به نام خداوند بخشنده‌ی مهربان

شماره‌ی ۵ ۲۶ آذر ۱۳۹۹

# کد نامه

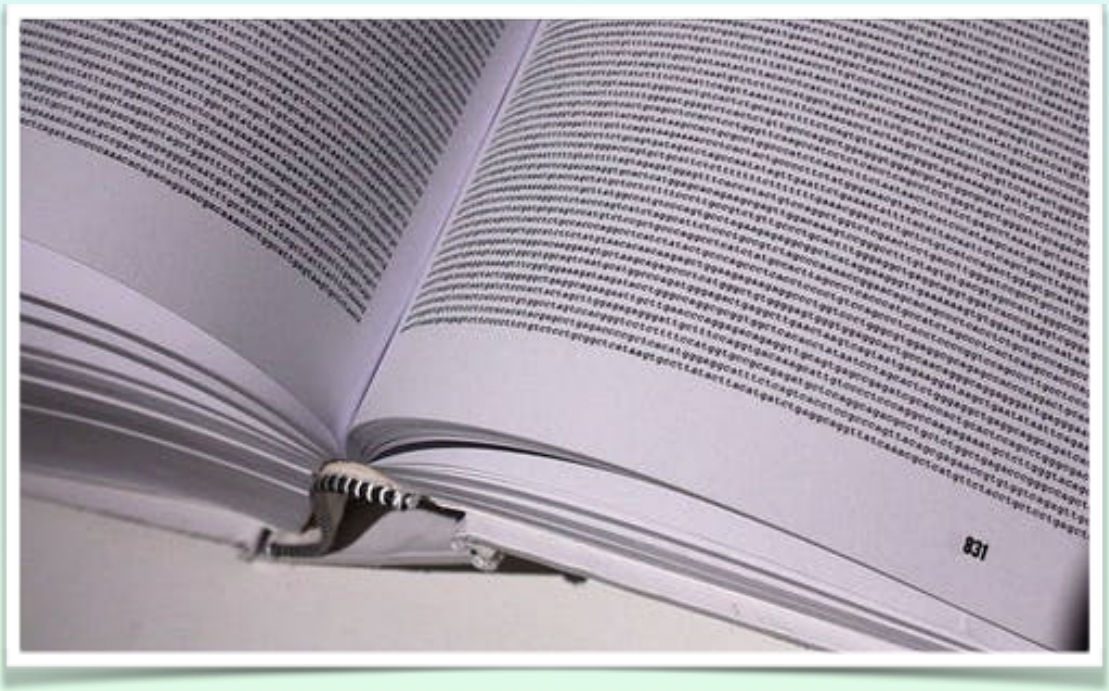
ویژه‌ی دانش‌جویان مبانی برنامه‌سازی نیم‌سال اول ۱۴۰۰-۱۳۹۹ دانشکده‌ی مهندسی کامپیوتر دانشگاه صنعتی شریف

## در این شماره از کدنامه، می‌خوانید:

به رشته بپردازیم!

## آیا می‌دانستید؟

پردازش متن و رشته، در حوزه‌ی هوش مصنوعی و به‌خصوص پردازش زبان طبیعی (NLP)، کاربرد فراوانی دارد!



## به رشته بپردازیم!

### با یکی از انواع بسیار مهم آرایه، به نام رشته، آشنا شوید

تاکنون در کلاس درس، کاربردهای متعددی از آرایه‌ها را دیده‌اید؛ عمده‌ی این کاربردها بر پردازش روی اعداد متمرکز بوده‌اند. یکی از مقوله‌های بسیار مهم در برنامه‌نویسی، پردازش روی آرایه‌ای از کاراکترهاست که به پردازش رشته یا String Processing معروف است و در درس مبانی برنامه‌سازی، لازم است با این مقوله به خوبی آشنا شوید. با کدنامه همراه باشید.

## مطالب تدریس شده در کلاس درس توسط استاد در هفته‌ی گذشته

### 22 آذر

**آرایه**

بیان نکات تکمیلی در خصوص یکی از مهم‌ترین ساختارهای داده در رایانه و برنامه‌نویسی

### 24 آذر

**الگوریتم‌های آرایه**

کسب مهارت در طراحی الگوریتم به کمک آرایه‌ها (مانند الگوریتم‌های مرتب‌سازی)

### 26 آذر

**الگوریتم‌های آرایه**

کسب مهارت در طراحی الگوریتم به کمک آرایه‌ها (مانند الگوریتم‌های مرتب‌سازی)

# به رشته بپردازیم!

**علیرضا حسین‌خانی، سید پارسا نشایی**

## رشته چیست و به چه دردی می‌خورد؟

همان‌طور که در زبان C می‌توانستید یک آرایه از اعداد تعریف کنید، می‌توانید یک آرایه از حروف (char ها) نیز تعریف کنید. رشته به دنباله‌ی از کاراکترها گفته می‌شود که برای کار با متن‌ها از آن استفاده می‌کنیم. یک رشته را می‌توانید دقیقا مثل یک آرایه‌ی ساده تعریف کنید:

```
char my_string[6] = {'h', 'e', 'l', 'l', 'o', '\0'};
```

