《程序设计基础》考前串讲

林嘉宁
971692392@qq.com

王钧石
bjwangjunshi@sina.com

学业与发展支持中心

2021 年 11 月 20 日

目录

1. 输入输出
   - 组数不定的数据输入
   - 格式化输入
2. 逻辑判断
3. 位运算
4. 函数
   - 重复计算
   - 递归
5. 二分查找
6. 排序
   - 多关键字排序
   - qsort函数
7. 数组
   - 二维数组的遍历
   - 使用数组简化遍历
8. 字符串
9. 高精度

1 输入输出
- 组数不定的数据输入
- 格式化输入

2 逻辑判断

3 位运算

4 函数
- 重复计算
- 递归

5 二分查找

6 排序
- 多关键字排序
- qsort函数

7 数组
- 二维数组的遍历
- 使用数组简化遍历

8 字符串

9 高精度

scanf使用EOF

```
1  while (scanf("%d%d", &n, &m) != EOF)
2  {
3      if (n == m)
4      {
5          printf("Equal!");
6      }
7      else
8      {
9          printf("Not equal!");
10     }
11 }
```

EOF是宏定义的int型整数-1。

scanf函数返回成功读入的变量数量。

gets使用NULL

```
1  while (gets(str) != NULL)
2  {
3      for (i = 0; str[i] != '\0'; ++i)
4      {
5          if (isupper(str[i]))
6          {
7              str[i] = str[i] - 'A' + 'a';
8          }
9      }
10     printf("%s\n", str);
11 }
```

NULL是空指针。

gets函数返回一个指针变量。

使用scanf函数进行格式化输入

```
1  for (i = 0; i < n; ++i)
2  {
3          scanf(" (%d,%d)", &x, &y);
4          // ...
5  }
```

格式字符串开头处有一空格，用于跳过输入的若干个连续空白字符。

1 输入输出
- 组数不定的数据输入
- 格式化输入

2 逻辑判断

3 位运算

4 函数
- 重复计算
- 递归

5 二分查找

6 排序
- 多关键字排序
- qsort函数

7 数组
- 二维数组的遍历
- 使用数组简化遍历

8 字符串

9 高精度

分支结构

建议使用if – else if – else 结构。

多多使用else，而非一系列if的排列。易造成程序逻辑上的错误。

```
1  if (op == 0)
2  {
3          // do something
4  }
5  else if (op == 1)
6  {
7          // do something
8  }
9  else if (op == 2)
10 {
11         // do something
12 }
```

1 输入输出
- 组数不定的数据输入
- 格式化输入

2 逻辑判断

3 位运算

4 函数
- 重复计算
- 递归

5 二分查找

6 排序
- 多关键字排序
- qsort函数

7 数组
- 二维数组的遍历
- 使用数组简化遍历

8 字符串

9 高精度

位运算

x	y	$\sim$ x	x & y	x | y	x $\wedge$ y
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

位运算的应用

- 求2的p次幂：`1 << p`
- 将n的第i位[1]置为1：`n | (1 << i)`
- 将n的第i位置为0：`n & ( ~ (1 << i))`
- 将n的第i位翻转：`n` $\wedge$ `(1 << i)`
- 使用异或求一组数中只出现1次（或2次）的数字：E8-J

[1]从第0位开始计数

1 输入输出
- 组数不定的数据输入
- 格式化输入

2 逻辑判断

3 位运算

4 函数
- 重复计算
- 递归

5 二分查找

6 排序
- 多关键字排序
- qsort函数

7 数组
- 二维数组的遍历
- 使用数组简化遍历

8 字符串

9 高精度

重复计算：程序模块化

将函数视为**黑盒**，准确定义其功能并实现后，按需取用。

在函数需要复制多次相同代码时，建议将其定义为函数。

程序模块化：抽象为函数

C6-F：给定一个9×9九宫格，判断是否满足：每行每列各数只出现1次，且每个3×3小格中各数只出现1次。

程序模块化：抽象为函数

C6-F：给定一个9×9九宫格，判断是否满足：每行每列各数只出现1次，且每个3×3小格中各数只出现1次。

定义函数`int Judge(int x, int y, int op)`：从第x行、第y列开始判断，若不合法返回0，否则返回1。

- 若op为0，则判断整行
- 若op为1，则判断整列
- 若op为2，则判断以(x, y)为左上角的3×3方格

程序模块化：抽象为函数

```
1  valid = 1;
2
3  for (i = 1; i <= 9; ++i)
4  {
5          valid = valid && Judge(i, 1, 0);
6          valid = valid && Judge(1, i, 1);
7  }
8
9  for (i = 1; i <= 9; i += 3)
10         for (j = 1; j <= 9; j += 3)
11                 valid = valid && Judge(i, j, 3);
```

