اشاره گرها(Pointers)

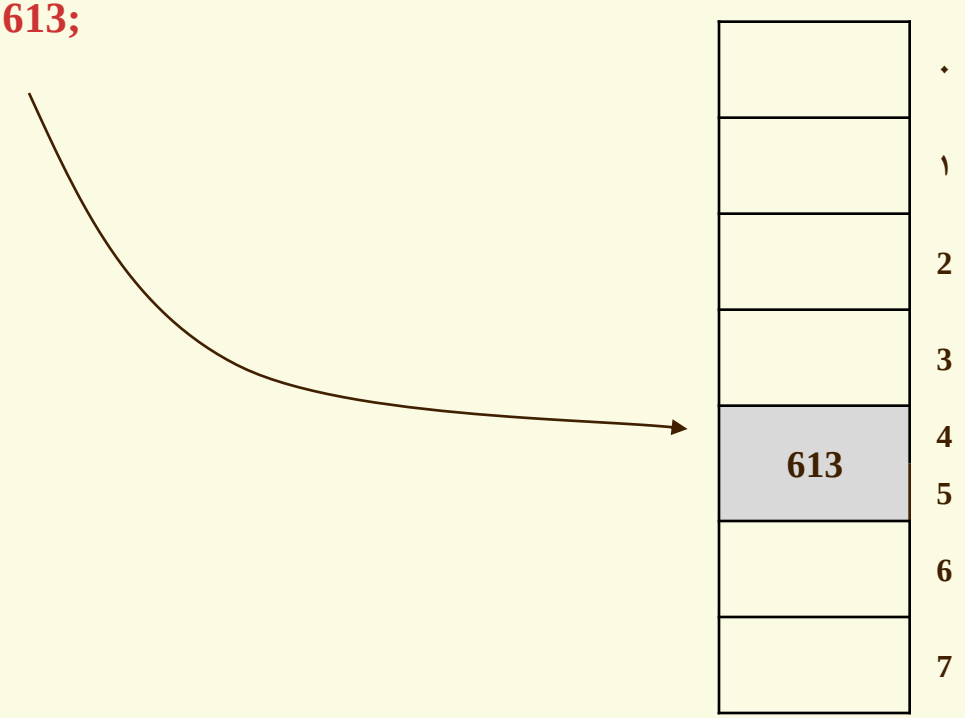
داده‌هائی که در کامپیوتر در حافظه اصلی ذخیره میشوند بایت‌های متوالی از حافظه بسته به نوع data اشغال می‌کنند.

نوع داده	مقادیر	حافظه لازم
int	32767 تا 32768–	۲ بایت
long int	2147483647 تا 2147483648–	۴ بایت
char	یک کارکتر	۱بایت
float	3.4e38 تا 1.2e-38	۴ بایت
double	1.8e308 تا 2.2e-308	۸ بایت

اشاره گرها (Pointers)

با داشتن آدرس داده در حافظهٔ اصلی می‌توان براحتی به آن داده دسترسی پیدا نمود و از طرف دیگر آدرس هر داده در حافظه آدرس بایت شروع آن داده می‌باشد.

```
int x = 613;
```



نکته:

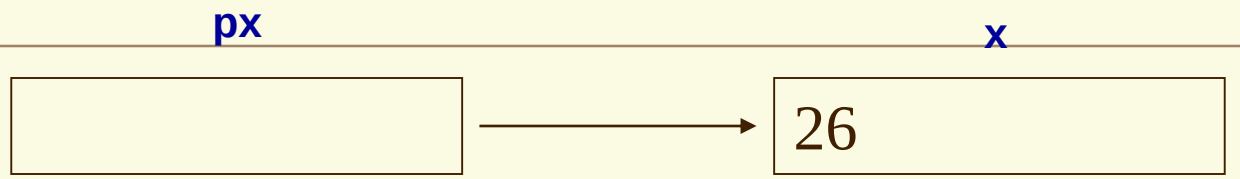
در کامپیوتر آدرس‌ها معمولاً دو بایت اشغال می‌نمایند. اگر آدرس x را در px قرار دهیم آنگاه می‌گوئیم که px به x اشاره می‌نماید.

