گام اول

برنامه نویسی

C++

محسن نصرتی • محمدرضا جوان

پیشگفتار

امروزه استفاده از فناوری‌ها، امری اجتناب ناپذیر گشته است. اگر نگاهی به اطراف خود بیاندازیم عموما تجهیزات کامپیوتری زیادی را میبینیم که زاده عصر فناوری کنونی بوده و به ما در زندگی راحت روزمره کمک می‌کنند.

با نگاهی به افراد و سطح آگاهی آن‌ها متوجه می‌شویم که افرادی که دانش بیشتری در زمینه ارتباط با تجهیزات کامپیوتری دارند، بهره بیشتری از خدمات کامپیوترها برده و زندگی آسان‌تری را برای خود ایجاد کرده‌اند.

در صورتی که قادر باشیم ارتباط خوبی با کامپیوترها برقرار کنیم و توان انتقال داده‌های ذهنی و افکار خود به آن‌ها را داشته باشیم، بیشتر می‌توانیم از آن‌ها در انجام امور بهره‌مند شویم. در این راستا زمینه‌های مختلفی برای ارتباط با کامپیوترها ایجاد گردیده است و زبان‌های واسط مختلفی برای ایجاد این ارتباط به وجود آمده است.

از بین زبان‌های مختلف به وجود آمده، زبان C پایه‌ای قوی برای زبان‌های نوین دیگر امروزه محسوب می‌شود و فراگیری آن پایه‌ای مستحکم برای ارتباط بهتر با کامپیوتر و فراگیری زبان‌های دیگر را بوجود می‌آورد.

در زمینه برنامه‌نویسی C کتاب‌های مختلفی وجود دارد اما عموما این کتاب‌ها مسائل را با فرض داشتن دانش اولیه در این زمینه بیان کرده‌اند و به مسائل از ابتدا و به صورت ساده و پایه‌ای نپرداخته‌اند.

با توجه به خلا موجود، کتاب حاضر با نگرش پایه‌ای و ساده، محیطی را فراهم می‌کند تا دست به کار شوید و گام اول را در مسیر برنامه‌نویسی C به صورت راسخ بردارید. این مجموعه به صورت خودآموز تدوین نشده است و خواندن آن همراه با آموزش‌های کلاسی توصیه می‌شود.

کتاب حاضر حاصل سال‌ها تجربه‌ی تدریس و تجربه حضور مولفان آن در مراکز استعدادهای درخشان بوده است و در برگیرنده مطالب ارائه شده برای این عزیزان نیز می‌باشد.

جا دارد از تمامی دست‌اندرکاران مراکز مختلف استعدادهای درخشان به خصوص مراکز علامه حلی و فرزانگان تهران که محیط مناسبی را برای گسترش این دانش و پیشبرد دانش روز بین نخبگان جوان برای ما ایجاد کرده‌اند، تشکر کنیم.

همچنین از همکاران گرانقدرمان آقایان سعید سرکاراتی، حمیدرضا نصرتی، عبدالله آراسته، محسن غفوریان و خانم‌ها سارا ابراهیمی و آرزو امیرجاملویی که همفکری با ایشان همواره مسیر روشن‌تری را پیش رویمان قرار داده است، کمال سپاس را داریم.

لازم به ذکر است مجموعه‌های مکمل این کتاب و نرم‌افزارها در سایتی به آدرس ctalk.ir برای مخاطبین گرامی گردآوری شده است. در نگارش این کتاب سعی بر آن بوده که با دقت، تمام نکات در نظر گرفته شود و خطا و نقص به حداقل رسانده شده باشد، اما تیزبینی شما خوانندگان محترم ما را در رفع نقص‌های باقیمانده کمک خواهد کرد. همچنین پیشنهادات شما عزیزان راه هموارتری را برایمان ایجاد خواهد کرد. امید است این نگاشته محیط مناسبی را برای شروع فعالیت علاقه‌مندان فراهم آورد.

