块空间，用于开可变长数组。如 int *numbers = (int *)malloc(len * sizeof(*numbers)) （相当于开了一个numbers[len]数组）。但是，malloc开的空间是在堆空间上的，而main函数里开的数组、变量都是在栈空间上的。main函数执行完后，数据会释放掉；但是堆空间上的不能释放。所以结束之后堆空间这个数组数据还在，有内存泄漏的风险。所以经常需要手动释放：free(numbers); 表示把它释放掉。释放后相当于这个数组不存在了，这之后不能再读写它。free释放的只能是malloc开的空间，不能释放栈空间！（未定义行为）。不允许多次释放！（未定义行为）（<stdlib.h>）
- NULL 表示空指针

作业

- 指针类型的强制类型转换：对于任意类型的指针，由于它们的字长是相同的，因此它们之间可以无阻碍地进行强制转换。如对于某个 int *a， 可以通过 (double *)a 来获取一个 double * 类型的指针。那么毫无疑问，如果你对这个指针进行解引用，即 *(double *)a ，则机器会按照 double 的类型约定，来解析从指针位置开始的8 字节作为解引用的值。(在 C 里，所有的指针类型都是机器相关的。指针长度与机器位宽相等，用来指示内存当中的某个位置。)
- 几个库函数：int toupper(int c)如果(char)c是小写字符，则返回对应的大写字符，如果不是小写字符，则返回其本身(<ctype.h>)。int tolower(int c)与刚才的相反（to upper/lower）。char *strstr(const char *s1，const char *s2); 在字符串s1中寻找s2，找到则返回s2第一次在s1中出现时的位置（指针），找不到则返回NULL（<string.h>）

第十二周

课堂

- 几个库函数（<string.h>）。int strcmp(const char *s1, const char *s2);比较两个字符串，如果一样则返回0，如果s1大则返回1，如果s2大则返回-1（大小是根据遇到的第一个不同字符的ASCII码比较，有点像英文词典的单词排序）。int strlen(const char *s);返回字符串s的长度（遇到\0表示字符串结束），注意，在循环里需要strlen的话一定先把它算出来并存到某个变量里之后再用这个变量开始循环，不要反复调用它！void strcpy(char *s1, const char *s2);把s2指向的字符串复制并存到s1指向的字符数组中。

作业

- 库函数：（<string.h>）char *strstr(const char *s, const char *c);在字符串s中找字符c，返回第一次出现的位置（指针），没有则返回NULL。如strstr(s,';')。char *strcat(char *s1, const char *s2)，把字符串s2拼接到s1后面（注意，s1必须足够大）。char *strncat(char *s1, const char *s2, size_t n)：将s2的前n个字符拼接到s1后。
- int sprintf(char *s1, char *s2, arg_list);作用：把后面s2这个字符串输出到s1指向的字符数组的位置，arg_list为s2的参数表，相当于printf到s1中而不是屏幕上。例如以下代码，输出结果为HEWORLD！114514

```
#include <stdio.h>

int main(){
    char s[100] = "HELLO ";
    sprintf(s+2,"WORLD!%d",114514);
    printf("%s",s);
    return 0;
}
```

第十三周

课堂

- 对于各种指针类型的理解！！区分函数指针与指针函数：int *a(……)是指针函数，本质上是个函数，其返回类型是指向int的指针；int (*a)(……)是函数指针，指向一个函数的首地址。函数在使用时会被转换为函数指针。因此如果某个函数的参数是一个函数的话，那么这个参数在声明的时候应该声明成函数指针，调用的时候实际参数直接用某个函数就行。

- double pointers(双重指针)，即指针指向的对象是一个指针。char *str[]:str数组，数组里的元素是指向char的指针

第十四周

课堂

- 结构：结构是一种数据类型，像数组一样可以包括多个元素，但是元素的类型可以不同。 每个元素有特定的名字来区分彼此。结构是程序员自定义的，为了区分不同的结构，需要给定义的某种特定结构命名。可以用结构标记来命名，也可用typedef来定义类型名。 例：part为结构标记，以后struct part就是像int、float这样的类型名，指这种类型。（注意分号）(不能单独用part！)

```
struct part{
    int number;
    char name[10];
    float score;
};

struct part part1 = {114,"syd",95.5};
```

- 还可以用typedef命名。例：此时是单独用part而不是struct part。

```
typedef struct{
    int number;
    char name[10];
    float score;
}part;

part part1 = {114,"syd",95.5};
```

- 注：用于链表等情况时，只能用第一种（因为这些情况出现了结构里套着这种类型的结构，typedef中part出现在后面，编译器还不知道part是什么就遇到了part结构体，从而失败）

- 结构成员的访问通过'.'运算符来实现，如：part1.number = 114;

- 结构的成员是左值；同类型的结构可以用于赋值操作，即part2 = part1; 甚至可以复制含数组成员的结构，由此可以实现数组复制。但不支持别的用于整个结构的操作了，比如判等（==，!=）

课本

- 联合：与结构在形式上类似，struct换成union，但是结构的成员储存在不同位置，占用空间至少为各部分占用之和（实际上总空间为占用空间最大的元素的整数倍，且每一成员起始地址均需要根据自己所占空间来对齐，如int起始地址为4的倍数）；而编译器只为联合中最大的成员分配足够内存空间，然后各成员在这块空间彼此覆盖。存储多选一的数据时可以用来节省空间。但实际上一般而言联合能做的事结构都能做……
- 枚举(enum)：类似将宏定义的数据集合到一起，每一个枚举常量有一个特定的值，默认为0、1、2、3……，也可自己指定。且同样可以用枚举标记或typedef来命名。枚举和宏其实非常类似：宏在预处理阶段将名字替换成对应的值，枚举在编译阶段将名字替换成对应的值。enum中的量都是常量，不能对它们赋值，只能将它们的值赋给其他的变量。我们可以将枚举理解为编译阶段的宏。如：`enum size{SMALL = 3,MIDDLE = 2,LARGE = 1};`注意枚举中的常量都是在作用域范围中可见的（如定义在main中则在main中可见，在main外则全文件可见），因此不能在定义重名的变量/常量了。

第十五周

课堂

- 大型程序：源文件、头文件。包含别的头文件（非c库）用""而不是<>。具体细节略，见课堂代码；Makefile略
- 头文件中一定要用#ifndef来保护！防止因多次包含头文件而出现错误。
- malloc/calloc/realloc：动态存储分配。malloc不初始化，calloc初始化，realloc调整大小（`<stdlib.h>`），得到的空间都是在堆空间上。用完后要及时free释放掉空间。注意free容易产生悬空指针！
- 动态存储分配对建立表、树、图都很有用。
- 链表：由一连串节点组成，每个节点中都含有指向下一节点的指针。常用结构体来作为链表的节点。
- ->运算符的使用，链表的插入、删除和搜索略，自己看课本。
- qsort函数！！！

作业

- 哈希表的概念与实现。

课本

- 预处理器：在编译前执行预处理指令，进行预处理；把所有注释替换为一个空格字符（个别情况下有可能因为这个原因导致有注释和没注释的结果不一样）。预处理指令：宏定义、文件包含、条件编译等。宏定义可以是简单的宏，也可以是带参数的宏；其它具体介绍略。