آدرس متغیر x را بوسیله &x نشان میدهیم و عملگر & را عملگر آدرس می‌نامند.

```
int  x , *px
px = &x ;
```

مثال:

```
int y , x , *px ;
x = 26 ;
px = &x ;
```



حال اگر دستور العمل ; 10 =+ x را بدهیم :

حال اگر دستورالعمل ; 7 + *px = *px بدهیم.

آرایه یک بعدی و اشاره گرها

0	26.5	x
1	24.7	
2	5.8	
3	-73.2	
4	69.0	
5	100.5	
6	-13.24	
7	424.3	
8	187.8	
9	358.2	

اولین عنصر آرایه بوسیله x[0] مشخص می‌شود.

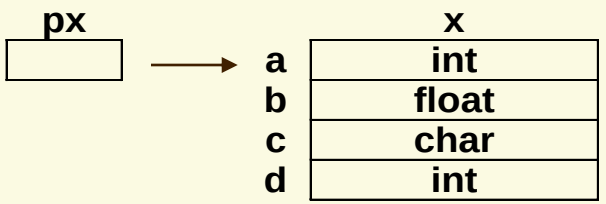
آدرس اولین عنصر آرایه بوسیله &x[0] یا بوسیله x مشخص می‌شود.

آدرس i امین عنصر آرایه بوسیله &x[i-1] یا بوسیله x(i-1) مشخص می‌شود.

ساختارها و اشاره گرها

می‌توان اشاره‌گری را تعریف نمود که به اولین بایت یک ساختار (struct) اشاره نماید.

```
struct    struc1
{
        int a ;
        float b ;
        char  c;
        int d ;
} x, *px ;

px = &x ;
```



عبارت x.a معادل px–>a معادل (*px).a می‌باشد.

آرایه‌های دوبعدی و اشاره‌گرها

یک آرایه دوبعدی بصورت تعدادی آرایه یک بعدی می‌توان تعریف نمود.
اگر x یک ماتریس 5 سطری و 4 ستونی از نوع اعشاری باشد قبلاً این ماتریس را با

