Bài 1a. Viết chương trình nhập vào 1 chuỗi và 1 ký tự, kiểm tra ký tự có trong chuỗi hay không, nếu có đưa ra số lần xuất hiện của ký tự đó trong chuỗi.

Ví dụ: Chuỗi = “Nhap mon lap trinh” ; ký tự = ‘a’

In ra: Có ký tự ‘a’ trong chuỗi. Số lần xuất hiện = 2

```
int KT_kt (char a[], char c)
{
    int dem = 0;
    for(int i = 0; i < strlen(a); i++)
    {
        if(a[i] == c)
        dem++;
    }
    return dem;
}
```

Giải thích: Tạo một biến dem = 0 để đếm số lần xuất hiện của ký tự c (c sẽ nhập vào từ bàn phím). Duyệt qua từng phần tử của chuỗi a[] nếu tại vị trí i có giá trị a[i] = c thì biến dem tăng lên 1

Bài 1b. Viết chương trình nhập vào một chuỗi tiếng Anh, sau đó cho biết số lần xuất hiện từ "the" có trong chuỗi (Từ "the" không phân biệt chữ hoa-thường).

Ví dụ: Chuỗi nhập vào: The earth, the sun, the moon and THE sky

In ra: Tu "the" xuat hien 4 lan trong chuoi

```
int Dem_The(char a[]) {
    int dem = 0;

    for (int i = 0; i < strlen(a) - 2; i++) {
        if ((a[i] == 't' || a[i] == 'T') && (a[i+1] == 'h' || a[i+1] == 'H') && (a[i+2] == 'e' || a[i+2] == 'E'))
        {
            if ((a[i + 3] == ' ' || a[i + 3] == ',' || a[i + 3] == '.' || a[i + 3] == '\n'))
            {
                dem++;
            }
        }
    }

    return dem;
}
```

Giải thích:

- Tạo biến đêm dem = 0 để đếm các từ “the” có trong chuỗi
- Duyệt từ ký tự đầu tiên của chuỗi đến ký tự có vị trí độ dài chuỗi trừ đi 2 (chuỗi dài 10 thì duyệt từ 0 → 7). Bởi vì tại mỗi vị trí i, ta đã xét đến vị trí i+1 và i+2.
- Điều kiện để kiểm tra

+ Kiểm tra xem tại vị trí i, i+1 và i+2 có phải là từ “the” (không phân biệt hoa thường) hay không.
+ Nếu đúng thì sẽ kiểm tra tiếp tại vị trí i+3 là một trong các ký tự khoảng trắng, dấu chấm, dấu phẩy hoặc xuống dòng thì biến dem sẽ tăng lên 1

Vị trí i	0	1	2	3	4	5	6
a[i]	t	h	e	‘space’	s	u	N
	i	i+1	i+2	i+3			

Bài 2. Viết chương trình nhập vào một chuỗi:

a. In ra các ký tự lên màn hình, mỗi ký tự nằm trên một dòng.

```
void Nhap(char a[])
{
    printf("Nhap chuoi: ");
    fgets(a, 100, stdin);

    if(a[strlen(a) - 1] == '\n') //vi fgets ket thuc bang '\n' nen thay bang '\0'
        a[strlen(a) - 1] = '\0';
}

void Xuat(char a[])
{
    printf("Noi dung chuoi\n");
    for(int i = 0; i < strlen(a); i++)
        if(a[i] != ' ')
            printf("%c\n", a[i]);
}
```

b. Đếm số ký tự trong chuỗi.

```
int dem_KT (char a[])
{
    int dem = 0;
    for(int i = 0; i < strlen(a); i++)
    {
        if(a[i] != ' ' && a[i] != '\0')
            dem++;
    }
    return dem;
}
```

c. Đếm số ký tự không phải nguyên âm trong chuỗi.

```c
int dem_kp_nguyenam(char a[])
{
    int dem = 0;
    for(int i = 0; i < strlen(a); i++)
    {
        if(a[i] == 'U' || a[i] == 'E' || a[i] == 'O' || a[i] == 'A' || a[i] == 'I' ||
           a[i] == 'u' || a[i] == 'e' || a[i] == 'o' || a[i] == 'a' || a[i] == 'i')
           dem++;
    }
    return dem_KT(a) - dem;
}
```

