习题解答

1、for循环

输入10个数字，输出其中最大的一个数字

答案：

1. 定义一个整型数组 `numbers` 来存储用户输入的10个数字。
2. 定义一个整型变量 `max` 来存储找到的最大数字。
3. 使用 `printf` 函数提示用户输入10个数字。
4. 使用 `for` 循环和 `scanf` 函数读取用户输入的10个数字，并将它们存储在 `numbers` 数组中。
5. 将数组中的第一个数字赋值给 `max` 变量，作为初始的最大值。
6. 再次使用 `for` 循环遍历数组中的其余数字，如果发现比当前 `max` 更大的数字，则更新 `max` 的值。
7. 循环结束后，使用 `printf` 函数输出找到的最大数字。

```c
#include <stdio.h>

int main() {
    int numbers[10]; // 创建一个数组来存储10个数字
    int max; // 用于存储最大数字的变量
    int i; // 循环计数器

    // 读取用户输入的10个数字
    printf("请输入10个数字：\n");
    for (i = 0; i < 10; i++) {
        scanf("%d", &numbers[i]);
    }

    // 初始化最大值为第一个数字
    max = numbers[0];

    // 遍历数组，找到最大的数字
    for (i = 1; i < 10; i++) {
        if (numbers[i] > max) {
            max = numbers[i]; // 更新最大值
        }
    }

    // 输出最大的数字
    printf("最大的数字是：%d\n", max);

    return 0;
}
```

2、字符串

请编程序将“China"译成密码，密码规律是:用原来的字母后面第4个字母代替原来的字母。例如,字母“A”后面第4个字母是“E”,用“E”代替“A”。因此,“China"应译为“Glmre”。请编一程序,用赋初值的方法使cl,c2,c3,c4,c5这5个变量的值分别为'C','h','i','n','a' ,经过运算,使c1,c2,c3,c4,c5 分别变为'G','l','m','r','e'。分别用putchar函数和printf函数输出这5个字符。

答案:

1. **理解问题**：我们需要将给定的单词"China"中的每个字母转换为其在字母表中后面第4个字母。这意味着我们需要对字母进行偏移操作。
2. **字母偏移**：英文字母表中有26个字母。我们需要对每个字母进行偏移，即向后移动4位。例如，‘A’（第一个字母）偏移4位后变成‘E’（第五个字母）。
3. **区分大小写**：英文字母有大小写之分，所以我们需要分别处理大写字母和小写字母。大写字母从‘A’到‘Z’，小写字母从‘a’到‘z’。
4. **循环偏移**：当偏移后的字母超过‘Z’或‘z’时，需要循环回到字母表的开始。例如，‘X’偏移4位后应该是‘D’，而不是超过‘Z’的字母。
5. **数学计算**：为了实现循环偏移，我们可以使用模运算（%）来处理。具体来说，对于大写字母，我们可以用`(original_char - 'A' + 4) % 26 + 'A'`来计算新字符；对于小写字母，我们可以用`(original_char - 'a' + 4) % 26 + 'a'`来计算。

```c
#include <stdio.h>

int main() {
    // 初始化字符变量
    char c1 = 'C', c2 = 'h', c3 = 'i', c4 = 'n', c5 = 'a';

    // 对每个字符进行转换，字母后面第4个字母
    c1 = (c1 - 'A' + 4) % 26 + 'A';
    c2 = (c2 - 'a' + 4) % 26 + 'a';
    c3 = (c3 - 'a' + 4) % 26 + 'a';
    c4 = (c4 - 'a' + 4) % 26 + 'a';
    c5 = (c5 - 'a' + 4) % 26 + 'a';

    // 使用putchar函数输出字符
    putchar(c1);
    putchar(c2);
    putchar(c3);
    putchar(c4);
    putchar(c5);
    putchar('\n'); // 输出换行符

    // 使用printf函数输出字符
    printf("%c%c%c%c%c\n", c1, c2, c3, c4, c5);

    return 0;
}
```

3、字符串

输入一行字符，分别统计其中英文字母、空格、数字和其他字符的个数。

答案：

- 通过比较字符的ASCII值来确定它们属于哪个类别。
- `c >= 'A' && c <= 'Z'`：检查字符`c`是否为大写英文字母。
- `c >= 'a' && c <= 'z'`：检查字符`c`是否为小写英文字母。
- `c == ' '`：检查字符`c`是否为空格。
- `c >= '0' && c <= '9'`：检查字符`c`是否为数字字符。

