درس «مبانی کامپیوتر و برنامه‌سازی»

رشته‌ها

صادق علی‌اکبری

سرفصل مطالب

- آشنایی با رشته
- ورودی و خروجی رشته
- توابع کار با رشته‌ها

تا به حال در بیشتر برنامه‌ها از اعداد استفاده می کردیم
اگر بخواهیم از کلمات استفاده کنیم، باید سراغ رشته‌ها برویم!

رشته (String)

```
char c;
c = 'A';
c = '#';
c = '\n';
```

- یادآوری: نوع داده کاراکتر: یک کاراکتر را نشان می‌دهد، مثال:
  - یک کاراکتر، با کوتیشن شروع می‌شود و با کوتیشن پایان می‌پذیرد
- **رشته**: آرایه‌ای از کاراکترها (رشته‌ای از کاراکترها)
  - مثلاً Ali یک رشته است که دارای سه کاراکتر A و l و i است
- در آرایه، انتهای رشته با کمک عدد صفر (کاراکتر NUL یا '\0') مشخص می‌شود
- مثال: `char name[] = {'A', 'l', 'i', '\0'};`
- همچنین می‌توانیم کل رشته را با کمک " (double quotation) نشان دهیم
- مثال: `char name[] = "Ali";`

A	l	i	\0

چند نکته درباره رشته‌ها

- ممکن است آرایه بزرگی داشته باشیم و بخش کوچکتری از آرایه شامل رشته موردنظر باشند
  - انتهای رشته، با کاراکتر صفر (\0) مشخص می‌شود، نه با انتهای آرایه

```
char* pointer = "Naghi";
char array[] = "Naghi";
```

- اشاره‌گر به کاراکتر، مانند آرایه‌ای از کاراکترها قابل استفاده است
- نوع داده «آرایه‌ای از کاراکترها» و یا «اشاره‌گری به کاراکتر» نوع داده خاصی است
  - که به عنوان رشته (String) در نظر گرفته می‌شود
  - با انواع دیگر آرایه و اشاره‌گر متمایز است
- از cin و cout می‌توانیم برای خواندن و نوشتن رشته استفاده کنیم
  - رفتار cin و cout برای آرایه‌ای از کاراکترها یا اشاره‌گری به کاراکتر متفاوت است
  - مثلاً برای آرایه‌ای از int ، cin تعریف نشده و cout اعضای آرایه را چاپ نمی‌کند

مثال

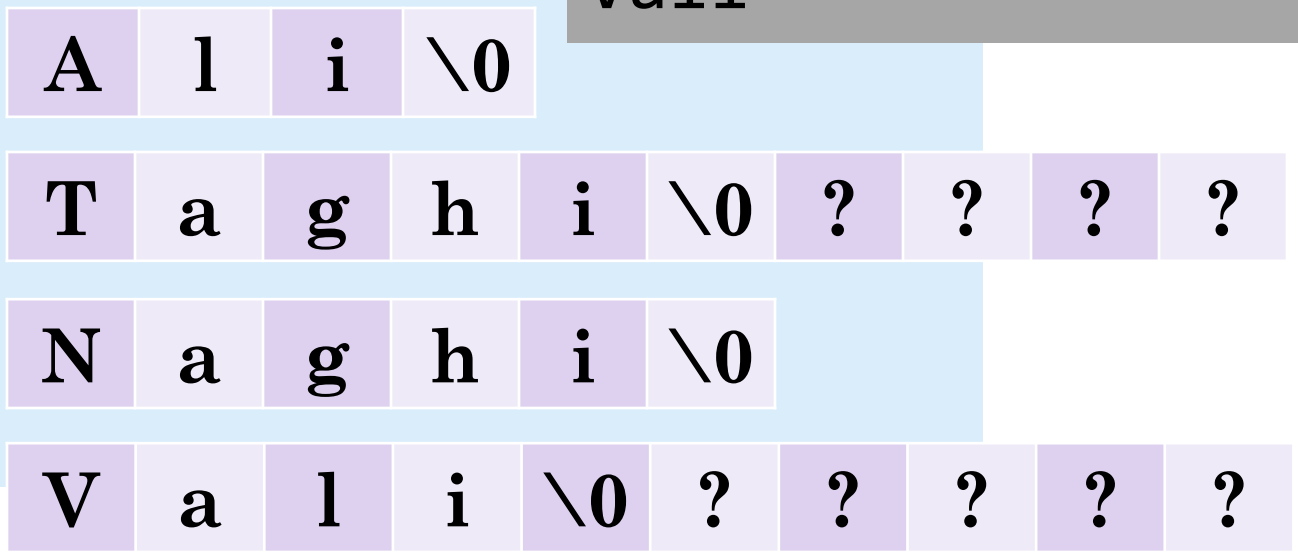
```
char name1[] = {'A', 'l', 'i', '\0'};
char name2[10] = {'T', 'a', 'g', 'h', 'i', '\0'};
char* pointer = "Naghi";

cout<<name1<<endl;
cout<<name2<<endl;
cout<<pointer<<endl;

char name[10];
cout<<"Enter a name:";
cin>>name;
cout<<name<<endl;

pointer = name;
cout<<pointer<<endl;
```

```
Ali
Taghi
Naghi
Enter a name:Vali
Vali
Vali
```



