

10. 将字符串中的所有字母都替换成该字母的下一个字母

```
#include <iostream>

using namespace std;

#include <ctype.h>

#include <stdio.h>

#include <string.h>

void func(char *p);

void main()

{

    char str1[20];

    printf("enter:");

    gets(str1);

    func(str1);

    puts(str1);

}

void func(char *p)

{

    char ch;

    while(*p)

    {

        ch=*p;

        if(isalpha(*p)&&(*p!='z')&&(*p!='Z'))

            *p=ch+1;

        else if(*p=='z')

            *p='a';

        else if(*p=='Z')

            *p='A';

        p++;

    }

}
```

}

}

11. 回文判断

```
#include<iostream>
using namespace std;

bool func(int m);
void main()
{
    int m;
    cout<<"enter a number:"<<endl;
    cin>>m;
    cout<<func(m)<<endl;
}
bool func(int m)
{
    int i,n=0;
    i=m;

    while(i)
    {
        n=n*10+i%10;
        i/=10;
    }
    if(m==n)
        return true;
    return false;
}
```

```

#include<iostream>
using namespace std;
#include<string.h>
bool is_huiwen(char a[],int length)
{
    const char *src=a;
    const char *end;
    end=src+length-1;
    while(src<end)
    { if(*src==*end)
      { src++;end--;}
      else return false;
    }

    return true;
}

int main()
{ int len;
  char c[10];
  cout<<"enter:"<<endl;
  cin>>c;
  len=strlen(c);
  bool h;
  h=is_huiwen(c,len);
  if(h) cout<<"hui_wen"<<endl;
  else cout<<"non_hui_wen"<<endl;
  return 0;
}

```

12. 将一个“1234”的字符串转化为1234整型

```
#include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

int func(char a[]);

void main()

{

    char a[]={'1','2','3','4','\0'};

    cout<<func(a)<<endl;

}

int func(char a[])

{

    int i=0;

    int sum=0;

    while(a[i]!='\0')

    {

        sum=sum*10+(a[i]-'0');

        i++;

    }

    return sum;

}
```

13. 求一个二维数组每列的最小值

```
#include <iostream>

using namespace std;

void fun(int input[3][4], const int m, const int n, int output[4]);

int main()
{
    int input[3][4] = { {21,48,86,92},
                        {10,23,12,69},
                        {46,78,49,13} };

    int output[4];

    fun(input, 3, 4, output);

    cout<<"原二维数组是: "<<endl;

    for(int i=0; i<3; i++)
    {
        for(int j=0; j<4; j++)
        {
            cout<<input[i][j]<<" ";
        }
        cout<<endl;
    }

    cout<<"每列最小值是: "<<endl;

    for (int k=0; k<4; k++)
```

```

    {

        cout<<output[k]<<" ";

    }

    cout<<endl;


    return 0;

}


void fun(int input[3][4], const int m, const int n, int output[4])
{

    int i = 0;

    int j = 0;


    for (i=0; i<n; i++)
    {

        output[i] = input[0][i];

        for (j=0; j<m; j++)
        {

            if (output[i] > input[j][i])
            {

                output[i] = input[j][i];

            }

        }

    }

}

```

14.连续字符统计 (如AABCCCD: A2B1C3D1)

```
#include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

void func(char str[],int len);

void main()

{

    char str[20];

    int len;

    cout<<"enter:";

    gets(str);

    len=strlen(str);

    func(str,len);

}

void func(char str[], int len)

{

    int count=1;

    int i;

    for(i=0;i<len;i++)

    {

        if(str[i]==str[i+1])

            count++;

        else

        {

            cout<<str[i]<<count;

            count=1;

        }

    }

}
```



```
}
```

```
}
```

```
cout<<endl;
```

```
}
```

15.找出一个字符串中是否包含相同的子字符串（要求子串长度大于等于2）

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int fun(char* str, int n)
```

```
{
```

```
    char *temp = new char[n+2];
```

```
    char *str2 = str;
```

```
    int sum = 0;
```

```
    int outsum = 0;
```

```
    char* t;
```

```
    for(int i = 0; i < n; i++)
```

```
    {
```

```
        for(int j = 2; j <= n-i; j++)
```

```
        {
```

```
            sum = 0;
```

```
            str2 = str;
```

```
            memset(temp, 0, (n+2)*sizeof(char));
```

```
            strncpy(temp, str+i, j);
```

```
            while((t = strstr(str2, temp)) != NULL)
```

```
            {
```

```
                sum++;
```

```
                str2 = t+j;
```

```
            }
```

```

        if( sum > outsum)
        {
            outsum = sum;
        }
    }
}

if(outsum == 1)
    return 0;

return outsum;