用户输入一行文本并按下回车键后，程序将统计并输出输入文本中的英文字母、空格、数字和其他字符的个数。

```c
#include <stdio.h>

int main() {
    char c; // 用于存储输入的字符
    int letters = 0; // 英文字母的个数
    int spaces = 0; // 空格的个数
    int digits = 0; // 数字字符的个数
    int others = 0; // 其他字符的个数

    printf("请输入一行字符：\n");

    // 读取字符直到遇到换行符
    while ((c = getchar()) != '\n') {
        // 判断英文字母（不区分大小写）
        if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z')) {
            letters++;
        }
        // 判断空格
        else if (c == ' ') {
            spaces++;
        }
        // 判断数字
        else if (c >= '0' && c <= '9') {
            digits++;
        }
        // 其他字符
        else {
            others++;
        }
    }

    // 输出结果
    printf("英文字母的个数：%d\n", letters);
    printf("空格的个数：%d\n", spaces);
    printf("数字字符的个数：%d\n", digits);
    printf("其他字符的个数：%d\n", others);

    return 0;
}
```

4、数组

将一个数组中的值按逆序重新存放，例如，原来顺序为8,6,5,4,1。要求改为1,4,5,6,8

答案:

```c
#include <stdio.h>

void reverseArray(int arr[], int size) {
    int temp;
    for (int i = 0; i < size / 2; i++) {
        // 交换元素
        temp = arr[i];
        arr[i] = arr[size - 1 - i];
        arr[size - 1 - i] = temp;
    }
}

int main() {
    int array[] = {8, 6, 5, 4, 1};
    int size = sizeof(array) / sizeof(array[0]);

    // 输出原始数组
    printf("原始数组顺序为：");
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");

    // 逆序数组
    reverseArray(array, size);

    // 输出逆序后的数组
    printf("逆序后的数组顺序为：");
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");

    return 0;
}
```

5、魔法阵

```c
//输出魔方阵
/*
1、第一行的中间一列为1，用j = n/2+1 确定a[1][j] = 1
2、每一个数存放的行比前一个数的行数减1， 列数加1
3、一个数行数为第一行，下一个数行数为最后一行
4、一个列数为最后一列，下一个数列数为第一列
5、如果按上面的规则所确定的位置上已有数，或上一个数是第一行最后一列，则把下一个数放在上一个数的下面
*/
```

```
void OutMagicCube(){
    int a[N][N]={0},i,j,k,n;
    n = 4;

    i = n+1;             //从最后一行开始存储
    j = n/2+1;
    a[1][j] = 1; //第一行中间一列的值为1

    for(k = 2; k <= n*n; k++){
        i = i-1;
        j = j+1;                 //每一个数为 行数减1，列数加1
        if((i<1) && (j>n)){
            i = i+2;             //如果为最后一列并且为第一行
            j = j-1;
        }
        else{
            if(i<1) i = n;       //上个数第一行，下个数第n行
            if(j>n) j = 1;       //上个数第n列，下个数第一列
        }
        if(a[i][j] == 0) a[i][j] = k;
        else{                    //有数，放在下面 i+2本来是-1，然后行数要+2才是下一行，列
不变，之前+1，现在-1
            i = i+2;
            j = j-1;
            a[i][j] = k;
        }
    }

    for(i = 1; i <= n; i++){
        for(j = 1; j<=n; j++){
            printf("%-4d", a[i][j]);
        }
        printf("\n");
    }
}
```

6、函数

写一个函数，将两个字符串连接

使用库函数：

```
#include <stdio.h>
#include <string.h>

void concatenateStrings(char *destination, const char *source) {
    // 复制destination到临时字符串，以便不覆盖原始destination
    char temp[strlen(destination) + 1]; // +1 为 '\0' 终止符
    strcpy(temp, destination);

    // 将source连接到destination
    strcat(destination, source);
}

int main() {
```

```
    char str1[100] = "Hello, ";
    char str2[] = "World!";

    // 连接字符串
    concatenateStrings(str1, str2);

    // 输出结果
    printf("连接后的字符串为: %s\n", str1);

    return 0;
}
```

