# C 语言复习资料

# 前言

这是 BUAA C 语言/程设复习资料, 由 "另一种" 小组 (一个传源书院学支学习互助小组) 的成员 (lzq, yzx, zjl) 共同整理而成, 仅供本校同学内部复习参考使用.

C 语言程序设计/程序设计基础对于刚进入大学的同学来说是一门上手较难的课程. 考试上机允许携带纸质资料, 以及可以访问电脑的本地代码, 所以我们可以存储一些常用的代码片段, 考试时直接参考.

本复习资料是 "另一种" 小组三位成员在开展学习互助活动中总结并整理各自的代码而形成的一些常见套路, 后经由 tsxb 修订. 希望能帮助各位在 C 语言/程设课程的期中/期末更好上分!

此外, 这一复习资料的形成也参考了许多其他资料, 我们会在资料的结尾注明. 感谢编写这些资料的大佬们 (特别是我们的助教们) 的付出!

首先是一个模板:

```c
#include <math.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

int main() {
    // Start Coding Here...
    return 0;
}
```

该模板包含了引用的头文件和 `main` 函数结构, 建议同学们善用自己的 IDE (如 vscode, Dev-Cpp, CLion, code::blocks 等) 的 "代码模板" 以及类似功能, 将该模板保存到这一功能当中, 在每次新建一个 c 语言文件时都能自动添加这些内容. 同时, 对于即将到来的上机, 练习赛或考试, 建议同学们提前准备好 10 或 11 个带模板的 c 语言文件 (视考试题量而定), 考前在编辑器中全部打开, 以便自己快速编写代码.

衷心祝愿每一位同学都能够在考试中取得自己理想的成绩!

# 代码基础

这里可以自行整理一些自己平常容易犯的错误, 比如多组数据输入相关:

```c
int T;
scanf("%d", &T);
while(T--) {
    int a, b, c;
    scanf("%d%d%d", &a, &b, &c);
}
```

以上就是定组数据输入的方法, 首先输入一个数据组数 `T`, 然后每组数据读入三个整型变量 `a,b,c`. 实际做题时, 每组的读入应当按照题目要求进行编写.

```
int a;
while(scanf("%d", &a) != EOF) {
    // TODO
}
```

以上是**不定组数据输入**的方法, 考试前应当熟练掌握不定组数据的输入: 输入数据之后敲回车, 再敲 `Ctrl+Z`, 再敲回车就可以结束输入. 注意, `while` 循环的条件中已经读入了每组数据的 1 个变量 `a`（实际做题时应当按照题目要求读入）, 一般循环中不用再次读入变量 `a`.

# 位运算

一般来说, 考试时不会出现太难的位运算操作题, 同学们掌握基础的位运算操作即可.

以下是位运算相关代码, 最低操作位为 0. 若未作说明, 则待操作数 x 均为 `unsigned int` 型, 能否转换为其他类型 (如 `(unsigned) long long`) 则视情况而定.

## 简单操作

```
/*获取 x 的第 k 位*/
a = (x >> k) & 1;
/*将 x 的第 k 位 置 1*/
x |= (1 << k);
/*将 x 的第 k 位 置 0*/
x &= ~(1 << k);
/*将 x 的第 k 位 取反*/
x ^= (1 << k);
/*将 x 最低的为 0 的位 置 1*/
x |= (x + 1);
/*将 x 最低的为 1 的位 置 0*/
x &= (x - 1);
/*求 x 最低的为 1 的位, 如 6 (2 进制下为 110) 则输出 2 (2 进制下为 10)*/
lowbit = (x & (-x));
/*将 x 的高 16 位与低 16 位对调*/
x = (x << 16) | (x >> 16);
/*对于 int 型, 检测 x 和 y 是否具有相反的符号, 若为 0 则相同, 否则相反*/
f = ((x ^ y) < 0);
/*确定 x 是否为 2 的幂, 即二进制表示中只有一个 1*/
f = x && !(x & (x - 1));
```

```
/*根据掩码 mask 合并 x 和 y 的位，掩码中每一位为 0 时选择 x 的位，1 则选择 y 的位*/
res = (x ^ ((x ^ y) & mask));
```