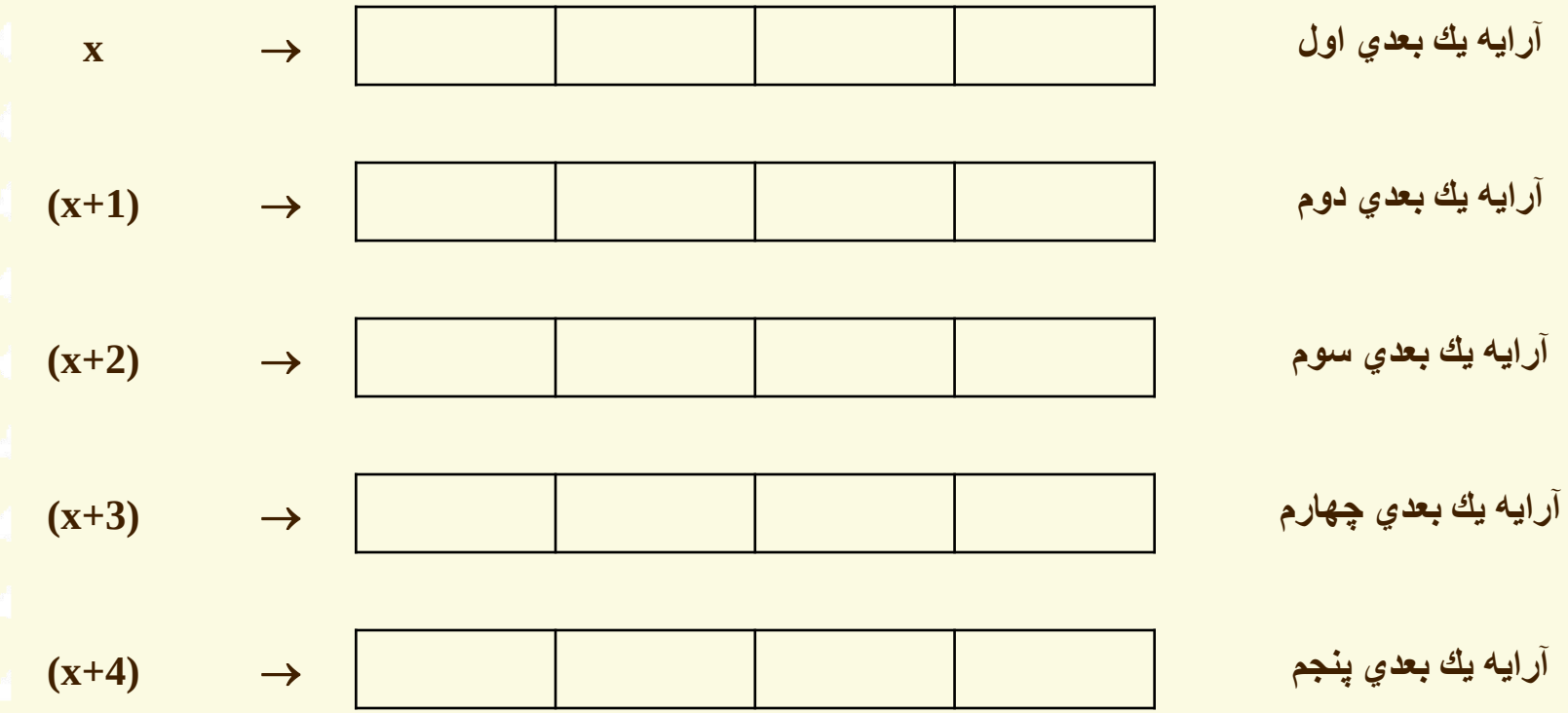
```
float x[5][4];
```

معرفی کردیم. حال با استفاده از اشاره‌گرها بصورت زیر معرفی نمائیم:

```
float  (*x)[4];
```

آرایه‌های دوبعدی و اشاره گرها

```
float  (*x)[4];
```



در برنامه زیر یک آرایه 5 عنصری از نوع int ایجاد شده و مقادیر عناصر آرایه را به چهار طریق نمایش می‌دهد.

```
#include      <iostream.h>
#include      <conio.h>
int main( )
{
int x[ ]={12, 25, 6, 19, 100};
clrscr( );
int *px=x;
//  نام آرایه بدون اندیس، اشاره به عنصر اول آرایه مي‌نماید
for(int i=0; i<=4; i++)
cout << *(x+i) << endl;
//the second method
for(i=0; i<5; i++)
cout << x[ i ] << ′ \n′;
//the third method
for(i=0; i<=4; i++)
cout << px[ i ]<<endl;
//the forth method
for(i=0; i<=4; i++)
cout << *(px+i)<<endl;
return 0; }
```