مثال

```
char s1[100];
s1[0]='A';
s1[1]='l';
s1[2]='i';
s1[3]='\0';

char s2[] = "Salam";

char* s3 = "Hello";

char* s4 = new char[10];
s4[0]='B';
s4[1]='y';
s4[2]='e';
s4[3]='\0';

char* s5 = s1;
```

```
cout<<s1<<endl;
cout<<sizeof s1<<endl;

cout<<s2<<endl;
cout<<sizeof s2<<endl;

cout<<s3<<endl;
cout<<sizeof s3<<endl;

cout<<s4<<endl;
cout<<sizeof s4<<endl;

cout<<s5<<endl;
cout<<sizeof s5<<endl;
```

```
Ali
100
Salam
6
Hello
4
Bye
4
Ali
4
```

مثال

```
const int Array_Size = 100;
char s[Array_Size];
cin >> s;
int i = 0;
while (s[i] != '\0')
{
   char c = s[i];
   cout << c;
   i++;
}
cout << endl;
cout << s << endl;
cout << Array_Size << endl;
cout << i << endl;
```

```
Salam
Salam
Salam
100
5
```

- دقت کنید: طول آرایه با طول رشته متفاوت است
- طول واقعی رشته از طول آرایه کمتر است

سرریز حافظه رشته

- سرریز حافظه رشته (همانند سرریز هر آرایه دیگری) ممکن و خطرناک است
  - ممکن است به خطا در زمان اجرا (runtime error) منجر شود
- مثال:

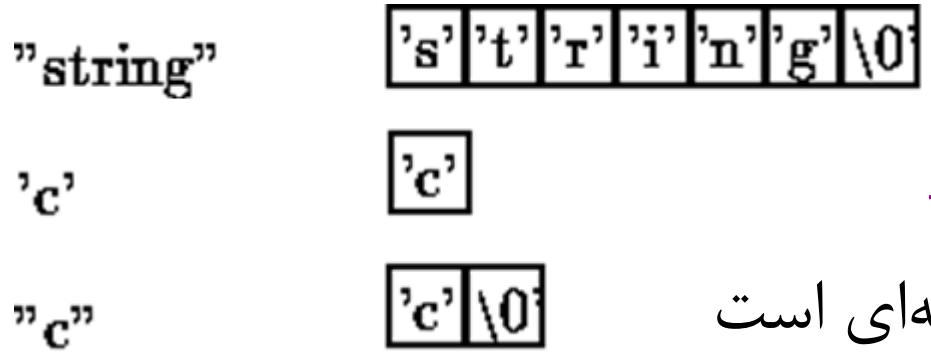
```
cout<<"Enter a small string:"<<endl;
char s[10];
cin >> s;
```

```
Enter a small string:
qwyeqwyetyuquywteuquqqwteuyqteyutquwyet
Segmentation Fault
```

- نکته: برای این مشکل، از cin.get یا از نوع داده string استفاده می‌شود
  - نوع داده string در کتابخانه استاندارد C++ وجود دارد و با char* یا آرایه متفاوت است
  - مطالعه در این زمینه به عهده شما (خارج از محدوده درس است)

```
std::string s;
cin >> s;
cout<<s<<endl;
cout<<s[0]<<endl;
cout<<s.size();
```

لیترال رشته (String Literals)



- عبارتی مانند "Salam" یک لیترال (یا ثابت) رشته‌ای است
  - مثال:

```
char* s = "Salam" ;
cout  << "Hello" ;
```

