中山大學
SUN YAT-SEN UNIVERSITY

指针

中山大学计算机学院

讲课人：万海

目录

CONTENTS

01 输出自身

02 函数返回数组

03 数组指针传递

04 泛型编程

输出自身

编写一个程序，输出自身代码

假设你写的程序A是

```
#include <stdio.h>
int main
{
        printf(“hello”);
        return 0;
}
```

那么程序A需要A的源代码，即输出上面这段代码。

代码输出展示

```
char a[]="#include <stdio.h>\n"
"int main()\n"
"{\n"
"    char *p=a;\n"
"    printf(\"char a[]=\\\"\");\n"
"    while(*p)\n"
"    {\n"
"        switch(*p)\n"
"        {\n"
"            case'\\\"':printf(\"\\\\\\\"\");break;\n"
"            case'\\\\':printf(\"\\\\\\\\\");break;\n"
"            case'\\n':printf(\"\\\\n\\\"\\n\\\"\");break;\n"
"            default:putchar(*p);\n"
"        }\n"
"        p++;\n"
"    }\n"
"    printf(\"\\\";\\n\");\n"
"    fwrite(a, sizeof(a)-1, 1, stdout);\n"
"    return 0;\n"
"}\n"
"";
```

```
#include <stdio.h>
int main()
{
    char *p=a;
    printf("char a[]=\"");
    while(*p)
    {
        switch(*p)
        {
            case'\"':printf("\\\"");break;
            case'\\':printf("\\\\");break;
            case'\n':printf("\\n\"\n\"");break;
            default:putchar(*p);
        }
        p++;
    }
    printf("\";\n");
    fwrite(a, sizeof(a)-1, 1, stdout);
    return 0;
}
```

函数返回数组

输入一个字符串和一个整数w，将字符串循环左移w位，输出原串和修改后的字符串

输入样例
abcd 3

输出样例
abcd
dabc

```
char *solve_correct(char *st, int delta)
{
    int len=strlen(st);
    char
*ans=(char*)malloc(sizeof(char)*(len+1));
    for (int i=0; i<len; ++i)
        ans[i]=st[(i+delta)%len];
    ans[len]='\0';
    return ans;
}
```

```
char *solve_wrong(char *st, int delta)
{
    int len=strlen(st);
    char ans[len+1];
    for (int i=0; i<len; ++i)
        ans[i]=st[(i+delta)%len];
    ans[len]='\0';
    return ans;
}
```

```
int main()
{
    char a[110]={'\0'};
    int delta;
    scanf("%s %d", a, &delta);
    char *ans=solve_wrong(a, delta);
    printf("origin: %s\n", a);
    printf("new: %s\n", ans);
    ans=solve_correct(a, delta);
    printf("origin: %s\n", a);
    printf("new: %s\n", ans);
    return 0;
}
```

输出结果

origin: abcd
new: (null)
origin: abcd
new: dabc

数组指针传递

输入一个字符串和一个整数w，将字符串循环左移w位，输出原串和修改后的字符串

输入样例
abcd 3

输出样例
abcd
dabc

```
void solve_correct(char *st, int delta,
char **ans)
{
    int len=strlen(st);
 *ans=(char*)malloc(sizeof(char)*(len+1));
    for (int i=0; i<len; ++i)
        (*ans)[i]=st[(i+delta)%len];
    (*ans)[len]='\0';
}
```

```
void solve_wrong(char *st, int delta,
char *ans)
{
    int len=strlen(st);
 ans=(char*)malloc(sizeof(char)*(len+1));
    for (int i=0; i<len; ++i)
        ans[i]=st[(i+delta)%len];
    ans[len]='\0';
}
```

```
int main()
{
    char a[110]={'\0'}, *ans=NULL;
    int delta;
    scanf("%s %d", a, &delta);
    solve_wrong(a, delta, ans);
    printf("origin: %s\n", a);
    printf("new: %s\n", ans);
    solve_correct(a, delta, &ans);
    printf("origin: %s\n", a);
    printf("new: %s\n", ans);
    return 0;
}
```

输出结果
origin: abcd
new: (null)
origin: abcd
new: dabc

泛型编程

输入一个有n个元素的数组，以及一个符号，根据符号对数组排序，并输出排序后的结果。

输入样例

9 <

1 3 2 6 8 4 3 5 1

输出样例

1 1 2 3 3 4 5 6 8

输入样例

9 >

1 3 2 6 8 4 3 5 1

输出样例

8 6 5 4 3 3 2 1 1

```
void sort(void *a, int n, int base, int (*cmp)(const void *, const void *))
{
    char buffer[base];
    for (int i=0; i<n-1; ++i)
        for (char *cur=(char*)a; cur<(char*)a+(n-i-1)*base; cur+=base)
            if (cmp(cur+base, cur))
            {
                memcpy(buffer, cur, base);
                memcpy(cur, cur+base, base);
                memcpy(cur+base, buffer, base);
            }
}
```

封装好一个排序函数，用户调用时只需要提供待排序的数组指针，排序个数，数组元素所占字节数，比较函数。

Void 类型指针清除数据类型信息，仅保留地址

用户根据需求写好比较函数

```
int main()
{
    int n;
    char ch;
    scanf("%d %c", &n, &ch);
    int *a=(int *)malloc(sizeof(int)*n);
    for (int i=0; i<n; ++i) scanf("%d", &a[i]);
    sort(a, n, sizeof(int), (ch=='<'?
smaller:bigger));
    for (int i=0; i<n; ++i) printf("%d ", a[i]);
    puts("");

    double *b=(double *)malloc(sizeof(double)*n);
    for (int i=0; i<n; ++i) b[i]=a[i]+0.5;
    sort(b, n, sizeof(double), (ch=='<'?
smaller_d:bigger_d));
    for (int i=0; i<n; ++i) printf("%.2lf ", b[i]
    return 0;
}
```

```
int smaller(const void *b, const
void *c)
{
    return *(int*)b<*(int*)c;
}
int bigger(const void *b, const void
*c)
{
    return *(int*)b>*(int*)c;
}
int smaller_d(const void *b, const
void *c)
{
    return *(double*)b<*(double*)c;
}
int bigger_d(const void *b, const
void *c)
{
    return *(double*)b>*(double*)c;
}
```