# 函数操作

```
/*求汉明权重，即 x 的二进制表示下 1 的个数*/
int popcount(unsigned int x) {
    int cnt;
    for(cnt = 0; x; ++cnt) x &= x - 1;
    return cnt;
}
```

```
/*计算 x 的奇偶校验码，有偶数个 1 则返回 0，否则返回 1*/
int parity(unsigned int x) {
    unsigned int p;
    for(p = 0; x; p = !p) x &= x - 1;
    return (p != 0);
}
```

```
/*输出 x 的二进制表示*/
void write_binary(unsigned int x) {
    int i, s = sizeof(x) * CHAR_BIT - 1;
    for(i = s; i >= 0; --i ) {
        putchar(((x >> i) & 1) + '0');
    }
}
```

# 进制转换

From 2024 级航类 C 语言 C3-I

```
// n 进制数转十进制，输入字符数组为 n 进制数
long long x2dec(char x[], int n)
{
    int len = strlen(x);
    long long t = 0;
    for (int i = 0; i < len; i++) {
        int w = x[i];
        if ('0' <= w && w <= '9')
            w -= 48;
        else
```

```
            w -= 55;
        t = t * n + w;
    }
    return t;
}

char sta[100005];

// 十进制转 m 进制
void dec2x(long long t, int m)
{
    if (t == 0) {
        printf("0\n");
    } else {
        int top = 0;
        while (t > 0) {
            int u = t % m;
            if (u <= 9)
                u += 48;
            else
                u += 55;
            top++;
            sta[top] = u;
            t /= m;
        }
        // 倒序输出
        while (top > 0) {
            printf("%c", sta[top]);
            top--;
        }
        printf("\n");
    }
}
```

# 数学相关

## 辗转相除法

```
int gcd(int a, int b) {
    return b ? gcd(b, a % b) : a;
}
```

## 素数相关

### 判断是否为质数

```
/*如果是质数则返回 1, 否则返回 0. 时间复杂度为 O(sqrt(n))*/
int isPrime(int x) {
    if(x == 1) return 0;
    if(x % 2 == 0) return 0;
    int i;
    for(i = 3; i * i <= x; i += 2) {
        if(x % i == 0) return 0;
    }
    return 1;
}
```

## 求 x 的所有素因子

```
/*数组 p 存储 x 的所有素因子, cnt 为素因子的个数*/
int p[10005];
int cnt = 0;
long long t = x, i;
for(i = 2; i * i <= t; ++i) {
    if(t % i == 0) {
        p[cnt++] = i;
        while(t % i == 0)
            t /= i;
    }
}
if(t != 1LL) p[cnt++] = t;
```

## 欧拉筛法求素数

```
/*数组 vis 存储 x 范围内每个数是否为质数的结果, 为 1 则是质数, 否则不是; 数组 pri 存储 x
范围内所有的质数, cnt 表示 x 范围内质数的数量. 时间复杂度为 O(n)*/
int vis[100], pri[100], cnt;
void init(int n) {
    int i, j;
    for(i = 2; i <= n; ++i) {
        if(!vis[i]) {
            pri[cnt++] = i;
        }
        for(j = 0; j < cnt; ++j) {
            if (1ll * i * pri[j] > n) break;
            vis[i * pri[j]] = 1;
            if (i % pri[j] == 0) break;
        }
    }
}
```

# 快速幂

```
/*快速求 a 的 b 次方模 p 的结果. 时间复杂度为 O(logN)*/
long long qpow(long long a, unsigned long long b, long long p) {
    long long ans = 1;
    a = a % p;
    while(b) {
        if(b & 1)
            ans = (ans * a) % p;
        b >>= 1;
        a = a * a % p;
    }
    return ans;
}
```

# 算法模板

# 输出一段能够运行并输出给定字符画的代码

即 2023 级程设 E7-C 题代码. 可能对第一题的快速解决有用处.

```
#include <stdio.h>
int main() {
    int i;
    char s[505];
    printf("#include<stdio.h>\nint main()\n{\n");
    while(gets(s)) {
        printf(" printf(\"");
        for(i = 0; s[i]; ++i) {
            if(s[i] == '\\' || s[i] == '\'' || s[i] == '\"')
                printf("\\");
            else if(s[i] == '%')
                printf("%%");
            putchar(s[i]);
        }
        printf("\\n\");\n");
    }
    printf(" return 0;\n}");
    return 0;
}
```

# 二分答案

适用于一些所有可能的答案为单调的问题, 如求一个一元三次方程在一段单调区间上的近似解等. 其实就是二分法. 可以用来解决:

1. 最大值中的最小值
2. 最小值中的最大值
3. 满足条件的最大值/最小值

```
/*如求一元三次方程的解*/
while(r - l > eps) {
    if(check(mid) > 0) r = mid;
    else l = mid;
    mid = (l + r) / 2;
}
```

这里的 `check` 函数就相当于求解一元三次方程中的 $f(x)$.