- در زمان اجرا، چنین عبارتی به صورت یک آرایه در حافظه جای می‌گیرد
- حافظه مربوطه همانند متغیرهای سراسری (global) و متغیرهای استاتیک، ماندگار است
  - در پشته نیست که بعد از مدتی آزاد شود
  - در Heap هم نیست و این حافظه نیازی به آزادسازی (delete) ندارد
  - یعنی این حافظه از ابتدا تا انتهای اجرای برنامه اشغال می‌شود

نکته درباره لیترال رشته‌ای

- در عبارتی مثل char str[ ]="Salam" آرایه str یک کپی از کاراکترهای لیترال Salam دارد
- اما در char* ptr="Salam" ، اشاره‌گر ptr دقیقاً به محل لیترال Salam اشاره می‌کند
- محل قرارگیری لیترال‌های رشته‌ای در حافظه قابل‌پیش‌بینی نیست
  - حتی ممکن است کامپایلر به واسطه بهینه‌سازی آن‌ها را کنار هم جای دهد
  - مثلاً شاید مقدار عبارت `bool b="bar"==3+"foobar";` برابر با true یا false شود
- تغییر مقدار در محتوای یک لیترال، اشتباه است
  - رفتار C++ در این زمینه «تعریف‌نشده» (undefined) است، شاید به خطا در زمان اجرا منجر شود
  - مثلاً ممکن است کامپایلر این رشته را در بخشی read-only جای دهد
- پس بهتر است برای اشاره به یک لیترال رشته‌ای، به جای char* از const char* استفاده شود

```
const char* s1 = "football";
char* s2 = "Hello"; Warning: deprecated conversion from string constant to 'char*'
```

رفتار cin برای رشته‌ها

- cin کاراکترهایی مانند فاصله و تب را به عنوان پایان رشته در نظر می‌گیرد

- مثال:

```
char s[100];
cin>>s;
cout<<s;
```

```
Salam Ali Khoobi?
Salam
```

- پس اگر بخواهیم رشته‌ای شامل فاصله یا تب بخوانیم چه؟
  - مثلاً یک رشته برابر با Salam Ali باشد
  - راه‌حل: cin.getline

```
char s1[100], s2[100], s3[100];
cin>>s1;
cin>>s2;
cin>>s3;
cout<<s1<<endl;
cout<<s2<<endl;
cout<<s3<<endl;
```

```
Salam. Ali! Khhobi?!
Salam.
Ali!
Khhobi?!
```

cin.getline

- تابع cin.getline یک رشته با حداکثر طول مشخص شده را از کاربر می‌گیرد
- فراخوانی cin.getline(str, MAX); یعنی:
  رشته‌ای با حداکثر طول MAX را از کاربر بخوان و در str ذخیره کن
- مثال:

```
const int MAX = 100;
char s[MAX];
cin.getline(s, MAX);
cout<<s<<endl;
cin.getline(s, 10);
cout<<s<<endl;
```

```
Salam Ali! Khoobi?
Salam Ali! Khoobi?
Salam Ali! Khoobi?
Salam Ali
```

- نکته: در واقع getline تابعی درون cin است
  - cin یک شیء است که توابعی روی آن قابل فراخوانی است
  - توضیحات بیشتر مربوط به برنامه‌نویسی شیءگرا خواهد بود

تمرین

- تابعی بنویسید که یک رشته به عنوان پارامتر بگیرد و طول رشته را برگرداند
- تابعی بنویسید که یک رشته به عنوان پارامتر بگیرد و تعداد کاراکترهای صدادار آن را برگرداند (AEOUI)
- تابعی بنویسید که یک جمله به عنوان پارامتر بگیرد و تعداد کلمات آن را برگرداند
  - فرض کنید کلمات فقط با فاصله از هم جدا می‌شوند
- تابعی بنویسید که یک رشته به عنوان پارامتر بگیرد و یک کپی از آن را برگرداند

- تابعی بنویسید که دو رشته به عنوان پارامتر بگیرد و مشخص کند این دو با هم برابرند یا خیر
  - true یا false برگرداند
- تابعی بنویسید که دو رشته به عنوان پارامتر بگیرد و مشخص کند اولی شامل دومی هست یا نه
  - true یا false برگرداند
  - مثلاً contains("Ali Alavi", "Ali“) مقدار true برمی‌گرداند