Bài 3. Viết chương trình nhập vào một chuỗi. Hiển thị chuỗi theo thứ tự đảo ngược.

```c
void Nhap(char a[])
{
    printf("Nhap chuoi: ");
    fgets(a, 100, stdin);

    if(a[strlen(a) - 1] == '\n') //vi fgets ket thuc bang '\n' nen thay bang '\0'
        a[strlen(a) - 1] = '\0';
}

void Xuat(char a[])
{
    printf("Noi dung chuoi: ");
    for(int i = strlen(a) - 1; i >= 0; i--)
        if(a[i] != ' ')
            printf("%c", a[i]);
}
```

Bài 4. Viết chương trình thay thế các ký tự không phải nguyên âm trong chuỗi đã cho thành khoảng trắng.

```c
void Thay_nguyenam(char a[])
{
    for(int i = 0; i < strlen(a); i++)
    {
        if(!(a[i] == 'U' || a[i] == 'E' || a[i] == 'O' || a[i] == 'A' || a[i] == 'I' ||
           a[i] == 'u' || a[i] == 'e' || a[i] == 'o' || a[i] == 'a' || a[i] == 'i'))
           a[i] = ' ';
    }
    printf("Noi dung chuoi: %s",a);
}
```

Bài 5. Viết chương trình nhập một chuỗi từ bàn phím. Kiểm tra tính đối xứng của chuỗi vừa nhập.

Ví dụ: Chuỗi abcdcba là chuỗi đối xứng; chuỗi abcd không phải chuỗi đối xứng

Gợi ý: Sử dụng vòng lặp để duyệt chuỗi. Nếu có 1 cặp ký tự (i, n – i -1) khác nhau => chuỗi không đối xứng (n là độ dài của chuỗi).

```
int KiemTraDoiXung(char a[])
{
    for (int i = 0; i < strlen(a) / 2; i++)
    {
        if (a[i] != a[strlen(a) - i - 1])
            return 0;
    }
    return 1;
}
```

Giải thích: Duyệt từ đầu chuỗi đến giữa chuôi là đủ, ứng với mỗi i thì đối xứng của i sẽ là strlen(a) – i – 1 (VD i = 1 thì đối xứng là 7 – 1 – 1 = 5)

Vị trí i	0	1	2	3	4	5	6
a[i]	a	b	c	d	c	b	a
n=strlen(a)	i						n-i-1

Bài 6. Viết chương trình nhập vào một chuỗi, sau đó loại bỏ những khoảng trắng thừa trong chuỗi.

```
void ChuanHoa(char a[])
{
    int   j = 0;

    for (int i = 0; i < strlen(a); i++)
    {
        if (a[i] != ' ' || (i > 0 && a[i - 1] != ' '))
        {
            a[j++] = a[i];
        }
    }

    if (j > 0 && a[j - 1] == ' ' && a[j-1] == '\n') {
        j--;
    }

    a[j] = '\0';
}
```

Giải thích: Duyệt qua các kỳ tự của chuỗi, nếu tại vị trí i giá trị là một ký tự hoặc là một khoảng trắng những không liên tiếp thì sẽ được gán vào một vị trí mới (vị trí j). Nếu ký tự cuối cùng (a[j-1]) là khoảng trắng hoặc ký tự xuống dòng thì sẽ loại bỏ đi bằng cách giảm giá trị j xuống 1. Sau cùng, ta cho ký tự cuối cùng là '\0' để kết thúc chuỗi.

Vị trí i	0	1	2	3	4	5	6	7
a[i]	H	I			B	y	e	
	i = 0	i-1	i > 0					
Vị trí j	0	1	2	3	4	5	6	
a[j]	H	I		B	y	e	\0	

Bài 7. Viết chương trình nhập vào một chuỗi (hai từ chỉ cách một khoảng trắng):

a. Đếm số từ có trong chuỗi.