程序模块化：抽象为函数

逻辑清晰，功能明确，方便调试。

程序模块化：抽象为函数

逻辑清晰，功能明确，方便调试。

复习建议

使用函数完成C5-C题，体会函数的作用与优势。

题目大意：给定4个三维坐标点，输出它们两两之间的距离。

可准备一些常用函数，供考场上使用，例如排序、字符串查找等。

递归

编写递归函数时，要牢记函数的功能，对函数进行抽象。

复习建议

C7-I（字符串解压）
E5-H（2的幂次方）

1 输入输出
- 组数不定的数据输入
- 格式化输入

2 逻辑判断

3 位运算

4 函数
- 重复计算
- 递归

5 二分查找

6 排序
- 多关键字排序
- qsort函数

7 数组
- 二维数组的遍历
- 使用数组简化遍历

8 字符串

9 高精度

二分查找

注意死循环的问题，仔细思考循环终止条件。

复习建议

可自行编写查找有序数组中的元素的代码（函数）。
C8-F（二分查找）

1 输入输出
- 组数不定的数据输入
- 格式化输入

2 逻辑判断

3 位运算

4 函数
- 重复计算
- 递归

5 二分查找

6 排序
- 多关键字排序
- qsort函数

7 数组
- 二维数组的遍历
- 使用数组简化遍历

8 字符串

9 高精度

多关键字排序

在排序的判断条件中进行修改。

```
1  for (i = 0; i < n; i++)
2  {
3      for (j = 0; j < n - i; j++)
4      {
5          if (Arr[j][0] > Arr[j + 1][0] ||  \
6              (Arr[j][0] == Arr[j + 1][0] && Arr[j][1] < Arr[j + 1][1]))
7          {
8              // Swap Arr[j][0], Arr[j + 1][0]
9              // Swap Arr[j][1], Arr[j + 1][1]
10         }
11     }
12 }
```

qsort函数的使用

函数实现排序功能，使用者需提供比较函数。

函数使用格式：
qsort(待排序数组名称，元素个数，每个元素占用字节数，函数名)；

qsort函数的使用

```
1  int Arr[10000 + 7][2];
2  int Cmp (const void *a, const void *b)
3  {
4      int *pa = (int *)a;
5      int *pb = (int *)b;
6      if (pa[0] > pb[0])          return 1;
7      else if (pa[0] < pb[0])     return -1;
8      else
9      {
10         if (pa[1] > pb[1])      return 1;
11         else                    return -1;
12     }
13 }
14 qsort(Arr, n, sizeof(Arr[0]), Cmp);
```

qsort函数中的比较函数

比较函数定义了元素间的前后关系，qsort函数在判断时调用。

```
1  for (i = 0; i < n; i++)
2  {
3      for (j = 0; j < n - i; j++)
4      {
5          if (cmp(&Arr[i], &Arr[j]) > 0)
6          {
7              // Swap
8          }
9      }
10 }
```

qsort函数

复习建议

C7-F（成绩处理）：含有字符串的多关键字排序
E8-G（多关键字排序）：k个关键字的排序方法

仔细阅读题解，整理方法（资料可带入考场）。

1 输入输出
- 组数不定的数据输入
- 格式化输入

2 逻辑判断

3 位运算

4 函数
- 重复计算
- 递归

5 二分查找

6 排序
- 多关键字排序
- qsort函数

7 数组
- 二维数组的遍历
- 使用数组简化遍历

8 字符串

9 高精度

二维数组的遍历

按行遍历、按列遍历。

```
1  int Arr[10 + 7][10 + 7];
2  for (i = 0; i < n; ++i)
3  {
4          for (j = 0; j < m; ++j)
5          {
6                  printf("%d ", Arr[i][j])
7          }
8          printf("\n");
9  }
```

使用数组简化遍历

给定一个东、南、西、北的序列，请你按照序列遍历一二维数组。

定义delta数组以简化遍历：

```
1  const int delta[4][2] =
2  {
3          {0, 1},          // 0: east
4          {1, 0},          // 1: south
5          {0, -1},         // 2: west
6          {-1, 0}          // 3: north
7  };
```

使用数组简化遍历

```
1  const int delta[4][2] =
2  {
3          {0, 1},          // 0: east
4          {1, 0},          // 1: south
5          {0, -1},         // 2: west
6          {-1, 0}          // 3: north
7  };
```

使用变量x和y记录当前位置，若dir为输入的方向信息（0表示东、1表示南，以此类推），则每次行走可表示为：

$$\texttt{x} = \texttt{x} + \texttt{delta[dir][0]}$$

$$\texttt{y} = \texttt{y} + \texttt{delta[dir][1]}$$

使用数组简化遍历

复习建议

C6-F（应急食品）：按行、列、块遍历
E6-F（寻找五子相连）：对角线遍历
E6-H（循环填数）：顺时针旋转遍历

1 输入输出
- 组数不定的数据输入
- 格式化输入

2 逻辑判断

3 位运算

4 函数
- 重复计算
- 递归

5 二分查找

6 排序
- 多关键字排序
- qsort函数

7 数组
- 二维数组的遍历
- 使用数组简化遍历

8 字符串

9 高精度

字符串的基本操作

- 查找
- 替换
- 比较
- 排序

字符串的与数组等价，输入和输出有简便方法（%s）。

字符串的相关函数

- 字符串查找：`strstr(strA, strB)`
- 字符串比较：`strcmp(strA, strB)`
- 字符串复制：`strcpy(strA, strB)`

字符串的相关函数

- 字符串查找：strstr(strA, strB)
- 字符串比较：strcmp(strA, strB)
- 字符串复制：strcpy(strA, strB)

复习建议

重点注意函数返回值。

if (strcmp(strA, strB))是错误写法。

1 输入输出
- 组数不定的数据输入
- 格式化输入

2 逻辑判断

3 位运算

4 函数
- 重复计算
- 递归

5 二分查找

6 排序
- 多关键字排序
- qsort函数

7 数组
- 二维数组的遍历
- 使用数组简化遍历

8 字符串

9 高精度

高精度

如果数字超过long long类型所表示的范围，则需使用数组模拟计算过程。

复习建议

自行编写两个高精度数字的加法、减法、乘法的函数。

考点清单

1. 输入输出
   - 组数不定的数据输入
   - 格式化输入
2. 逻辑判断
3. 位运算
4. 函数
   - 重复计算与复杂遍历
   - 递归
5. 二分查找
6. 排序
   - 多关键字排序
   - qsort函数
7. 数组
   - 二维数组的使用
8. 字符串
   - 查找
   - 替换
   - 比较
   - 排序
9. 高精度