مثال

- تابعی که یک رشته به عنوان پارامتر بگیرد و طول رشته را برگرداند

```
int length(char* s) {
    int index = 0;
    while (s[index] != '\0')
        index++;
    return index;
}
```

```
int main() {
    cout << length("Ali") << endl;
    char s1[100] = "Taghi";
    cout << length(s1) << endl;
    char s2[100] = { 'N', 'a', 'g', 'h', 'i', '\0' };
    cout << length(s2) << endl;
    char* ptr = s1;
    cout << length(ptr) << endl;
    ptr = new char[100];
    ptr[0]='\0';
    ptr[1]='A';
    cout << length(ptr) << endl;
}
```

```
3
5
5
5
0
```

مثال

- تابعی که یک رشته به عنوان پارامتر بگیرد و تعداد کاراکترهای صدادار آن را برگرداند (کاراکترهای صدادار: AEIOU)

```
2
2
```

```
bool is_vowel(char c){
   switch(c){
   case 'A':case 'a':
   case 'E':case 'e':
   case 'I':case 'i':
   case 'O':case 'o':
   case 'U':case 'u':
   return true;
   }
   return false;
}
```

```
int vowels(char* s) {
   int i = 0, num=0;
   while (s[i] != '\0')
      if(is_vowel(s[i++]))
            num++;
   return num;
}
```

```
int main() {
   cout << vowels("Ali") << endl;
   char s1[100] = "Taghi";
   cout << vowels(s1) << endl;
}
```

مثال

- تابعی که یک جمله به عنوان پارامتر بگیرد و تعداد کلمات آن را برگرداند (فرض: کلمات فقط با فاصله از هم جدا می‌شوند)

```
int count(char* s, char c) {
   int i = 0, num = 0;
   while (s[i] != '\0')
      if (s[i++] == c)
            num++;
   return num;
}
```

```
int words(char* s) {
   if(*s=='\0')
      return 0;
   return count(s, ' ')+1;
}
```

```
int main() {
   cout << words("") << endl;
   cout << words("Ali") << endl;
   cout << words("Ali Alavi is a student.") << endl;
}
```

```
0
1
5
```

مثال

- تابع count را بدون کمک متغیر i بازنویسی کنید

```
int count(char* s, char c) {
   int i = 0, num = 0;
   while (s[i] != '\0')
      if (s[i++] == c)
            num++;
   return num;
}
```

- هر دو صحیح است:

```
int count(char* s, char c) {
   int num = 0;
   while (*s != '\0'){
      if (*s == c)
      num++;
      s++;
   }
   return num;
}
```

```
int count(char* s, char c) {
   int num = 0;
   for(;*s!='\0';s++)
      if (*s == c)
            num++;
   return num;
}
```

مثال

- تابعی که یک رشته به عنوان پارامتر بگیرد

```
int length(char* s) {
    int index = 0;
    while (s[index] != '\0')
        index++;
    return index;
}
```

```
char* copy(char* s) {
    int len = length(s);
    char* news = new char[len + 1];
    for (int i = 0; i < len+1; ++i)
        news[i]=s[i];
    return news;
}
```

```
int main() {
    char*s = copy("Ali");
    char*t = copy(s);
    printf("%s\n", "Ali");
    printf("%s\n", s);
    printf("%s\n", t);
    printf("%d\n", "Ali");
    printf("%d\n", s);
    printf("%d\n", t);
    delete[]s;
    delete[]t;
}
```

```
Ali
Ali
Ali
4239792
3478832
3478848
```

مثال

```
int length(char* s) {
    int index = 0;
    while (s[index] != '\0')
        index++;
    return index;
}
```

- تابعی که دو رشته به عنوان پارامتر بگیرد

و مشخص کند این دو با هم برابرند یا خیر

```
bool compare(char* s, char* t) {
   int len1 = length(s);
   int len2 = length(t);
   if (len1 != len2)
      return false;
   for (int i = 0; i < len1; ++i)
      if (s[i] != t[i])
            return false;
   return true;
}
```

```
int main() {
  cout.setf(std::boolalpha);

  cout<<compare("Ali", "Ali");
  cout<<compare("Ali", "Taghi");
  cout<<compare("Ali", "ali");
}
```

```
true
false
false
```

مثال

- تابعی که بخشی از یک رشته را برگرداند (اندیس شروع و طول موردنظر را به عنوان پارامتر بگیرد)

```
char* substring(char* s, int begin, int size) {
   char* news = new char[size+1];
   for (int i=0, j = begin; i < size; ++i, j++)
      news[i] = s[j];
   news[size] = '\0';
   return news;
}
```

```
int main() {
   cout<<substring("AliAlavi", 3, 5)<<endl;
   cout<<substring("AliAlavi", 0, 3)<<endl;
}
```

```
Alavi
Ali
```