Cách 1:

```
int DemTu(char a[])
{
    int dem = 0;
    int flag = 0;
    for (int i = 0; i < strlen(a); i++)
    {
        if (a[i] != ' ' && a[i] != '\n' && flag == 0)
        {
            dem++;
            flag = 1;
        }
        else if (a[i] == ' ')
        {
            flag = 0;
        }
    }
    return dem;
}
```

Giải thích: Ta sẽ duyệt qua tất cả các ký tự của chuỗi, nếu tại vị trí i là một ký tự khác khoảng trắng, ta sẽ dùng một biến flag đánh dấu để thể hiện rằng ta vẫn còn đang trong phạm vi một từ.

***Không cần chuẩn hóa chuỗi**

Vị trí i	0	1	2	3	4	5	6	7
a[i]	D	a	i			h	o	c
Flag = 0	1	1	1	0	0	1	1	1
Dem = 0	1	1	1	1	1	2	2	2
	Tại đây vẫn còn đang trong 1 từ và flag = 1 nên dem không tăng			Giá trị flag = 0. Nhưng a[i] hiện tại là một khoảng trắng nên dem không tăng				

Cách 2:

```
int Dem_tu (char a[])
{
    int dem = 1;
    for (int i = 0; i < strlen(a); i++)
    {
        if(a[i] == ' ')
            dem++;
    }
    return dem;
}
```

***Muốn dùng hàm này thì phải có hàm chuẩn hóa chuỗi của bài 6. Nên sử dụng cách này để thuận tiên cho các câu sau.**

b. In ra từ đầu tiên.

```
void Tu_DT(char a[])
{
    printf("\nTu dau tien: ");
    for (int i = 0; i < strlen(a); i++)
    {
        if(a[i] != ' ')
        {
            printf("%c",a[i]);
        }
        else
            break;
    }
}
```

Giải thích: Cần phải có hàm ChuanHoa ở bài 6. Bởi vì đã chuẩn hóa nên không xuất hiện khoảng trắng thừa.Ta duyệt và in ra các ký tự cho đến khi gặp khoảng trắng đầu tiên thì ta sẽ cho dừng vòng lặp.

Vị trí i	0	1	2	3	4	5	6
a[i]	D	a	i		h	o	c
	In ra	In ra	In ra	Break			

c. In ra từ cuối cùng.

```
void Tu_CC(char a[])
{
    char b[100];
    int j = 0;
    printf("\nTu cuoi cung: ");
    for (int i = strlen(a) - 1; i >= 0; i--)
    {
        if(a[i] != ' ')
        {
            b[j++] = a[i];
        }
        else
            break;
    }
    for (int i = j - 1; i >= 0; i--)
    {
        printf("%c",b[i]);
    }
}
```

Giải thích: Ta duyệt từ cuối chuỗi, lấy các phân tử ký tự của chuỗi a gắn vào chuỗi b cho đến khi gặp khoảng trắng thì dựng lại. Vì khi ta lấy các phần tử của chuỗi a gán vào chuôi b ở vị trước ngược thì khi in chuỗi b ta sử dụng for để in ngược lại các ký tự.

Vị trí i	0	1	2	3	4	5	6
a[i]	D	a	i		h	o	c
b[j]				Dừng	2	1	0

d. In ra các từ trong chuỗi, mỗi từ nằm trên một dòng.

```
void In_Dong (char a[])
{
    printf("\nNoi dung in tren mot dong: \n");
    for(int i = 0; i < strlen(a); i++)
    {
        if(a[i] == ' ')
            printf("\n");
        else
            printf("%c",a[i]);
    }
}
```

Giải thích: Ta duyệt qua các phần tử ký tự. Nếu phần tử đó khác khoảng trắng thì sẽ in ra bình thường. Nếu phần tử ký tự đó là khoảng trắng thì ta sẽ in ra ký tự xuống dòng.