不使用库函数:

```
#include <stdio.h>

void myStrCat(char *dest, const char *src) {
    // 找到dest的末尾
    while (*dest) {
        dest++;
    }

    // 复制src到dest的末尾
    while (*src) {
        *dest = *src;
        dest++;
        src++;
    }

    // 添加字符串终止符
    *dest = '\0';
}

int main() {
    char str1[100] = "Hello, ";
    char str2[] = "World!";

    // 连接字符串
    myStrCat(str1, str2);

    // 输出结果
    printf("连接后的字符串为: %s\n", str1);

    return 0;
}
```

7、学生与课程

输人10个学生5门课的成绩,分别用函数实现下列功能:

①计算每个学生的平均分;
②计算每门课的平均分;
③找出所有50个分数中最高的分数所对应的学生和课程;
④计算平均分方差:

```c
#include <stdio.h>

#define STUDENTS 10  // 定义学生数量常量
#define COURSES 5    // 定义课程数量常量

// 函数声明
void calculateStudentAverage(float grades[STUDENTS][COURSES]);
void calculateCourseAverage(float grades[STUDENTS][COURSES]);
void findHighestScore(float grades[STUDENTS][COURSES]);
float calculateVariance(float averages[], int size);

int main() {
    float grades[STUDENTS][COURSES];  // 存储所有学生各科成绩的二维数组
    float studentAverages[STUDENTS];   // 存储每个学生的平均分
    float courseAverages[COURSES];     // 存储每门课的平均分

    // 输入学生成绩
    printf("请输入10个学生5门课的成绩：\n");
    for (int i = 0; i < STUDENTS; i++) {  // 遍历每个学生
        printf("学生 %d:\n", i + 1);
        for (int j = 0; j < COURSES; j++) {  // 遍历每门课程
            scanf("%f", &grades[i][j]);  // 读取成绩
        }
    }

    // 计算每个学生的平均分
    calculateStudentAverage(grades);

    // 计算每门课的平均分
    calculateCourseAverage(grades);

    // 找出最高分数的学生和课程
    findHighestScore(grades);

    // 计算平均分方差
    // 首先计算所有学生的平均分
    for (int i = 0; i < STUDENTS; i++) {
        float sum = 0;  // 存储单个学生的总分
        for (int j = 0; j < COURSES; j++) {
            sum += grades[i][j];  // 累加每门课的成绩
        }
        studentAverages[i] = sum / COURSES;  // 计算平均分
    }
    // 然后计算方差
    float variance = calculateVariance(studentAverages, STUDENTS);
    printf("平均分方差为: %.2f\n", variance);  // 输出方差

    return 0;
}

// 计算每个学生的平均分
void calculateStudentAverage(float grades[STUDENTS][COURSES]) {
    printf("每个学生的平均分：\n");
    for (int i = 0; i < STUDENTS; i++) {  // 遍历每个学生
        float sum = 0;  // 存储单个学生的总分
```

```
        for (int j = 0; j < COURSES; j++) {
            sum += grades[i][j];  // 累加每门课的成绩
        }
        printf("学生 %d: 平均分 = %.2f\n", i + 1, sum / COURSES);  // 输出平均分
    }
}

// 计算每门课的平均分
void calculateCourseAverage(float grades[STUDENTS][COURSES]) {
    printf("每门课的平均分：\n");
    for (int j = 0; j < COURSES; j++) {  // 遍历每门课程
        float sum = 0;  // 存储单门课程的总分
        for (int i = 0; i < STUDENTS; i++) {
            sum += grades[i][j];  // 累加每个学生的成绩
        }
        printf("课程 %d: 平均分 = %.2f\n", j + 1, sum / STUDENTS);  // 输出平均分
    }
}

// 找出最高分数的学生和课程
void findHighestScore(float grades[STUDENTS][COURSES]) {
    float highest = 0;  // 初始化最高分数为0
    int studentIndex = 0;  // 初始化最高分数对应的学生索引
    int courseIndex = 0;  // 初始化最高分数对应的课程索引

    // 遍历所有学生的所有成绩
    for (int i = 0; i < STUDENTS; i++) {
        for (int j = 0; j < COURSES; j++) {
            // 如果当前成绩大于已记录的最高分数，则更新最高分数和对应的索引
            if (grades[i][j] > highest) {
                highest = grades[i][j];
                studentIndex = i;
                courseIndex = j;
            }
        }
    }
    // 输出最高分数及其对应的学生和课程
    printf("最高分数为 %.2f，学生 %d 的课程 %d\n", highest, studentIndex + 1,
courseIndex + 1);
}

// 计算方差
float calculateVariance(float averages[], int size) {
    float mean = 0;  // 存储平均值
    float variance = 0;  // 存储方差

    // 计算平均值
    for (int i = 0; i < size; i++) {
        mean += averages[i];
    }
    mean /= size;

    // 计算方差
    for (int i = 0; i < size; i++) {
        variance += (averages[i] - mean) * (averages[i] - mean);
    }
```

```
    variance /= size;  // 计算样本方差

    return variance;  // 返回方差
}
```