مثال

```
int length(char* s) {...}
bool compare(char* s, char* t) {...}
```

- تابعی که دو رشته به عنوان پارامتر بگیرد

و مشخص کند اولی شامل دومی هست یا نه

```
bool contains(char* main, char* sub) {
   int len_sub = length(sub);
   while(*main!='\0'){
      if(compare(sub, substring(main, 0, len_sub)))
            return true;
      main++;
   }
   return false;
}
```

```
int main() {
  cout.setf(std::boolalpha);
  cout<<contains("AliAlavi", "Ali");
  cout<<contains("AliAlavi", "Alavi");
  cout<<contains("AliAlavi", "i");
  cout<<contains("AliAlavi", "Taghi");
}
```

```
true
true
true
false
```

توابع استاندارد cstring

- مثال‌هایی که دیدیم، توابعی مفید بودند که در بسیاری از برنامه‌ها کاربرد دارند
- کتابخانه cstring بسیاری از این توابع مفید را به خوبی پیاده‌سازی کرده است

تابع	کاربرد
strcpy(s1, s2)	رشته s2 را در رشته s1 کپی می‌کند
strcat(s1, s2)	s2 را به انتهای s1 (اضافه) می‌کند
strlen(s)	طول رشته s را برمی‌گرداند
strcmp(s1, s2)	اگر s1 و s2 برابر باشند، صفر برمی‌گرداند اگر s1<s2 مقداری منفی برمی‌گرداند اگر s1>s2 مقداری مثبت برمی‌گرداند
strchr(s1, ch)	اشاره‌گری به اولین رخداد ch در s1 برمی‌گرداند
strstr(s1, s2)	اشاره‌گری به اولین رخداد s2 در s1 برمی‌گرداند
...	

مثال

```
char str1[11] = "Hello";
char str2[12] = "World";
char str3[12];
int  len ;

len = strlen(str1);
cout << len << endl;

strcpy( str3, str1);
cout << str3 << endl;

strcat( str1, str2);
cout << str1 << endl;

cout << strstr("Ali Alavi is OK!", "Alavi") << endl;
if(strstr("Ali Alavi is OK!", "Taghavi")!=NULL)
   cout<<"Contains1";
if(strstr("Ali Alavi is OK!", "Alavi")!=NULL)
   cout<<"Contains2";
```

```
5
Hello
HelloWorld
Alavi is OK!
Contains2
```

مثال: آرایه‌ای از رشته‌ها

```
char strings [3][10]={"Ali", "Taghi", "Naghi"};
for (int i = 0; i < 3; ++i)
      cout<<strings[i]<<endl;
```

```
Ali
Taghi
Naghi
```

```
char* ptrs[2];
ptrs[0] = new char[10];
strcpy(ptrs[0], "Ali");
ptrs[1] = new char[10];
strcpy(ptrs[1], "Taghi");
for (int i = 0; i < 2; ++i) {
   cout<<ptrs[i]<<endl;
   delete[] ptrs[i];
}
```

```
Ali
Taghi
```

```
char** ptr_ptrs;
ptr_ptrs = new char*[2];
ptr_ptrs[0] = new char[10];
strcpy(ptr_ptrs[0], "Ali");
ptr_ptrs[1] = new char[10];
strcpy(ptr_ptrs[1], "Taghi");
for (int i = 0; i < 2; ++i) {
   cout<<ptr_ptrs[i]<<endl;
   delete[] ptr_ptrs[i];
}
delete[] ptr_ptrs;
```

```
Ali
Taghi
```

مرتب‌سازی رشته‌ها

```
void swap(char*&s,char*&t){
   char* temp = s;
   s=t;
   t=temp;
}
```

```
void bubble_sort(char** strings, int size) {
   for (int i = 0; i < size; i++)
      for (int j = 0; j < size - i - 1; j++)
         if (strcmp(strings[j + 1], strings[j])<0)
            swap(strings[j], strings[j + 1]);
}
```

```
int main() {
   char* s[4] ={"Taghi", "Naghi", "Vali", "Ali"};
   bubble_sort(s, 4);
   for (int i = 0; i < 4; ++i)
      cout<<s[i]<<endl;
}
```

```
Ali
Naghi
Taghi
Vali
```

پایان