Vị trí i	0	1	2	3	4	5	6
a[i]	D	a	i		h	o	c

	In	In	In	In ‘\n’	In	In	In

Bài 8. Viết chương trình nhập một chuỗi từ bàn phím. Sau đó:

a. Chuyển các ký tự chữ cái trong chuỗi thành chữ in hoa (không dùng hàm strupr)

```
void Chu_Hoa (char a[])
{
    for (int i = 0; i < strlen(a); i++)
    {
        if(a[i] >= 'a' && a[i] <= 'z')
            a[i] -= 32;
    }
    Xuat(a);
}
```

b. Chuyển các ký tự chữ cái trong chuỗi thành chữ in thường (không dùng hàm strlwr)

```
void Chu_Thuong (char a[])
{
    for (int i = 0; i < strlen(a); i++)
    {
        if(a[i] >= 'A' && a[i] <= 'Z')
            a[i] += 32;
    }
    Xuat(a);
}
```

Giải thích: Trong bảng mã ASCII, giá trị của ‘a’ = 97 và A = ‘65’. Giữa chứ thừa và hoa chênh lệch nhau 32 nên tùy trường hợp thì ta sẽ cộng hoặc trừ đi 32.

c. Chuyển mỗi chữ cái đầu mỗi từ thành chữ in hoa, các chữ cái còn lại trong mỗi từ thành chữ in thường.

```
void Hoa_DT(char a[])
{
    int flag = 1;
    for (int i = 0; i < strlen(a); i++)
    {
        if (a[i] == ' ')
        {
            flag = 1;
        }
        else if (flag == 1 && a[i] >= 'a' && a[i] <= 'z')
        {
            a[i] -= 32;
            flag = 0;
        }
        else if (flag == 0 && a[i] >= 'A' && a[i] <= 'Z')
        {
            a[i] += 32;
        }
        else
        {
            flag = 0;
        }
    }

    Xuat(a);
}
```

Giải thích: Ta sẽ sử dụng một biến flag để đánh dấu cho biết ký tự đang xét là vị trí của chữ đầu tiên trong một từ. Khi flag = 1 và a[i] thuộc a → z thì sẽ chuyển thành ký tự hoa, đồng thời ta thay đổi flag = 0 báo hiệu rằng đã xử lý xong chữ đầu tiên của từ này. Ngược lại, sẽ chuyển thành ký tự thường

Vị trí i	0	1	2	3	4	5	6
a[i]	D	a	i		h	o	c
Flag = 1	1	0	0	1	1	0	0
	Sau khi xử lý xong chữ đầu tiên của từ “Dai”thì Flag sẽ chuyển sang 1 khi gặp khoảng trắng. Tiếp tục xử lý từ “hoc”						

Bài 9. Viết chương trình nhập vào 2 chuỗi s1 và s2. Tìm kiếm chuỗi s1 có nằm trong chuỗi s2, nếu có thì cho biết vị trí xuất hiện của chuỗi s1 trong s2.

```
void TimKiemChuoi(char a[], char b[])
{
    int len1 = strlen(a);
    int len2 = strlen(b);
    for (int i = 0; i <= len2 - len1; i++)
    {
        int j;
        for (j = 0; j < len1; j++)
        {
            if (b[i + j] != a[j])
                break;
        }

        if (j == len1)
            printf("Chuoi '%s' xuat hien tai vi tri %d trong chuoi '%s'.\n", a, i, b);
    }

}
```

Giải thích: Ta duyệt các phần tử từ i = 0 đến (strlen(a) – strlen(b)) để đảm bảo ứng với mỗi vị trí của i trong chuỗi a sẽ không bị vượt quá vị trí i chuỗi b.