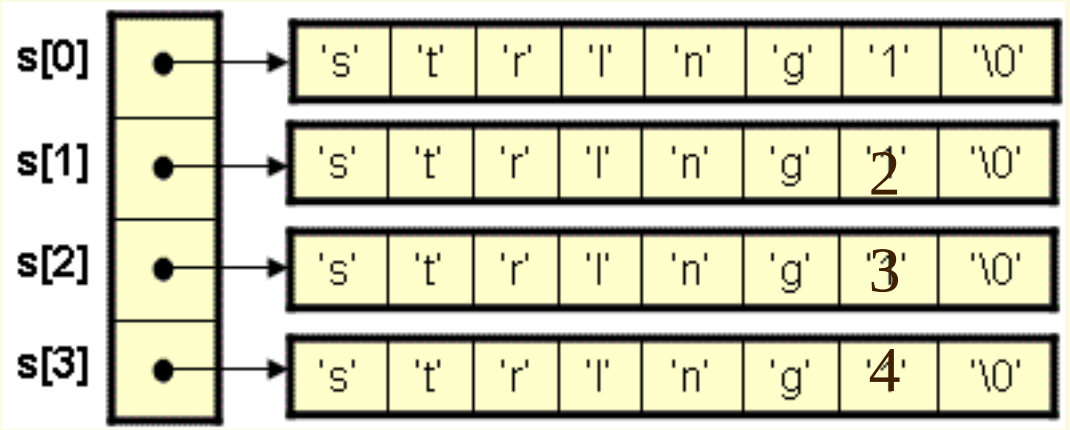
ارتباط اشاره گرها و آرایه ها

```
#include <iostream.h>
void main()
{
int b[] = { 10, 20, 30, 40 };
int *bPtr = b;
int i;
cout << "b[i]:\n";
for ( i = 0; i < 4; i++ )
cout << "b[" << i << "] = " << b[ i ] << '\n';
cout << "\n*(b + i):\n";
for ( i = 0; i < 4; i++ )
cout << "*(b + " << i << ") = " << *( b + i ) << '\n';
cout << "\nbPtr[i]:\n";
for ( i = 0; i < 4; i++ )
cout << "bPtr[" << i << "] = " << bPtr[ i ] <<'\n';
cout << "\n*(bPtr + i):\n";
for ( i = 0; i < 4; i++ )
cout << "*(bPtr + " << i << ") = " << *( bPtr + i ) << '\n';
}
```

```
b[i]:
b[0] = 10
b[1] = 20
b[2] = 30
b[3] = 40

*(b + i):
*(b + 0) = 10
*(b + 1) = 20
*(b + 2) = 30
*(b + 3) = 40

bPtr[i]:
bPtr[0] = 10
bPtr[1] = 20
bPtr[2] = 30
bPtr[3] = 40

*(bPtr + i):
*(bPtr + 0) = 10
*(bPtr + 1) = 20
*(bPtr + 2) = 30
*(bPtr + 3) = 40
```

آرایه ای از اشاره گرها

```
const char *s[4] =
{ "string1","string2","string3","string4" }
```



برنامه زیر دو مقدار اعشاری را گرفته مقادیر آنها را بکمک تابع swap جابه‌جا می‌نماید.

```
#include      <iostream.h>
#include     <conio.h>
void swap(float *, float *);
int main( )
{
float a,b;
cin >> a >> b;
cout << a << endl << b << endl ;
return 0;
}
void swap(float *px , float *py)
{
   float t;
   t = *px;
   *px = *py;
   *py = t ;
    return;
 }
```

رشته‌ها و توابع مربوطه

رشته‌ها در C++ ، آرایه‌ای از کرکترها می‌باشند که با کرکتر '\0' ختم میشوند.

```
char name[ ]= " sara";
```

s
a
r
a
\0

رشته و اشاره گر

هر رشته از طریق اشاره‌گری به اولین کارکتر آن در دسترس قرار می‌گیرد. آدرس یک رشته، آدرس کارکتر اول آن می‌باشد. به رشته‌ها می‌توان مقدار اولیه تخصیص داد.

```
char    *name = " sara";
```

برنامه ذیل پنج اسم را بصورت 5 رشته در نظر گرفته آنها را بترتیب حروف الفباء مرتب نموده نمایش می‌دهد.

```
#include      <iostream.h>
#include      <string.h>
void sort(char *[ ]);
int main( )
{
char *name[5] = {"sara", "afsaneh", "babak", "saman", "naser" };
sort(name);// display sorted strings
for(int i=0; i<5; ++i)
       cout << name[ i ] << endl;
return 0; }
sort(char *name[ ])
{
char  *t;
for(int i=0; i<4; ++i)
for(int j=i+1; j<5; ++j)
if(strcmpi(name[ i ], name[ j ]> 0) )
{// interchange  the two strings
t= name[ i ];
name[ j ] = name[ i ];
name[ i ] = t ;}
return ; }
```

تابع strcmpi(s1, s2)

رشته‌های s1 و s2 را با هم مقایسه نموده (بدون توجه به حروف کوچک و بزرگ) اگر رشته s1 برابر با رشته s2 باشد مقدار صفر و اگر رشته s1 کوچکتر از رشته s2 باشد یک مقدار منفی در غیر اینصورت یک مقدار مثبت بر می‌گرداند.

