

03班第一次机考复习文档

TA: 孟昭明

03班第一次机考复习文档

考试相关

复习内容

易错点

常见算法

常见库函数

祝大家考试顺利!

1. 考试相关

- 共4道题，每题25分，满分100分
- 考试范围：指针之前的所有内容
- 考试时间：60分钟
- 在正式机考之前会进行一次模拟考试，如发现问题请及时反馈给助教或老师
- 2次机考一共占20分（最后总评）
- 可以多次提交，只能看到得分，不能看到测试数据
- 必须使用机房电脑进行考试
- 建议先在DevC++或VsCode等IDE中完成代码编写和调试，再复制到机考系统中提交
- 目前，课程组要求：考多少分就是多少分，机考批改不给同情分，后续是否会根据答题情况进行调整未知
- 模拟考试及正式考试结束后，仍有上机实验，请大家到时候继续完成上机实验

2. 复习内容

2.1 易错点

- scanf函数何时使用取地址符&，何时不使用取地址符&

```
1  int a;  
2  scanf("%d", &a); // 变量a需要取地址符&  
3  char str[100];  
4  scanf("%s", str); // 数组名str本身就是地址，不需要取地址符&
```

- printf函数中格式控制符与变量类型的对应关系

变量类型	格式控制符
int	%d

变量类型	格式控制符
float	%f
double	%lf
char	%c
字符数组/字符串	%s

printf函数不需要加取地址符&

- = 赋值运算符与 == 关系运算符的区别

```
1  int a = 5; // 赋值运算符，将5赋值给变量a
2  if (a == 5) { // 关系运算符，判断变量a是否等于5
3      // 条件成立
4  }
```

- switch语句中case后面的值只能是常量，不能是变量或表达式，每个case后面要加break语句，否则会发生“贯穿”现象

```
1  switch (num) {
2      case 1:
3          // 处理num等于1的情况
4          break;
5      case 2:
6          // 处理num等于2的情况
7          break;
8      default:
9          // 处理其他情况
10         break;
11 }
```

- 注意循环语句中for语句，while语句，do...while语句的区别
 - 一般情况下，我们会用for语句来进行已知次数的循环，用while语句来进行未知次数的循环
 - do...while语句的特点是至少会执行一次循环体
- 建议结束循环后将循环变量重置为初始值，避免影响后续代码，在此给出一个没有重置循环变量的错误示例：

```
1  #include <stdio.h>
2
3  int main() {
4      int i = 0;
5      int array[5] = {10, 20, 30, 40, 50};
6      while (i < 5) { // 打印数组元素
7          printf("%d ", array[i]);
8          i++;
9      }
```

```

10 // 循环结束后，i 的值为 5
11 int sum = 0;
12 // 因为 i 没有重置为 0，这里就无法进入while循环正确计算前三个元素的和
13 while (i < 3) {
14     sum += array[i];
15     i++;
16 }
17 printf("前三个元素的和为: %d\n", sum);
18 return 0;
19 }

```

正确的处理方式要么是重置循环变量*i*，即在第一次循环结束后添加 `i = 0;`，要么使用for循环：

```

1 for (int i = 0; i < 3; i++) {
2     sum += array[i];
3 }

```

当然也可以直接换一个新的变量名，不过变量名过多会导致代码可读性变差

- 不要使用三层以上的嵌套循环或者嵌套条件语句
 - 多重嵌套循环——算法时间复杂度高
 - 多重嵌套条件语句——代码可读性差（屎山代码）
- 对数组或变量定义后，不要忘记赋初值

```

1 int arr[10]; // 定义数组arr
2 // 忘记给arr赋初值，可能会导致C语言给arr分配的内存中有垃圾值
3 for (int i = 0; i < 10; i++) {
4     arr[i] = 0; // 给数组arr赋初值
5 }

```

- 注意防止数组下标越界
- **尽量不要使用全局变量!!!**
- 如果涉及到math.h中的数学函数，最好选择double类型进行计算，避免精度问题
- C语言中没有指数运算符[^]，[^]是按位异或运算符，如需计算指数，需使用math.h中的pow函数（当然也可以自定义指数函数）

```

1 // 计算2的3次方
2 #include <stdio.h>
3 #include <math.h>
4
5 int main() {
6     double result = pow(2, 3); // 使用math.h中的pow函数计算2的3次方
7     printf("2的3次方是: %lf\n", result);
8     return 0;
9 }