Vị trí	0	1	2	3	4	5	6	7
a[j]	h	i						
b[i+j]	h	e	l	l	o		h	i

*len2 – len1 = 8 – 2 = 6

i = 0 và j = 0➔ b[0] == a[0]

i = 0 và j = 1 ➔ b[1] != a[1] ➔Dừng

i = 1 và j = 0 ➔ b[1] != a[0] ➔ Dừng

i = 2 và j = 0 ➔ b[2] != a[0] ➔ Dừng

i = 3 và j = 0 ➔ b[3] != a[0] ➔ Dừng

i = 4 và j = 0 ➔ b[4] != a[0] ➔ Dừng

i = 5 và j = 0 ➔ b[5] != a[0] ➔ Dừng

i = 6 và j = 0 ➔ b[6] == a[0]

i = 6 và j = 1 ➔ b[7] == a[1]

i = 6 và j = 2 ➔ j > len1 ➔ Dừng vòng for

Lúc này j = len 1 = 2 và vị trí i = 6

➔ Chuỗi a xuất hiện trong chuỗi b tại vị trí i = 6

Bài 10. Viết hàm tra xem trong chuỗi có kí tự số hay không, nếu có thì sao chép các ký số vào một mảng số riêng.

```
void KySo (char a[])
{
    char b[100];
    int j = 0;
    for(int i = 0; i < strlen(a); i++)
    {
        if(a[i] >= '0' && a[i] <= '9')
        {
            b[j++] = a[i];
        }
    }
    if(j != 0) Xuat(b);
    else printf("Chuoi khong co ky so");
}
```

Giải thích: Ta tạo một mảng phụ b[] chứ các ký số của chuỗi a. Duyệt tất cả các phần tử nếu là ký số thì gán vào b[j++]. Sau cùng, kiểm tra xem nếu j != 0 tức là mảng b[] có phần tử thì sẽ in ra.

Bài 11. Viết chương trình nhập một chuỗi bất kì, yêu cầu nhập 1 kí tự muốn xóa. Thực hiện xóa tất cả những kí tự đó trong chuỗi.

```
void XoaKT (char a[], int c)
{
    char b[100];
    int j = 0;
    for(int i = 0; i < strlen(a); i++)
    {
        if(a[i] != c)
        {
            b[j++] = a[i];
        }
    }
    Xuat(b);
}
```

Giải thích: Ta sử dụng một mảng phụ để chứa các ký tự khác với ký tự mà ta muốn xóa. Duyệt qua tất cả các ký tự, nêu tại vị trí i giá trị a[i] != c thì ta gán a[i] vào mảng phụ. Sau cùng, ta in ra mảng phụ là thỏa yêu cầu xóa các ký tự c nhập vào.

Bài 12. Cho chuỗi str, nhập vào vị trí vt và số kí tự cần xóa n, hãy xóa n kí tự tính từ vị trí vt trong chuỗi str.

```
void Xoa_Khoang (char a[], int vt, int n)
{
    char b[100];
    int j = 0;
    for(int i = 0; i<strlen(a); i++)
    {
        if(i < vt || i > vt+n)
        {
            b[j++] = a[i];
        }
    }
    Xuat(b);
}
```

Giải thích: Ta tạo một mảng phụ để chứa các ký tự không bị xóa. Ta duyệt qua tất cả các ký tự nếu tại vị trí i mà i < vt (vị trí bắt đầu xóa) và i > vt+n (Số lượng xóa) thì ta sẽ gán các ký tự đó vào mảng phụ.

Bài 13. Viết chương trình tìm kiếm xem ký tự nào xuất hiện nhiều nhất trong chuỗi.

```
int Max_KT(char a[])
{
    int n = strlen(a);
    int maxCount = 0;

    for (int i = 0; i < n; i++)
    {
        int count = 1;
        for (int j = i + 1; j < n; j++)
        {
            if (a[j] == a[i] && a[j] != '\0' && a[i] != ' ')
            {
                count++;
            }
        }
        if (count > maxCount)
        {
            maxCount = count;
        }
    }
    return maxCount;
}
```

Giải thích: Bởi vì trong một chuỗi rất có thể có nhiều từ có số lượng giống nhau. Nên ta xây dựng một hàm kiểm tra thử xem số lượng mà một từ xuất hiện nhiều nhất là bao nhiêu.

```
void KT_Max(char a[])
{
    int n = strlen(a);
    char checked[100] = {0};

    for (int i = 0; i < n-1; i++)
    {
        int count = 1;
        if (checked[i] == 1) continue;

        for (int j = i + 1; j < n; j++)
        {
            if (a[j] == a[i] && a[j] != '\0' && a[i] != ' ')
            {
                count++;
                checked[j] = 1;
            }
        }
        if (count == Max_KT(a))
            printf("%c ", a[i]);
    }
}
```