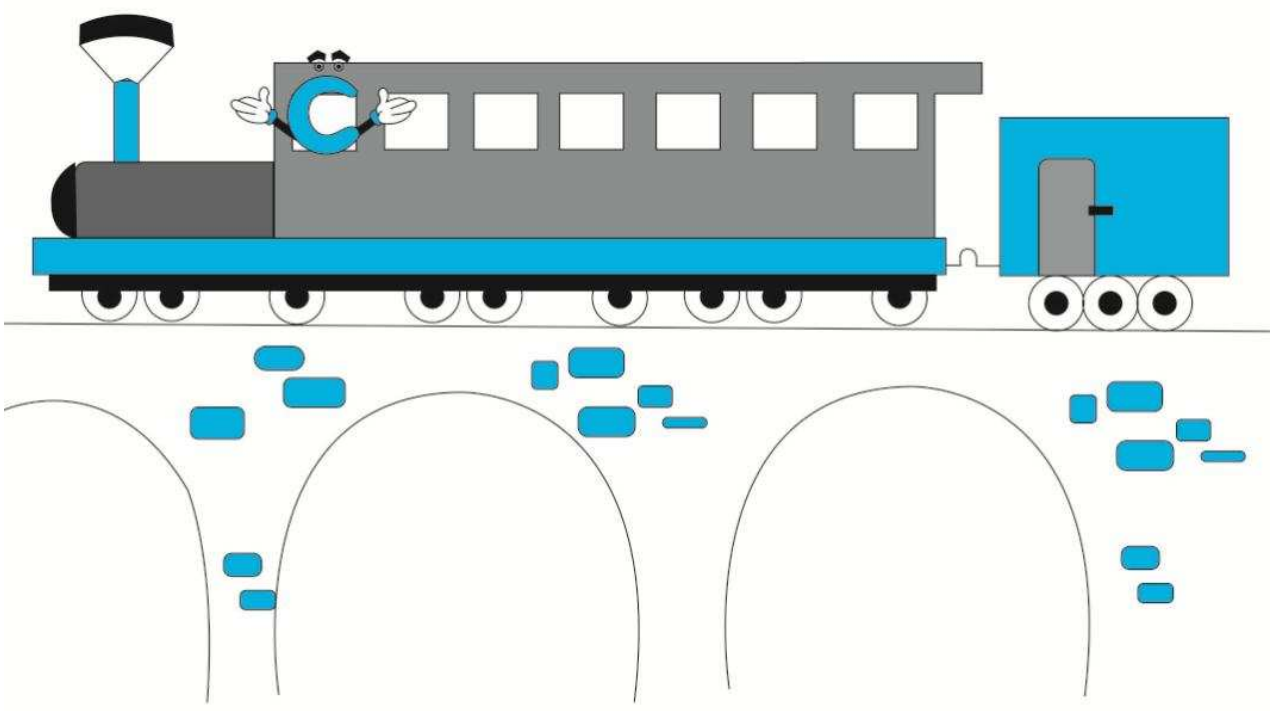
محسن نصرتی – محمدرضا جوان

0110

فصل ششم

قطار حافظه

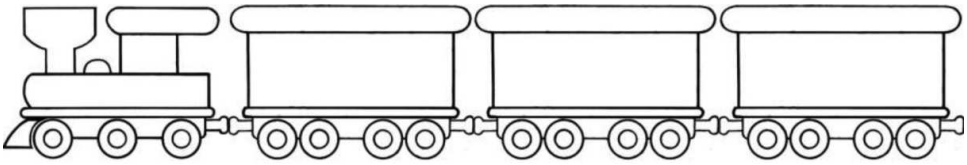
0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110 0110



قطار حافظه

اسم‌گذاری خانه‌های حافظه به ما، برای ارتباط راحت‌تر با حافظه کمک زیادی می‌کند. فرض کنید بخواهیم نمره سه دانش آموز را بگیریم و بعد آن‌ها را رتبه‌بندی کنیم. می‌دانید برای اینکار ۳ خانه حافظه برای نگهداری این اعداد لازم است. حال اگر این کار را بخواهیم برای ۱۰۰ نفر یا بیشتر انجام دهیم، در نظر گرفتن این تعداد خانه منطقی است؟

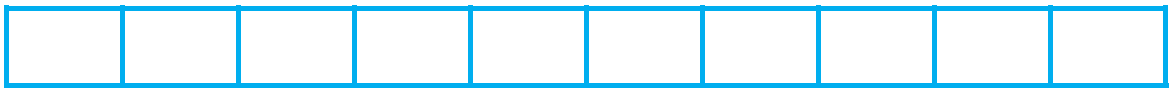
همانطور که به نظر می‌رسد، تعریف ۱۰۰ متغیر در مثال بالا امری زمان‌بر و همچنین غیر منطقی به نظر می‌رسد. زبان C برای حل این مشکل، نحوه تعریف مجموعه‌ای بهم پیوسته از متغیرها را که نام آن آرایه است، در نظر گرفته است.



شکل ۶-۱: قطاری متشکل از واگنه‌های پیاپی

۶-۲. آرایه چیست؟

آرایه مجموعه به هم پیوسته‌ای از متغیرهاست که دارای نام یکسان هستند و هر کدام یک شماره مربوط به خود را دارند که با آن از هم تفکیک می شوند. این مجموعه متغیرهای هم نام، لزوما از یک نوع می‌باشند.



شکل ۶-۲: نمایی از یک آرایه با ۱۰ خانه حافظه

هر قطاری از خانه‌های حافظه باید دارای یک نام مشخص و تعداد خانه‌های مشخص در زمان تعریف باشد. به صورت مثال آرایه a با ۱۰ خانه از اعداد صحیح به صورت زیر تعریف می‌شود.