# 手搓二分查找

### 二分查找

```
/*在数组 arr 从下标 l 到 r 的范围中查找 key, 找到了返回对应的下标, 否则返回 -1*/
int binary_search(int key, int l, int r) {
    int ret = -1;
    int mid;
    while (start <= end) {
        mid = start + ((end - start) >> 1);
        if (arr[mid] < key)
            start = mid + 1;
        else if (arr[mid] > key)
            end = mid - 1;
        else {
            ret = mid;
            break;
        }
    }
    return ret;
}
```

### 查找首个不小于/大于给定值的元素

```
/*在数组 arr 从下标 l 到 r 的范围中查找首个不小于 key 的元素, 找到了返回对应的下标, 否则
返回 -1*/
int lower_bound(int key, int l, int r) {
```

```
    while(l < r) {
        int mid = l + (r - l) / 2;
        if(arr[mid] >= key) r = mid;
        else l = mid + 1;
    }
    return arr[l] >= key ? l : -1;
}

/*在数组 arr 从下标 l 到 r 的范围中查找首个大于 key 的元素，找到了返回对应的下标，否则返回 -1*/
int upper_bound(int key, int l, int r) {
    while(l < r) {
        int mid = l + (r - l) / 2;
        if(arr[mid] > key) r = mid;
        else l = mid + 1;
    }
    return arr[l] > key ? l : -1;
}
```

# 冒泡排序

这也要写个板子?

```
/*冒泡排序，使用时传入数组名，开始排序的下标，以及要排序的长度。*/

void bubbleSort(int arr[], int l, int n) {
    for (int i = l; i < n; ++i) {
        for (int j = i + 1; j < n; ++j) {
            if (arr[i] > arr[j]) {
                int tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
            }
        }
    }
}

/*例如，对数组 a 从 a[0] 到 a[n-1] 排序，调用如下:*/
bubbleSort(a, 0, n);
```

# 日期相关

```
/*星期数组*/
const char *week[7] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"};

/*蔡勒公式*/
```

```c
void zeller(int date) {
    int year, month, day;
    year = date / 10000;
    month = date / 100 % 100;
    day = date % 100;
    if(month < 3) {
        month += 12;
        year -= 1;
    }
    int c = year / 100;
    int y = year % 100;
    int D = c / 4 - 2 * c + y + y / 4 + (13 * (month + 1) / 5) + day - 1;
    int W = (D % 7 + 7) % 7;
    printf("%s\n", week[W]);
}

/*月份天数数组*/
const int mon[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int isLeapYear(int year) {
    if(((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)) {
        return 1;
    } else {
        return 0;
    }
}

/*判断八位数字日期是否合法. 返回 1 为合法, 0 为不合法*/
int isValidDate(int date) {
    int year, month, day;
    year = date / 10000;
    month = date / 100 % 100;
    day = date % 100;
    if(day < 0 || day > 31 || month < 1 || month > 12) return 0;
    if(isLeapYear(year)) {
        if((month == 2 && day > 29) || (month != 2 && day > mon[month])) return 0;
    } else {
        if(day > mon[month]) return 0;
    }
    return 1;
}
```

# 后记

这份资料仅仅涵盖了位运算, 日期/时间计算, 和一些可能用到的算法, 对于我们学过的递归等没有提及, 因为递归等算法没有什么套路可循, 需要我们在解题过程中自己思考.

编写这份复习资料的过程中, 我们小组参考了许多外部资料, 列举如下, 以表感谢.

1. 2023 年信息类程设题解, 由我们的助教编写;

2. 网页 Bit Twiddling Hacks
3. 网站 洛谷
4. 网站 OI-Wiki

最后, 如果你发现这份资料有什么错误, 或有什么可以改进的地方的话, 欢迎发送邮件到 tsxb 的邮箱 提建议. 感谢各位斧正!