**توجه بسیار مهم:** هنگام نگه‌داری یک رشته به صورت یک آرایه در زبان C، حتما باید مانند مثال فوق، آخرین کاراکتر آرایه را کاراکتر ویژه‌ی پوچ (بک اسلش صفر) قرار دهیم و در غیر این صورت، به مشکلات متعددی برخواهیم خورد. تعداد اعضای آرایه نیز باید حداقل یکی از تعداد حروف رشته مدنظرمان بیش‌تر باشد، تا بتواند رشته‌ی ما به همراه کاراکتر ویژه‌ی بک‌اسلش صفر در انتهای رشته را در خود نگه دارد.

**نکته:** راه دیگری برای تعریف رشته، به صورت زیر است:

```
char my_string[100] = "hello";
```

در این شیوه‌ی جای‌گزین و سریع‌تر از تعریف رشته، کاراکتر بک اسلش صفر نوشته نمی‌شود، اما در اصل، هم‌چنان وجود دارد. معمولا طول آرایه را به اندازه‌ی کافی بیش‌تر از طول رشته تعریف می‌کنیم تا اگر بعدا بخواهیم به رشته چیزی اضافه کنیم، به مشکل برنخوریم.

می‌توانیم یک رشته را به صورت زیر از کاربر ورودی بگیریم (**توجه کنید** که از کاراکتر & در ورودی گرفتن رشته، برخلاف آن‌چه در ورودی گرفتن اعداد داشتیم، استفاده **نمی‌شود**):

```
scanf("%s", my_string);
```

و می‌توانیم یک رشته را به شکل زیر، چاپ کنیم:

```
printf("%s", my_string);
```

**سوال:** کامپیوتر از کجا متوجه می‌شود که رشته تا کجا ادامه دارد که بتواند آن را به درستی چاپ کند؟

**پاسخ:** کامپیوتر از ابتدای آرایه تا جایی که به کاراکتر ویژه‌ی پوچ برسد را رشته‌ی شما در نظر می‌گیرد؛ به این معنا که مثلا اگر شما بخواهید رشته‌ی موردنظر را چاپ کنید، اسم آرایه‌ی رشته را به تابع printf می‌دهید و این تابع به ترتیب تا جایی که به کاراکتر ویژه‌ی پوچ نرسیده است، جلو می‌رود و تک تک کاراکترها را چاپ می‌کند.

**توجه:** هیچ نظارتی بر سایز آرایه‌ی کاراکترها موقع تعریف رشته نیست؛ یعنی شما می‌توانید یک آرایه‌ی ۱۰ کاراکتری تعریف کنید و انتهای آن کاراکتر ویژه‌ی پوچ قرار ندهید و با این کار، printf را به اشتباه انداخته و بیش‌تر از ۱۰ کاراکتر چاپ کنید، که به هیچ عنوان مطلوب نیست و سبب بروز خطاهای متعدد (undefined behavior) می‌شود. اگر در یک خانه‌ی حافظه، کاراکتر پوچ نوشته شده بود، یعنی آن خانه علامتی برای انتهای یک رشته است. کد اسکی این کاراکتر ویژه، عدد صفر است. **دقت کنید!** این کاراکتر با خود کاراکتر صفر فرق دارد؛ کد اسکی صفر (که کاراکتری است که به شکل صفر در خروجی چاپ می‌شود)، ۴۸ است و نه صفر.

**هشدار:** مانند سایر آرایه‌ها که نباید هرگز قبل از مقداردهی از آن‌ها استفاده کرد، از رشته‌ها نیز نباید قبل از مقداردهی اولیه، استفاده کرد تا undefined behavior پیش نیاید.

## پردازش رشته

در بسیاری از برنامه‌ها، لازم می‌شود عملیاتی روی رشته‌ها انجام دهیم که در این مقاله، پرکاربردترین توابعی که برای این کار به کمک شما می‌آیند را بیان کرده‌ایم. برای استفاده از توابع زیر، لازم است خط زیر را نیز به ابتدای کد خود، بیفزایید:

```
#include <string.h>
```

### به دست آوردن کاراکتر i ام رشته

برای این کار مانند هر آرایه‌ی دیگری، خانه‌ی i ام آرایه را می‌بینیم. توجه کنید که رشته‌ها نیز مانند سایر آرایه‌ها، از صفر شروع می‌شوند.