Giải thích: Sử dụng hàm Max_KT() để kiểm tra xem có những từ nhiều có số lần xuất hiện bằng với Max_KT() sẽ in ra hết. Sử dụng một mảng checked để đánh dấu các ký tự đã đếm.
VD: đếm ký tự ‘h’

Vị trí	0	1	2	3	4	5	6	7
a[]	h	e	l	l	o		h	i
i	i							
j = i+1							j	
checked[]	1	0	0	0	0	0	1	0

Max_KT(a) của ví dụ này là 2 (ký tự ‘l’ và ‘h’ xuất hiện 2 lần là nhiều nhất)

Bài 14. Viết chương trình nhập một chuỗi từ bàn phím (Các từ cách nhau một khoảng trắng). Tìm từ có chiều dài dài nhất và in ra.

```
void Tu_max (char a[])
{
    char b[100], c[100];
    int j = 0;
    int maxlength = 0;
    for(int i = 0; i <= strlen(a); i++)
    {
        if(a[i] != ' ' && a[i] != '\0')
            b[j++] = a[i];
        else
        {
            b[j] = '\0';
            if(strlen(b) > maxlength)
            {
                strcpy(c, b);
                maxlength = strlen(b);
            }
            j = 0;
        }
    }
    Xuat(c);
    printf("\nChieu dai: %d ",maxlength);
}
```

Giải thích: Tạo ra các mảng phụ, mảng b[] đẻ chứa các từ để kiểm ta chiều dài và so sánh với Maxlength, mảng c[] chứa từ dài nhất. Duyệt qua các phần từ, nếu như tại vị trí i là ký tự khác khoảng trắng và '\0' thì sẽ gán từng ký tự vào mảng b[] cho tới khi gặp khoảng trắng sẽ dừng lại. Sau đó, sẽ gán ký tự kết thúc mảng b[] là '\0', so sánh chiều dài của mảng b[] với maxlength nếu lớn hơn thì sẽ cập nhật maxlength và copy từ ở mảng b[] vào c[]. Cuối cùng, ta cập nhật lại biến vị trí của b[] để tiếp tực xét các từ khác trong chuỗi.

Vị trí	0	1	2	3	4	5	6	7
a[i]	h	e	l	l	o		h	i
b[j]	h	e	l	l	o	Dừng		

i = 0, a[i] != ' ' ➔ b[j = 0] = a[0]

i = 1, a[i] != ' ' ➔ b[j = 1] = a[0]

….

i = 5, a[i] == ' ' ➔ b[j=5] = '\0'

⇨ So sánh với maxlength và cập nhật vào mảng c[]
⇨ Cập nhật lại vị trí j = 0

i = 6, a[i] != ' ' ➔ b[j = 0] = a[6]

…

(Tiếp tục cho đến hết chuỗi.)

Bài 15. Viết chương trình nhập một chuỗi từ bàn phím (Các từ cách nhau một khoảng trắng). Tìm từ có chiều dài ngắn nhất và in ra.

```
void Tu_min (char a[])
{
    char b[100], c[100];
    int j = 0;
    int minlength = 100;
    for(int i = 0; i <= strlen(a); i++)
    {
        if(a[i] != ' ' && a[i] != '\0')
            b[j++] = a[i];
        else
        {
            b[j] = '\0';
            if(strlen(b) < minlength)
            {
                strcpy(c, b);
                minlength = strlen(b);
            }
            j = 0;
        }
    }
    Xuat(c);
    printf("\nChieu dai: %d ",minlength);
}
```

Giải thích: Tương tự bài 14, nhưng minlength sẽ được cho bằng 100 vì mảng chuỗi có kích thước dài nhất là 100 (a[100])