}

int main()
{
    char strstr[1000];
    memset(strstr,0,sizeof(strstr));
    char *s = strstr;
    cin >> s;
    int n =strlen(s);
    int outsum;
    outsum = fun(s,n);
    cout << outsum << endl;

    system("pause");
    return 0;
}

```

16. 已知: yi er san si wu liu qi ba jiu 分别对应123456789, 对一段只含有这几种字符的字符串进行转换, 转换成相应的数字

如: yiersansan: 1233

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
char* sss[9] = {"yi", "er", "san", "si", "wu", "liu", "qi", "ba", "jiu"};
```

```
int fun(char* str)
```

```
{
```

```
    int i;
```

```
    int sum = 0;
```

```
    int d = 0;
```

```
    i = 0;
```

```
    int j;
```

```
    while(str[i] != '\0')
```

```
    {
```

```
        if(str[i] == 'y' || str[i] == 'e' || str[i] == 'w' || str[i] == 'q' || str[i] == 'b')
```

```
            d = 2;
```

```
        else if(str[i] == 'l' || str[i] == 'j')
```

```
            d = 3;
```

```
        else if(str[i] == 's')
```

```
        {
```

```
            if(str[i+1] == 'a')
```

```

                d = 3;

                else d = 2;

            }

        for(int k = 0; k < 9; k++)

            if(strncmp(str+i,sss[k],d) == 0)

                j = k+1;

        sum = 10*sum + j;

        i = i+d;

    }

    return sum;

}

int main()

{

    char strstr[1000];

    memset(strstr,0,sizeof(strstr));

    char *s = strstr;

    cin >> s;


    int outsum;

    outsum = fun(s);


    cout << outsum << endl;


    system("pause");

    return 0;

```

}

```

#include <iostream>
#include <string>

using namespace std;

char* fun(char* str, int n)
{
    int hash[256] = {0};
    char *result = new char[n+1];
    memset(result,0,(n+1)*sizeof(char));
    for(int i = 0; i < n; i++)
    {
        hash[str[i]]++;
    }
    int min = 0x7fffffff;

    for(int i = 0; i < 256; i++)
    {
        if(hash[i] < min && hash[i] != 0)
            min = hash[i];
    }

    for(int i = 0,j = 0; i < n; i++)
    {
        if(hash[str[i]] != min)
        {
            result[j++] = str[i];
        }
    }

    return result;
}

int main()
{
    char strstr[1000];
    memset(strstr,0,sizeof(strstr));
    char *s = strstr;
    cin >> s;

    char *re = NULL;

    re = fun(s,strlen(s));

    cout << re << endl;
    system("pause");
    return 0;
}

```

1. 找出一个数组中满足 2^N 的元素

```
#include <iostream>

using namespace std;

int find(int a[],int len);

void main()
{
    int a[]={1,2,3,5,7,8,16};
    int len=sizeof(a)/sizeof(int);
    cout<<find(a,len)<<endl;
}

int find(int a[],int len)
{
    int i;
    int count=0;
    for(i=0;i<len;i++)
    {
        if(0==(a[i]&(a[i]-1)))
            count++;
    }
    return count;
}
```



```

#include <iostream>
using namespace std;
void func(int n, int m, int s, int a[]);
void main()
{
    int a[9]={0};
    func(9,3,1,a);
    for(int i=8;i>=0;i--)
        cout<<a[i]<<" ";
    cout<<endl;
}
void func(int n, int m, int s, int a[])
{
    int s1;
    int w;

    s1=s;
    for(int k=0;k<n;k++)
    {
        a[k]=k+1;
    }
    for(int i=n;i>=2;i--)
    {
        s1=(s1+m-1)%i;
        if(s1==0)
            s1=i;
        w=a[s1-1];

        for(int j=s1;j<i;j++)
            a[j-1]=a[j];
        a[i-1]=w;
    }
}
}