روش تعریف آرایه

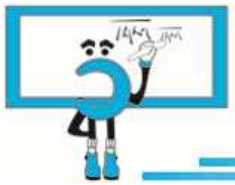
نوع متغیر	نام متغیر و تعداد خانه‌ها
int	a [10];

باید توجه داشته باشید که شماره این خانه‌ها از صفر شروع می‌شود و در زمانی که آرایه ۱۰ خانه دارد، شماره خانه‌ها از ۰ تا ۹ می‌باشد.

خانه‌های آرایه a هر کدام به کدام شکل a[0]، a[1]، a[2]، ... و a[9] استفاده می‌شوند و هر یک تمام خواص یک متغیر را به تنهایی دارند.

◄ توجه

توجه داشته باشید که زمانی که آرایه را به صورت a[10] تعریف می‌کنیم، آخرین خانه‌ای که می‌تواند مورد استفاده قرار بگیرد a[9] می‌باشد. همچنین استفاده از خانه‌های با شماره منفی نیز غلط می‌باشد.



◄ اشتباه ننویسیم!

در آرایه با ۱۰ خانه، خانه شماره ۱۰ وجود ندارد.

در شماره آرایه‌ها خانه با شماره منفی مجاز نیست.

```
int a [10] ;
a [10] = 8 ;
a [-1] = 5 ;
```

◄ مثال

برنامه‌ای بنویسیم که نمرات دانش آموزان کلاسمان را در ابتدا گرفته و سپس نمره ما را بگیرد و مشخص کند نمره چند نفر از ما بیشتر شده است. (در این مثال کلاس ۱۵ نفره در نظر گرفته شده است.)

برنامه ٦-١ رتبه در کلاس

```
#include <iostream>
#include <conio.h>
using namespace std;
main()
{
 int  c [15], i, rank=0, score;
 cout << “Enter all students scores(15):”;
 for (i=0; i<15; i++)
 {
    cin >> c[i];
 }
 cout << “Enter your score:”;
 cin >> score;
 for (i=0; i<15; i++)
 {
    if ( score  > c[i] )
    {
      rank++;
    }
 }
 cout << “Your rank is:” << rank;
 getch();
}
```

در این مثال ممکن است این نکته مطرح شود که چرا نمره شما را در ابتدا نگرفتیم تا سپس آن را با نمره سایرین مقایسه کنیم؟

برای پاسخ به سوال فوق باید به این نکته توجه داشته باشید که در برخی از موراد نیازی به نگهداری کلیه اعداد ورودی در طول برنامه نیست. در این صورت نیازی به تعریف آرایه نیست. (مانند مثال پیدا کردن بزرگترین عدد بین اعداد)

اما در برخی مواقع لازم است تمام اعداد ورودی را در طول برنامه در اختیار داشته باشیم. برای بهتر متوجه شدن موضوع برنامه بالا را به گونه‌ای تغییر دهید که بتوان بعد

از وارد کردن نمره تمام دانش آموزان، نمره افراد مختلف را به ترتیب وارد کرده و سیستم رتبه او را به ما بگوید.

◄ بیشتر بدانیم

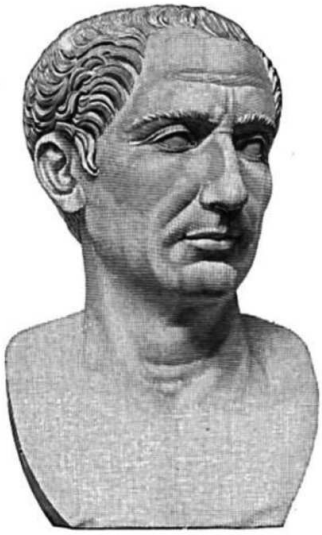
در فصل‌های قبل دیدیم که در برخی از برنامه‌ها نیاز است تا متغیرها توسط برنامه‌نویس مقداردهی شوند. حال ممکن است این سوال پیش بیاید که برنامه‌نویس چگونه می‌تواند آرایه‌ها را در برنامه خود مقداردهی کند؟

```
int a[3] = {11,22,33};
```

همچنین برای صفر کردن همه خانه‌های آرایه میتوان از دستور زیر استفاده کرد.

```
int a[3] = {0};
```

۶-۳. رمزنگاری به سبک سزار



شکل ۶-۳: ژولیوس سزار

رمزنگاری با رمز سزار یکی از ساده‌ترین و شناخته شده‌ترین تکنیک‌های رمزنگاری است که با عناوین رمز جابجایی، کد سزار یا جابجایی سزار شناخته می‌شود.