ورودی/خروجی رشته‌های کاراکتری:

در C++ به چند روش می‌توان رشته‌های کاراکتری را دریافت کرده یا نمایش داد.

یک راه استفاده از عملگرهای کلاس string است که در بخش‌های بعدی به آن خواهیم پرداخت.

روش دیگر، استفاده از توابع کمکی است که آن را در ادامه شرح می‌دهیم.

تابع cin.get()

این برنامه تعداد حرف 'e' **در جریان ورودی را شمارش می‌کند. تا وقتی** cin.get(ch) **کاراکترها را با موفقیت به درون** ch **می‌خواند، حلقه ادامه می‌یابد:**

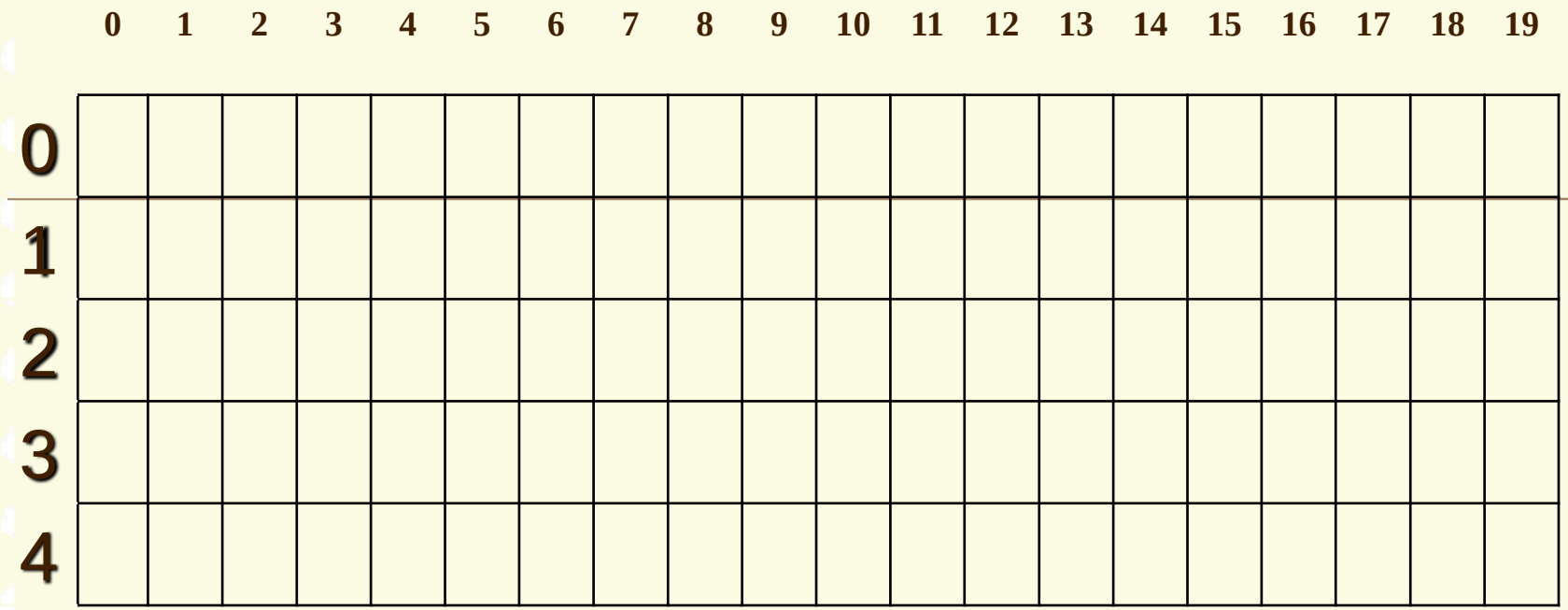
```
int main()
{  char ch;
   int count = 0;
   while (cin.get(ch))
      if (ch = = 'e') ++count;
   cout << count << " e's were counted.\n";
}
```