```

2.报数：共n个人 从1编号，设从第s个人报号，报到m出队

```
#include<iostream>

using namespace std;

void Joseph(int n, int m, int s);

int main()
{

    Joseph(9,3,1);

    return 0;
}

void Joseph(int n, int m, int s)
{
    int i,j,w;
    int s1 = s;
    int a[100] = {0};

    for(i = 0; i < n; i++)                //把n个人的序号放入数组a[]中;
    {
        a[i] = i + 1;
    }

    for(i = n; i >= 2; i--)
    {
        s1 = (s1+m-1)%i;                //s1每次出圈人的位置
    }
}
```

if(s1 == 0)	//如果s1等于0，则说明要开始报数的人是最后一个人
{	
s1 = i;	//把此时变量i的值赋给s1
}	
w = a[s1-1];	//把每次出圈人的序号赋给w
for(j = s1; j < i; j++)	
{	
a[j-1] = a[j];	
}	
a[i-1] = w;	//把每次出圈人的序号赋给倒数第i个位置上
}	
for(int k = n-1; k >= 0; k--)	
cout<<a[k]<<" ";	
cout<<endl;	
}	

3.统计一个数二进制表达中0的个数（首位1之前0不计）

```
#include <iostream>
```

```
using namespace std;
```

```
int fun(int num);
```

```
int main()
```

```
{
```

```
    int num;
```

```
    cout<<"Please enter a integer:\n";
```

```
    cin>>num;
```

```
    cout<<fun(num)<<endl;
```

```
    return 0;
```

```
}
```

```
int fun(int num)
```

```
{
```

```
    int count = 0;
```

```
    int i = 0;
```

```
    while (num)
```

```
    {
```

```
        if (num & 1)
```

```
        {
```

```
            count++;
```

```
        }
```

```
num = num >> 1;
```

```
i++;
```

```
}
```

```
return (i-count);
```

```
}
```

4. 镜像反转二进制表达式，并输出十进制值

```
#include<iostream>

using namespace std;

int func(int a);

main()
{
    int n;
    cout<<"enter:";
    cin>>n;
    cout<<func(n)<<endl;
}

int func(int a)
{
    int val=0;
    int temp;
    int i;
    int n=0;
    int b[100];
    while(a!=0)
    {
        temp=(a&1);
        b[n++]=temp;
        a=(a>>1);
    }
    for(i=0;i<n;i++)
        val=val*2+b[i];

    return val;
}
```

}

5.判断一个字符串中 () 是否配对

```
#include<iostream>

using namespace std;

bool match(char a[],int length);

int main()
{
    char b[100];
    int len;
    bool m;
    cout<<"enter:"<<endl;
    gets(b);
    len=strlen(b);
    m=match(b,len);
    if(m) cout<<"match"<<endl;
    else cout<<"nonmatch"<<endl;
    return 0;
}

bool match(char a[],int length)
{
    char *p=a;
    int count1=0;
    int count2=0;

    while(*p!='\0')
    {
        if(*p=='(') count1++;
        if(*p==')') count2++;
        if(count2>count1)
```



```
        return false;
```

```
        p++;
```

```
    }
```

```
    if(count1==count2)
```

```
        return true;
```

```
    else
```

```
        return false;
```

```
}
```

7. 查找子字符串个数

```
#include <iostream>

#include <string>

using namespace std;

int fun(char *str, char *substr);

int main()
{
    char str[100];
    char substr[10];
    cout<<"输入字符串: \n";
    gets(str);
    cout<<"输入子串: \n";
    gets(substr);
    cout<<fun(str, substr)<<endl;

    return 0;
}

int fun(char *str, char *substr)
{
    int count = 0;

    while (*str)
    {
        char *p = str;
        char *q = substr;
```

```
while (*q)
{
    if (*p == *q)
    {
        p++;
        q++;
    }
    else
    {
        break;
    }
}

if (*q == 0)
{
    str = str + strlen(substr);
    count++;
}
else
{
    str++;
}
}

return count;
}
```