### چسباندن دو رشته

برای چسباندن دو رشته به هم، از تابع strcat استفاده می‌کنیم که پارامتر ورودی دوم خود را به انتهای اولی می‌چسباند (و در انتها نیز کاراکتر ویژه‌ی بک اسلش صفر را قرار می‌دهد). مثال زیر، اسم کاربر را از ورودی می‌گیرد و به او سلام می‌گوید. **توجه کنید!** باید فضای خالی در آرایه‌ی پارامتر اول تابع به اندازه‌ی کافی موجود باشد تا رشته‌ی آرایه‌ی دوم به آن اضافه شود:

```
char name[100];
```

```
char hello[100] = "Hello ";
scanf("%s", name);
strcat(hello, name);
printf("%s", hello);
```

**ریختن یک رشته در دیگری**

یکی از اشتباهات متداول دانش‌جویان در کار با رشته، استفاده از نماد تساوی (مثل str1 = str2) برای ریختن مقدار یک رشته در دیگری است که **نادرست** است و باید برای این کار از تابع strcpy استفاده کرد که رشته‌ی ورودی دوم خود را در اولی می‌ریزد:

```
strcpy(str1, str2);
```

**ریختن i کاراکتر از یک رشته در دیگری**

برای این کار، از تابع strncpy استفاده می‌کنیم که به تعداد عدد پارامتر سوم خود، حرف از رشته‌ی پارامتر دوم در رشته‌ی پارامتر اول کپی می‌کند. به عنوان مثال، کد زیر تنها ۵ کاراکتر اول str۲ را در str۱ می‌ریزد:

```
strncpy(str1, str2, 5);
```

**به دست آوردن طول (تعداد کاراکترهای) یک رشته**

تابع strlen، یک رشته را به عنوان ورودی گرفته و تعداد کاراکترهای رشته (**بدون** حساب کاراکتر ویژه‌ی پوچ در انتهای آن) را به عنوان خروجی خود، برمی‌گرداند.

```
unsigned long x = strlen(str1);
```

**مقایسه‌ی دو رشته**

گاهی لازم می‌شود دو رشته را با هم مقایسه کنید؛ اگر از نماد == که برای مقایسه‌ی اعداد از آن استفاده می‌کردید بهره گیرید، نتایج **نادرستی** دریافت خواهید کرد؛ در نتیجه لازم است از تابع strcmp استفاده کنید که اگر دو رشته‌ی ورودی این تابع، برابر بودند، مقدار صفر و اگر نابرابر بودند، یک مقدار ناصفر بر می‌گرداند.

```
int result = strcmp(str1, str2)
// result == 0 ==> equal
// result != 0 ==> different
```

**سایر عملیات رشته‌ای**

بسیاری از سایر عملیات روی رشته را می‌توانید با حلقه زدن به تعداد طول رشته (که از strlen به دست می‌آید) و بررسی تک تک کاراکترهای رشته، انجام دهید. به عنوان پیشنهاد، توصیه می‌شود سعی کنید برنامه‌های زیر را خودتان بنویسید. نوشتن برنامه‌های پیشنهادی زیر، برای افزایش تسلط در پاسخ‌گویی به سوالات امتحانات، مفید خواهند بود:

- برنامه‌ای بنویسید که مکان اولین بار ظاهر شدن کاراکتر داده شده در ورودی را در یک رشته‌ی دل‌خواه، پیدا کند.
- برنامه‌ای بنویسید که مکان آخرین بار ظاهر شدن کاراکتر داده شده در ورودی را در یک رشته‌ی دل‌خواه، پیدا کند.
- برنامه‌ای بنویسید که اولین مکان پیدا شدن کلمه‌ی داده شده در ورودی را در یک رشته‌ی دل‌خواه، پیدا کند.
- برنامه‌ای بنویسید که تمامی مکان‌های پیدا شدن کلمه‌ی داده شده در ورودی را در یک رشته‌ی دل‌خواه، پیدا کند.
- برنامه‌ای بنویسید که دو کلمه و یک رشته بزرگ از ورودی گرفته، به جای تمامی تکرارهای کلمه‌ی اول در رشته‌ی بزرگ، کلمه‌ی دوم را نوشته و حاصل را چاپ کند.
- برنامه‌ای بنویسید که کلمات رشته‌ی ورودی را از آخر به اول، به ترتیب بنویسد، مثلا hello world را گرفته و world hello را چاپ کند.

می‌توانید با برخی دیگر از توابع پرکاربرد زبان C در پردازش رشته، از طریق مشاهده‌ی این وب‌سایت، آشنا شوید.

**با Regex، آسان‌تر رشته‌ها را پردازش کنید!**

برای بسیاری از پرسش‌های مطرح شده در فوق، راه‌حل‌های ساده‌تری نیز وجود دارد؛ یکی از این راه‌حل‌ها، استفاده از عبارات منظم (Regular Expression) است. در شماره‌ی آینده‌ی کدنامه، شما را با این عبارت‌ها و کاربرد آن‌ها در پردازش رشته‌ها، آشناتر خواهیم کرد.

## ارتباط با کدنامه

خوش‌حال می‌شویم اگر پیشنهادات و انتقاداتی نسبت به کدنامه دارید یا سوال خاصی دارید که تمایل دارید در کدنامه پاسخ داده شود، به دستیاران آموزشی درس اطلاع دهید.