```

- 函数中形参为数组的情况，下面提供一个示例，供参考（实现了打印一维数组和二维数组的功能）

```

1  #include <stdio.h>
2  #define COLS 4 // 定义二维数组的列数
3  // 一维数组
4  void func1(int arr[], int size) {
5      // 一维数组的[]中啥也不写，数组大小通过形参size传入
6      for (int i = 0; i < size; i++) {
7          printf("%d ", arr[i]);
8      }
9  }
10
11 // 二维数组
12 void func2(int arr[][COLS], int rows) {
13     // 二维数组的第一个[]中啥也不写，第二个[]中必须写列数COLS，行数通过形参rows传入
14     for (int i = 0; i < rows; i++) {
15         for (int j = 0; j < COLS; j++) {
16             printf("%d ", arr[i][j]);
17         }
18         printf("\n");
19     }
20 }
21
22 int main() {
23     int arr1[5] = {1, 2, 3, 4, 5};
24     func1(arr1, 5); // 传入一维数组的数组名和大小
25
26     int arr2[3][4] = {
27         {1, 2, 3, 4},
28         {5, 6, 7, 8},
29         {9, 10, 11, 12}
30     };
31     func2(arr2, 3); // 传入二维数组的数组名和行数
32     return 0;
33 }

```

- 注意区分字符型变量和字符串
 - 字符型变量：用单引号括起来的单个字符，如 `char ch = 'A';`
 - 字符串：用双引号括起来的一串字符，如 `char str[] = "Hello";`
- C语言会对于字符串自动添加结束符 `\0`，所以定义字符串数组时要考虑结束符所占的空间

```

1  char str[6] = "Hello"; // 正确，数组大小为6，包含5个字符和1个结束符
2  char str[5] = "Hello"; // 错误，数组大小为5，无法容纳结束符

```

如果需要我们自定义字符串操作函数，如读取，复制，连接等操作时，我们可以**手动为字符数组结尾添加结束符 `\0` 以简化操作**（这样能够方便我们知道字符串的结束位置，也可以直接使用printf函数打印字符串）

```

1  printf("%s", str); // 直接打印字符串，直到遇到结束符\0

```

- 比较两个字符串是否相等，不能使用 `==` 运算符，而应该使用 `strcmp` 函数

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main() {
5      char str1[] = "Hello";
6      char str2[] = "Hello";
7      if (strcmp(str1, str2) == 0) {
8          printf("字符串相等\n");
9      } else {
10         printf("字符串不相等\n");
11     }
12     return 0;
13 }
```

2.2 常见算法

- 交换两个变量的值

```
1  int a, b, temp;
2  temp = a;
3  a = b;
4  b = temp;
```

思考：以下代码能否实现交换两个变量的值？为什么？

```
1  void swap(int x, int y) {
2      int temp = x;
3      x = y;
4      y = temp;
5  }
6
7  int main() {
8      int a = 5, b = 10;
9      swap(a, b);
10     printf("a = %d, b = %d\n", a, b);
11     return 0;
12 }
```

答案：**不能实现交换两个变量的值**，因为C语言中函数参数是按值传递的，`swap`函数中的`x`和`y`只是`a`和`b`的副本，对`x`和`y`的修改不会影响到`main`函数中的`a`和`b`。应该使用**指针**来实现交换：

```

1 void swap(int *x, int *y) {
2     int temp = *x;
3     *x = *y;
4     *y = temp;
5 }
6
7 int main() {
8     int a = 5, b = 10;
9     swap(&a, &b);
10    printf("a = %d, b = %d\n", a, b);
11    return 0;
12 }

```

- 最大公约数与最小公倍数——辗转相除法

```

1 // 计算最大公约数
2 int gcd(int a, int b) {
3     while (b != 0) {
4         int temp = b;
5         b = a % b;
6         a = temp;
7     }
8     return a;
9 }
10
11 // 计算最小公倍数
12 int lcm(int a, int b) {
13     return (a * b) / gcd(a, b);
14 }

```

- 素数判断

```

1 int isPrime(int n) {
2     if (n <= 1) return 0; // 1及以下的数不是素数
3     if (n == 2) return 1; // 2是素数
4     for (int i = 3; i * i <= n; i++) { // 只需检查到sqrt(n)
5         if (n % i == 0) return 0; // 能被整除，非素数
6     }
7     return 1; // 是素数
8 }

```

- 贪心算法——找零问题

```

1 void makeChange(int amount) {
2     int coins[] = {25, 10, 5, 1}; // 美分硬币面值
3     int count[4] = {0}; // 各种硬币的数量
4     for (int i = 0; i < 4; i++) {
5         while (amount >= coins[i]) {
6             amount -= coins[i];
7             count[i]++;

```

```

8         }
9     }
10    printf("25分硬币: %d\n", count[0]);
11    printf("10分硬币: %d\n", count[1]);
12    printf("5分硬币: %d\n", count[2]);
13    printf("1分硬币: %d\n", count[3]);
14 }

```

- 递归函数——Hanoi塔问题