8.关于数组的循环移位

```
#include <iostream>

using namespace std;

void func(int *p, int n, int k);

void main()
{
    int a[]={1,2,3,4,5};
    int i;
    func(a,5,2);

    for(i=0;i<5;i++)
        cout<<a[i]<<" ";
    cout<<endl;
}

void func(int *p ,int n, int k)
{
    int temp;
    int i;
    k=k%n;
    if(k>=0)
    {
        while(k)
        {
            temp=p[n-1];
            for(i=n-1;i>0;i--)
                p[i]=p[i-1];
            p[0]=temp;
            k--;
        }
    }
}
```

```

        }

    }
    else if(k<0)
    {
        k=k*(-1);
        while(k)
        {
            temp=p[0];
            for(i=1;i<n;i++)
                p[i-1]=p[i];
            p[n-1]=temp;
            k--;
        }
    }
}

```

```

#include <iostream>

#include <stdio.h>

#include <string.h>

#include <conio.h>

using namespace std;

typedef struct student
{
    int data;
    struct student *next;
} node;

/* 链表的创建*/

node *creat()
{
    node *head,*p,*s;
    int x;
    int cycle=1;
    head=(node*)malloc(sizeof(node));
    p=head;
    while(cycle)
    {
        cout<<"please enter a number"<<endl;
        cin>>x;
        if(x!=0)
        {
            s=(node*)malloc(sizeof(node));
            s->data=x;
            p->next=s;

```

```

        p=s;    //前一个节点的地址

    }

    else

        cycle=0;

}

head=head->next;

p->next=NULL;

return (head);

}


void reverse(node **head)
{

    node *p1,*p2,*p3;

    //if(*head==NULL||*head->next==NULL) return *head;

    p1=*head;

    p2=p1->next;

    while(p2)

    {
        p3=p2->next;

        p2->next=p1;

        p1=p2;

        p2=p3;

    }

    (*head)->next=NULL;

    *head=p1;

    //return *head;

}

```

```

void main()
{
    node *p;
    node *head;
    //int n;
    head=(node*)malloc(sizeof(node));
    head=creat();
    //n=length(head);
    p=head;
    cout<<"原始链表是： ";
    if(head!=NULL)                                     //原始链表的打印
        while(p!=NULL)
        {
            cout<<p->data<<" ";
            p=p->next;
        }
        cout<<endl;

    node **q = &head;
    reverse(q);
    p=head;
    //      cout<<"链表长度为： "<<n;

    /*****/

    cout<<"逆置后链表是： ";
    if(head!=NULL)                                     //逆置后链表的打印
        while(p!=NULL)

```



```
{  
  
    cout<<p->data<<" ";  
  
    p=p->next;  
  
}  
  
cout<<endl;
```

```
}
```

```

#include <iostream>

#include <stdio.h>

#include <string.h>

#include <conio.h>

using namespace std;

typedef struct student
{
    int data;
    struct student *next;
} node;

/* 链表的创建*/

node *creat()
{
    node *head,*p,*s;
    int x;
    int cycle=1;
    head=(node*)malloc(sizeof(node));
    p=head;
    while(cycle)
    {
        cout<<"please enter a number"<<endl;
        cin>>x;
        if(x!=0)
        {
            s=(node*)malloc(sizeof(node));
            s->data=x;
            p->next=s;

```

```

                p=s;

            }

            else

                cycle=0;

        }

        head=head->next;

        p->next=NULL;

        return (head);

}

```

/*链表的测长*/

```
int length(node *head)
```

```

{

    int n=0;

    node *p;

    p=head;

    while(p!=NULL)

    {

        p=p->next;

                n++;

    }

    return n;

```

```

}

```

/*链表的排序*/

```
node *sort(node* head, int n)
```

```

{

    int i;

    int j;

    int temp;

    node *p;

    if(head==NULL||head->next==NULL)

        return (head);

    p=head;

    for(i=1;i<n;i++)

    {

        p=head;

        for(j=0;j<n-i;j++)

        {

            if(p->data>p->next->data)

            {

                temp=p->next->data;

                p->next->data=p->data;

                p->data=temp;

            }

            p=p->next;

        }

    }

    return (head);

}

```

/*链表的逆置*/

```

node *reverse(node* head)
{
    node *p1;
    node *p2;
    node *p3;
    if(head==NULL || head->next==NULL)
        return head;
    p1=head;
    p2=p1->next;
    while(p2)
    {
        p3=p2->next;
        p2->next=p1;
        p1=p2;
        p2=p3;
    }
    head->next=NULL;
    head=p1;
    return head;
}