آرایه‌ای از رشته‌ها

به خاطر دارید که گفتیم یک آرایهٔ دوبعدی در حقیقت آرایه‌ای یک بعدی است که هر کدام از اعضای آن یک آرایهٔ یک بعدی دیگر است. مثلا در آرایهٔ دو بعدی که به شکل مقابل اعلان شده باشد:

```
char name[5][20];
```

این آرایه در اصل پنج عضو دارد که هر عضو می‌تواند بیست کاراکتر داشته باشد. اگر آرایهٔ فوق را با تعریف رشته‌های کاراکتری مقایسه کنیم، نتیجه این می‌شود که آرایهٔ بالا یک آرایهٔ پنج عنصری است که هر عنصر آن یک رشتهٔ کاراکتری بیست حرفی است. این آرایه را می‌توانیم به شکل مقابل تصور کنیم.



از طریق name[0] و name[1] و name[2] و name[3] و name[4] می‌توانیم به هر یک از رشته‌های کاراکتری در آرایهٔ بالا دسترسی داشته باشیم. یعنی آرایهٔ name گرچه به صورت یک آرایهٔ دوبعدی اعلان شده لیکن به صورت یک آرایهٔ یک بعدی با آن رفتار می‌شود.

توابع استاندارد رشته‌های کاراکتری:

سرفایل<cstring> که به آن «کتابخانهٔ رشته‌های کاراکتری» هم می‌گویند، شامل خانوادهٔ توابعی است که برای دست‌کاری رشته‌های کاراکتری خیلی مفیدند.

مثال بعدی ساده‌ترین آن‌ها یعنی تابع طول رشته را نشان می‌دهد. این تابع، طول یک رشتهٔ کاراکتری ارسال شده به آن (یعنی تعداد کاراکترهای آن رشته) را برمی‌گرداند.

تابع strlen(s)

رشتهٔ S را بعنوان آرگومان گرفته طول رشته را مشخص می‌نماید.

تابع strlen():

برنامهٔ زیر یک برنامهٔ آزمون ساده برای تابع strlen() است.

وقتی strlen(s) فراخوانی می‌شود، تعداد کاراکترهای درون رشتهٔ s که قبل از کاراکتر NUL قرار گرفته‌اند، بازگشت داده می‌شود:

```
#include <cstring>
int main()
{  char s[] = "ABCDEFG";
   cout << "strlen(" << s << ") = " << strlen(s) << endl;
   cout << "strlen(\"\") = " << strlen("") << endl;
   char buffer[80];
   cout << "Enter string: ";
   cin >> buffer;
   cout << "strlen(" << buffer << ") = " << strlen(buffer)  << endl;
}
```

تابع strcpy(s1,s2)

دو رشتهٔ s1 و s2 را بعنوان آرگومان گرفته رشتهٔ s2 را در رشتهٔ s1 کپی می‌نماید و نهایتاً مقدار رشتهٔ s1 را بر می‌گرداند.

تابع strcpy():

برنامهٔ زیر نشان می‌دهد که فراخوانی strcpy(s1, s2) چه تاثیری دارد:

```
Before strcpy(s1,s2):
s1 = [ABCDEFG], length = 7
s2 = [XYZ], length = 3
After strcpy(s1,s2):
s1 = [XYZ], length = 3
s2 = [XYZ], length = 3
```

```
#include <iostream>
#include <cstring>
int main()
{  char s1[] = "ABCDEFG";
   char s2[] = "XYZ";
   cout << "Before strcpy(s1,s2):\n";
   cout << "\ts1 = [" << s1 << "], length = " << strlen(s1) << endl;
   cout << "\ts2 = [" << s2 << "], length = " << strlen(s2) << endl;
   strcpy(s1,s2);
  cout << "After strcpy(s1,s2):\n";
   cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
   cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
}
```

تابع strncpy(s1, s2,n)

دو رشتهٔ s2 , s1 و مقدار صحیح و مثبت n را بعنوان آرگومان گرفته، حداکثر n کرکتر را از رشتهٔ s2 در رشتهٔ s1 کپی نموده، نهایتاً مقدار رشتهٔ s1 را برمیگرداند.