این رمز یک نوع رمز جانشینی است که هر حرف در متن اصلی با حرف دیگری با فاصله ثابت جایگزین می‌شود. برای مثال با مقدار انتقال ۳، A با D، D با G و به همین ترتیب جانشین می‌شوند. نام این نحوه رمزگزاری از ژولیوس سزار یکی از فرماندهان روم گرفته شده است. ژولیوس سزار از این شیوه رمزنگاری برای ارتباط با فرماندهان خود استفاده می کرده است.

متن اصلی: ABCDEFGHIJKLMNOPQRSTUVWXYZ

متن رمز شده: DEFGHIJKLMNOPQRSTUVWXYZABC

◄ دست به کار شویم

برنامه‌ای بنویسیم که یک جمله حداکثر ۲۰ کاراکتری که با نقطه (.) تمام می‌شود را گرفته و رمز سزار آن را تولید کند.

برای این کار لازم است ابتدا جمله را از کاربر بگیریم و سپس هر یک از کاراکترها را با توجه به کد اسکی به سه حرف جلوتر ببریم. توجه داشته باشید که در این برنامه صرفا حروف کوچک را رمزنگاری می‌کنیم.

برنامه ٦-٢ رمز سزار

```
#include <iostream>
#include <conio.h>
using namespace std;
main()
{
 char  s [20], ch=0;
 int l=0, i;
 cout << “Enter words to code (Ends with .):”;
 while (ch != '.' && l < 20)
 {
    ch = getch();
    s[l]=ch;
    cout << ch;
    l++;
```

```
    }
    cout << endl << "Ceasar code: " ;
    for (i=0; i<l; i++)
    {
       if  ( 97 <= s [i] && s [i] <= 122 )
         ch = ((s[i] - 97 + 3) % 26) + 97;
       else
         ch = s [i];
       cout << ch;
    }
    getch();
}
```

البته باید توجه داشته باشید که همانند همه رمزهای با جانشینی حروف، رمز سزار نیز به راحتی شکسته میشود و به طور اساسی هیچ امنیت ارتباطی‌ای ایجاد نمی‌کند. آیا میتوانید رمز سزار تولید شده را بشکنید؟

۶-۴. زنجیره حروف

با توجه به پرکاربرد بودن استفاده مجموعه کاراکترها برای گرفتن کلمات و جمله‌ها، در زبان C مفهومی به نام رشته[۳۰] وجود دارد. رشته در اصطلاح به مجموعه‌ای از کاراکترها گفته می‌شود که در کنار هم قرار می‌گیرند.

شکل ۶-۴: زنجیره حروف

راه ساده استفاده از رشته‌ها در زبان C استفاده از یک آرایه کاراکتری می‌باشد. در زبان C امکان ورودی گرفتن برای آرایه کاراکتری به صورت یکجا توسط cin ایجاد شده است.

```
char s[10];
cin >> s;
```

30 String

حال اگر در زمان اجرا در ورودی مقدار ctalk وارد شود لازم است آرایه بداند که تا کدام خانه مقدار ورودی جای گرفته و از کدام خانه به بعد آرایه بدون استفاده است.

c	t	a	l	k	?	?	?	?	?

C برای مشخص کردن انتهای یک رشته کاراکتری با محتوای NULL دارد که به معنی پوچ است. وقتی به این کاراکتر در آرایه برسیم به این معنی است که رشته به انتها رسیده است.

c	t	a	l	k	NULL	?	?	?	?

چگونه انتهای زنجیره را پیدا کنیم؟

حال برای پیدا کردن انتهای یک رشته لازم است با یک حلقه به دنبال خانه پوچ در آن بگردیم. توجه داشته باشیم که کاراکتر خانه پوچ را با ’\0‘نیز نمایش میدهند و پیدا کردن این کاراکتر در آرایه پایان آرایه را نیز مشخص می‌کند.

ارتباط با زنجیره حروف

در C کتابخانه‌ای وجود دارد که کمی کار را با زنجیره حروف راحت تر میکند. این کتابخانه string.h نام دارد و برای استفاده از آن باید خط زیر را به ابتدای برنامه‌های خود اضافه کنیم.

با استفاده از string.h یک روش راحت‌تر به نام strlen، برای پیدا کردن طول رشته حروف وجود دارد. استفاده از strlen به صورت زیر برای ما طول رشته را به صورت ساده‌تر محاسبه می‌کند و مقدار آن را در متغیر l می‌ریزد.

```
int l;
l = strlen (s);
```

باید توجه داشته باشیم که امکان ریختن مستقیم یک رشته درون یک آرایه وجود ندارد. برای حل مشکل می‌توانیم از strcpy به صورت زیر استفاده کنیم. strcpy مقدار دوم درون دستان خود را درون مقدار اول جایگزین می‌کند.

```
strcpy ( s , “ctalk”);
```

با استفاده از دستور فوق رشته "ctalk“ درون آرایه s قرار داده می‌شود.