8、函数

写几个函数:

①输人10个职工的姓名和职工号;

②按职工号由小到大顺序排序,姓名顺序也随之调整;

③要求输人一个职工号,用折半查找法找出该职工的姓名,从主函数输人要查找的职工号,输出该职工姓名

答案:

```
#include <stdio.h>
#include <string.h>

#define EMPLOYEES 10

// 职工结构体
typedef struct {
    char name[50];  // 职工姓名
    int id;         // 职工号
} Employee;

// 函数声明
void inputEmployees(Employee employees[], int size);
void sortEmployeesById(Employee employees[], int size);
void halfSearchEmployee(Employee employees[], int size, int searchId);
void swapEmployees(Employee *a, Employee *b);

int main() {
    Employee employees[EMPLOYEES];
    int searchId;

    // 输入职工姓名和职工号
    inputEmployees(employees, EMPLOYEES);

    // 按职工号排序
    sortEmployeesById(employees, EMPLOYEES);

    // 输入要查找的职工号
    printf("请输入要查找的职工号: ");
    scanf("%d", &searchId);

    // 折半查找职工姓名
    halfSearchEmployee(employees, EMPLOYEES, searchId);

    return 0;
}

// 输入职工姓名和职工号
void inputEmployees(Employee employees[], int size) {
```

```
    for (int i = 0; i < size; i++) {
        printf("输入职工 #%d 的姓名: ", i + 1);
        scanf("%s", employees[i].name);
        printf("输入职工 #%d 的职工号: ", i + 1);
        scanf("%d", &employees[i].id);
    }
}

// 冒泡排序职工按职工号
void sortEmployeesById(Employee employees[], int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (employees[j].id > employees[j + 1].id) {
                swapEmployees(&employees[j], &employees[j + 1]);
            }
        }
    }
}

// 交换两个职工的位置
void swapEmployees(Employee *a, Employee *b) {
    Employee temp = *a;
    *a = *b;
    *b = temp;
}

// 折半查找职工姓名
void halfSearchEmployee(Employee employees[], int size, int searchId) {
    int low = 0, high = size - 1, mid;
    while (low <= high) {
        mid = low + (high - low) / 2;
        if (employees[mid].id == searchId) {
            printf("找到职工姓名: %s\n", employees[mid].name);
            return;
        } else if (employees[mid].id < searchId) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    printf("没有找到职工号 %d 的职工。\n", searchId);
}
```

9、进制转换

写一个函数，输出一个16进制，输出相应的10进制

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// 函数声明
long hexToDecimal(const char *hex);

int main() {
```

```
    char hexNumber[100]; // 16个字符加上一个字符串结束符'\0'

    // 用户输入16进制数
    printf("请输入一个16进制数（如 1A3F）: ");
    scanf("%s", hexNumber);

    // 输出对应的10进制数
    printf("对应的10进制数是: %ld\n", hexToDecimal(hexNumber));

    return 0;
}

// 将16进制字符串转换为10进制数
long hexToDecimal(const char *hex) {
    long decimalValue = 0;
    int length = strlen(hex);

    for (int i = 0; i < length; i++) {
        char currentChar = hex[i];

        // 将当前字符转换为对应的数值
        int digitValue;
        if (currentChar >= '0' && currentChar <= '9') {
            digitValue = currentChar - '0';
        } else if (currentChar >= 'a' && currentChar <= 'f') {
            digitValue = currentChar - 'a' + 10;
        } else if (currentChar >= 'A' && currentChar <= 'F') {
            digitValue = currentChar - 'A' + 10;
        } else {
            printf("非法的16进制数。\n");
            exit(EXIT_FAILURE);
        }

        // 累加到结果中
        decimalValue = decimalValue * 16 + digitValue;
    }

    return decimalValue;
}
```