تابع strncpy():

برنامهٔ زیر بررسی می‌کند که فراخوانی strncpy(s1, s2, n) چه اثری دارد:

```
Before strncpy(s1,s2,2):
s1 = [ABCDEFG], length = 7
s2 = [XYZ], length = 3
After strncpy(s1,s2,2):
s1 = [XYCDEFG], length = 7
s2 = [XYZ], length = 3
```

```
int main()
{  char s1[] = "ABCDEFG";
   char s2[] = "XYZ";
   cout << "Before strncpy(s1,s2,2):\n";
   cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
   cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
   strncpy(s1,s2,2);
   cout << "After strncpy(s1,s2,2):\n";
   cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
   cout << "\ts2 = [" << s2 << "], length = " << strlen(s2) << endl;
}
```

تابع strcat(s1, s2)

دو رشتهٔ s1 و s2 را بعنوان آرگومان گرفته رشتهٔ s2 را به انتهای رشتهٔ s1 اضافه می‌نماید. کرکتر اول رشتهٔ s2 روی کرکتر پایانی '\0' رشتهٔ s1 نوشته می‌شود ونهایتاً رشتهٔ s1 را برمیگرداند.

تابع الصاق رشته strcat():

برنامهٔ زیر بررسی می‌کند که فراخوانی strcat(s1, s2) چه تاثیری دارد:

```
int main()
{  char s1[] = "ABCDEFG";
   char s2[] = "XYZ";
   cout << "Before strcat(s1,s2):\n";
   cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
   cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
   strcat(s1,s2);
   cout << "After strcat(s1,s2):\n";
   cout << "\ts1 = [" << s1 << "], length = " << strlen(s1)<< endl;
   cout << "\ts2 = [" << s2 << "], length = " << strlen(s2)<< endl;
}
```

```
Before strcat(s1,s2):
s1 = [ABCDEFG], length = 7
s2 = [XYZ], length = 3
After strcat(s1,s2):
s1 = [ABCDEFGXYZ], length = 10
s2 = [XYZ], length = 3
```

تابع strncat(s1, s2,n)

دو رشته s1 و s2 ومقدار صحیح و مثبت n را بعنوان آرگومان گرفته، حداکثر n کرکتر از رشتهٔ s2 را در انتهای رشتهٔ s1 کپی می‌نماید. اولین کرکتر رشته s2 روی کرکتر پایانی '\0' رشتهٔ **s1** می‌نویسد ونهایتاً مقدار رشتهٔ **s1** را برمیگرداند.

تابع strcmp(s1, s2)

رشته‌های s1 و s2 را با هم مقایسه نموده اگر s1 برابر با s2 باشد مقدار صفر و اگر رشته s1 کوچکتر از رشته s2 باشد یک مقدار منفی در غیر اینصورت یک مقدار مثبت برمی‌گرداند.

تابع strncmp(s1, s2,n)

حداکثر n کرکتر از رشتهٔ s1 را با n کرکتر از رشتهٔ s2 مقایسه نموده در صورتیکه s1 کوچکتر از s2 باشد یک مقدار منفی، اگر s1 مساوی با s2 باشد مقدار صفر در غیر اینصورت یک مقدار مثبت برمیگرداند.

مثال:

```
#include      <iostream.h>
#include      <string.h>
#include      <conio.h>
int main( )
{
char *s1= ″happy birthday″;
char *s2= ″happy holidays ″;
clrscr( );
cout << strcmp(s1, s2) << endl;
cout << strncmp(s1, s2, 7) << endl ;
return 0;
 }
```