```

/*链表插入一个节点*/

```

node *insert(node* head,int num)
{
    node *p0,*p1,*p2;
    p1=head;
    p0=(node*)malloc(sizeof(node));
    p0->data=num;

```

```

while((p0->data)>(p1->data)&& p1->next!=NULL)
{
    p2=p1;
    p1=p1->next;
}
if(p0->data<=p1->data)
{
    if(head==p1)                                //插入的是头结点
    {
        p0->next=p1;
        head=p0;
    }
    else
    {
        p2->next=p0;                            //插入的是中间结点
        p0->next=p1;
    }
}
else                                            //插入的是尾结点
{
    p1->next=p0;
    p0->next=NULL;
}

return (head);
}

/*链表删除一个节点*/
node *del(node* head,int num)
{

```

```

node *p1,*p2;

p1=head;

while (num!=p1->data&& p1->next!=NULL)
{
    p2=p1;
    p1=p1->next;
}

if (num==p1->data)
{
    if (head==p1)
    {
        head=p1->next;
        free(p1);
    }
    else
        p2->next=p1->next;
}

else

cout<<num<<"couldn't be found"<<endl;

return (head);
}

```

```

void main()
{
    node *p;
    node *head;

    int n;

    head=(node*)malloc(sizeof(node));

```

```

head=creat();
n=length(head);
p=head;
cout<<"原始链表是： ";
if(head!=NULL)                                     //原始链表的打印
    while(p!=NULL)
    {
        cout<<p->data<<" ";
        p=p->next;
    }
    cout<<endl;

    head=reverse(head);
p=head;
    cout<<"链表长度为： "<<n;

/*****/

cout<<"逆置后链表是： ";
if(head!=NULL)                                     //逆置后链表的打印
    while(p!=NULL)
    {
        cout<<p->data<<" ";
        p=p->next;
    }
    cout<<endl;

/*****/

head=sort(head,n);
p=head;

```



```

        cout<<"排序后链表： ";

if(head!=NULL)                                     //排序后链表的打印

    while(p!=NULL)

    {

        cout<<p->data<<" ";

        p=p->next;

    }

    cout<<endl;

/*****/

    int a;

    cout<<"insert a number:"<<endl;

    cin>>a;

    head=insert(head,a);

    p=head;

    cout<<"插入后链表是： ";

if(head!=NULL)                                     //插入后链表的打印

    while(p!=NULL)

    {

        cout<<p->data<<" ";

        p=p->next;

    }

    cout<<endl;

/*****/

    int b;

    cout<<"delete number:"<<endl;

    cin>>b;

    head=del(head,b);

    p=head;

    cout<<"删除后链表是： ";

```

```
if(head!=NULL)                                     //删除后链表的打印

    while(p!=NULL)

    {

        cout<<p->data<<" ";

        p=p->next;

    }

    cout<<endl;

}
```

```

//超大整数加法运算

//#include "stdafx.h"

#include <string>

#include<iostream>

using namespace std;

#define ln 100 //数字长度


int main(int argc, char* argv[])
{
    int la = 0, lb = 0;

    int A[ln];

    int B[ln];

    for(int l=0; l<ln; l++)
    {
        A[l]=0;
        B[l]=0;
    }


    char a[ln];

    cout << "输入一个高精度数(小于100位) 作被加数:" << endl;

    cin.getline(a, ln);

    la =strlen(a);

    for (int i=0; i<la; i++)
    {
        A[i] = int(a[la-1-i])-48;
    }


    char b[ln];

    cout << "输入另一个高精度数(小于100位) 作加数:" << endl;

```

```

cin.getline(b, 1n);
lb =strlen(b);
for (i=0; i<lb; i++)
{
    B[i] = int(b[lb-1-i])-48;
}

int lc;
if (la>lb)
{
    lc = la;
}
else
    lc = lb;

for (int j=0; j<lc; j++)
{
    A[j] = A[j]+B[j];
    A[j+1] = A[j+1]+A[j]/10;
    A[j] = A[j]%10;

}
while (A[lc-1]>10)
{
    A[lc]=A[lc-1]/10;
    lc += 1;

}
cout << "相加结果: " << endl;

```

```
for (j=lc-1; j>=0; j--)  
{  
    cout << A[j];  
}  
cout << endl;  
  
return 0;  
}
```