```

1 void hanoi(int n, char src, char aux, char dest) {
2     if (n == 1) {
3         printf("将圆盘 1 从 %c 移动到 %c\n", src, dest);
4         return;
5     }
6     // 1. 将 n-1 个圆盘从 src 移动到 aux (借助 dest)
7     hanoi(n - 1, src, dest, aux);
8     // 2. 将最大的第 n 个圆盘从 src 移动到 dest
9     printf("将圆盘 %d 从 %c 移动到 %c\n", n, src, dest);
10    // 3. 将 n-1 个圆盘从 aux 移动到 dest (借助 src)
11    hanoi(n - 1, aux, src, dest);
12 }

```

递归函数设计关键在于找到递归关系和递归终止条件

- 数组元素的插入与删除

```

1 // 在数组arr的index位置插入value
2 void insert(int arr[], int *size, int index, int value) {
3     for (int i = *size; i > index; i--) {
4         arr[i] = arr[i - 1];
5     }
6     arr[index] = value;
7     (*size)++;
8 }
9
10 // 删除数组arr中index位置的元素
11 void delete(int arr[], int *size, int index) {
12     for (int i = index; i < *size - 1; i++) {
13         arr[i] = arr[i + 1];
14     }
15     (*size)--;
16 }

```

- 冒泡排序

```

1 void bubbleSort(int arr[], int size) {
2     for (int i = 0; i < size - 1; i++) {
3         for (int j = 0; j < size - 1 - i; j++) {
4             if (arr[j] > arr[j + 1]) {
5                 // 交换arr[j]和arr[j + 1]
6                 int temp = arr[j];
7                 arr[j] = arr[j + 1];
8                 arr[j + 1] = temp;
9             }
10        }
11    }
12 }

```

- 插入排序

```

1 void insertionSort(int arr[], int size) {
2     for (int i = 1; i < size; i++) {
3         int key = arr[i];
4         int j = i - 1;
5         // 将arr[i]插入到已排序的子数组arr[0..i-1]中
6         while (j >= 0 && arr[j] > key) {
7             arr[j + 1] = arr[j];
8             j--;
9         }
10        arr[j + 1] = key;
11    }
12 }

```

- 选择排序

```

1 void selectionSort(int arr[], int size) {
2     for (int i = 0; i < size - 1; i++) {
3         int minIndex = i;
4         for (int j = i + 1; j < size; j++) {
5             if (arr[j] < arr[minIndex]) {
6                 minIndex = j;
7             }
8         }
9         // 交换arr[i]和arr[minIndex]
10        int temp = arr[i];
11        arr[i] = arr[minIndex];
12        arr[minIndex] = temp;
13    }
14 }

```

- 二分查找

```

1 int binarySearch(int arr[], int size, int target) {
2     int left = 0;
3     int right = size - 1;

```

```

4     while (left <= right) {
5         int mid = left + (right - left) / 2; // 防止溢出
6         if (arr[mid] == target) {
7             return mid; // 找到目标元素，返回索引
8         } else if (arr[mid] < target) {
9             left = mid + 1; // 在右半部分继续查找
10        } else {
11            right = mid - 1; // 在左半部分继续查找
12        }
13    }
14    return -1; // 未找到目标元素
15 }

```

二分查找要求数组必须是有序的

- 字符串反转

```

1     void reverseString(char str[]) {
2         int left = 0;
3         int right = strlen(str) - 1; // strlen函数需要#include <string.h>
4         while (left < right) {
5             char temp = str[left];
6             str[left] = str[right];
7             str[right] = temp;
8             left++;
9             right--;
10        }
11    }

```

- 整数反转输出

```

1     void reversePrint(int num) {
2         if (num == 0) {
3             printf("0");
4             return;
5         }
6         if (num < 0) {
7             printf("-");
8             num = -num; // 处理负数
9         }
10        int reversed = 0;
11        while (num > 0) {
12            reversed = reversed * 10 + (num % 10);
13            num /= 10;
14        }
15        printf("%d", reversed);
16    }

```

2.3 常见库函数

- math库函数

```
1  #include <math.h>
2  double sqrt(double x);           // 计算平方根
3  double pow(double base, double exp); // 计算base的exp次方
4  double fabs(double x);           // 计算绝对值
```

- string库函数

```
1  #include <string.h>
2  int strlen(const char *str);      // 计算字符串长度
3  char *strcpy(char *dest, const char *src); // 复制字符串
4  char *strcat(char *dest, const char *src); // 连接字符串
5  int strcmp(const char *str1, const char *str2); // 比较两个字符串
```

3. 祝大家考试顺利!
