برنامه سازی پیشرفته به زبان C
(DRAFT)

رحمت قاسمی

۴ دی ۱۳۹۵

فهرست مطالب

فصل ۱

مقدمه

فصل ۲

مفاهیم اولیه

۱.۲ متغییر و ثابت

متغییر، محلی از حافظه است که برای ذخیره داده‌ها استفاده می‌شود. یک متغییر دارای نام، نوع داده، مقدار و آدرس حافظه است. نام متغییر ترکیبی از حروف الفبا و اعداد است و با یک حرف شروع می‌شود. قبل از اینکه بتوانیم از یک متغییر استفاده کنیم، باید آن را تعریف کنیم. تعریف متغییر در زبان C به صورت زیر است:

نوع داده نام متغییر=مقدار اولیه;

مثال ۲.۱. تعریف متغییر

```
1 int    age=5;
2 float  pi=3.14;
```

در خط اول یک متغییر با نام age و از نوع int(که یک عدد صحیح را نشان می‌دهد) با مقدار اولیه ۵ تعریف شده است. آدرس متغییر در اختیار ما قرار ندارد و نمی‌توانیم آن را مشخص نماییم. در خط دوم یک متغییر با نام pi و از نوع $float$ (که یک عدد اعشاری را نشان می‌دهد) با مقدار اولیه ۳.۱۴ تعریف شده است. ■

علاوه بر نوع داده int می‌توانیم از سایر انواع داده‌ای مانند $(char, float, double)$ استفاده نماییم. در بخش ؟؟ انواع داده‌ها در زبان C آورده شده است.

۲.۲ انواع داده‌ها

زبان C از انواع داده‌ای صحیح و اعشاری پشتیبانی می‌کند. انواع داده‌ها در زبان C عبات است از: $char, int, float, double$. از $char$، int برای ذخیره اعدد صحیح و $float$، $double$ برای ذخیره اعداد اعشاری استفاده می‌شود. علاوه بر این در هنگام تعریف متغییر می‌توان از $signed, unsigned, short, long$ به عنوان پیشوند، استفاده نمود. $signed$ برای اعداد صحیح علامت‌دار

نوع داده	حدپایین	حدبالا	طول بر حسب بایت
char	۱۲۸_	۱۲۷	۱
unsigned char	۰	۲۵۵	۱
int	۳۲۷۶۸_	۳۲۷۶۷	۲
unsigned int	۰	۶۵۵۳۵	۲
long int	۲۱۴۷۴۸۳۶۴۸_	۲۱۴۷۴۸۳۶۴۷	۴
unsigned long int	۰	۴۲۹۴۹۶۷۲۹۵	۴
float	$۳٫۴ * ۱۰^{-۳۸}$	$۳٫۴ * ۱۰^{+۳۸}$	۴
double	$۱٫۷ * ۱۰^{-۳۰۸}$	$۱٫۷ * ۱۰^{+۳۰۸}$	۸
long double	$۳٫۴ * ۱۰^{-۴۹۳۲}$	$۱٫۱ * ۱۰^{۴۹۳۲}$	۱۰

استفاده می‌شود و *unsigned* برای اعداد بدون علامت کاربرد دارد. اگر پیشوندی مشخص نشود *signed* در نظر گرفته می‌شود. هر یک از پیشوندها را می‌توان برای نوع داده *int* استفاده نمود. *signed, unsigned* برای نوع داده *char* هم قابل استفاده است. همچنین *long* به همراه *double* هم قابل استفاده است. در جدول ؟؟ انواع داده‌ای قابل قابل استفاده در زبان *C* مشخص شده است. در هنگام تعریف یک متغییر باید ی به دامنه آن توجه داشته باشیم و نوع داده مناسب آن را استفاده نماییم.

مثال ۲.۲. تعریف متغییر

```
1 long int  f;
2 unsigned int s;
```

در مثال فوق f *یک متغییر صحیح علا ت دار است که ۴ بایت حافظه اشغال می‌کند و* s *یک متغییر عددی بدون علامت است که ۲ بایت حافظه مصرف می‌کند.* ■

۳.۲ عملگرها

عملگرها برای پردازش اطلاعات استفاده می‌شوند. در زبان C عملگرهای مختلفی وجود دارند که در جدول ۱.۲ دسته‌بندی‌های مختلفی از آنها لیست شده است.

۴.۲ ساختار برنامه

یک برنامه به زبان C مجموعه‌ای از دستورها است که در داخل *main* که به آن تابع *main* هم گفته می‌شود، نوشته می‌شود. در ادامه مدل ساده‌ای از یک برنام به زبان C را مشاهده می‌کنید.

```
1 main()  {
2    statement;
3    statement;
```

جدول ۱.۲: انواع عملگرها در زبان C

رده	عملگرها
عملگرهای محاسباتی	+(جمع) -(تفریق) *(ضرب) /(تقسیم) ٪(باقیمانده تقسیم) ++(افزایش یک واحد) --(کاهش یک واحد)
عملگرهای مقایسه‌ای	==(برابری) <(کوچکتراز) >(بزرگتر از) =<(کوچکتر یا مساوی) =>(بزرگتر یا مساوی) =!(نابرابری)
عملگرهای منطقی	&&(و) ||(یا) !(نقیض)
عملگرهای بیتی	&(و) |(یا) (مکمل) (یای انحصاری) <<(شیفت به راست) >>(شیفت به چپ)
عملگرهای انتساب	=(انتساب) =+ =- =* =/ =٪ =>> =<< =& =|
عملگرهای دیگر	&(آدرس موثر) sizeof(اندازه یک متغییر بر حسب بایت) *(محتوای یک آدرس) :؟ (عبارت شرایطی)

```
4     statement;
5     ...
6  }
```

دستور: یک دستور در زبان C می‌تواند ساده، مرکب و یا پوچ باشد.

یک دستور ساده شامل یک عبارت است که در انتهای آن علامت ;(سمی‌کالن) قرار گرفته است. مثال:

```
1  y=2+3;
2  3+4;
3  x=v*t+x0;
4  delta= b*b-4*a*c;
5  printf("Hello");
```

در خط اول یک دستور محاسباتی وجود دارد، در این دستور مقدار محاسبه شده در متغییر y ذخیره می‌شود. در خط دوم هم یک عبارت محاسباتی قرار دارد ولی تاثیری در برنامه ندارد و قابل حذف است. در خط سوم و چهارم هم دستورات محاسباتی دیگری مشاهده می‌کنید. در خط پنجم تابع $printf$ فراخوانی شده است. تابع $printf$ عملیات خروجی را انجام می‌دهد.

یک دستور مرکب می‌تواند شامل ترکیبی از دستورات ساده و مرکب باشد. یک دستور مرکب در داخل بلاک نوشته می‌شود. شروع یک بلاک با علامت { و پایان بلاک با علامت } مشخص می‌شود. ساختارهای کنترلی مانند $if, for, while, do$ به عنوان دستور مرکب تلقی می‌شوند. مثال:

```
1  {
2    y=2+3;
3    x=v*t+x0;
4  }
```

یک دستور پوچ توسط علامت ؛ مشخص می‌شود.

مثال ۲.۳. *دایره‌ای به شعاع ۱۵ سانتی‌متر موجود است، برنامه‌ای بنویسید که مساحت دایره را محاسبه*

نماید و در خروجی نمایش دهد.

```
1 int main() {
2    float r,s;
3    r=15;
4    s=3.14 * r * r;
5    printf("%f", s);
6    return 0;
7 }
```

■

مثال ۲.۴. یک مستطیل به طول ۱۵ متر و عرض ۸ متر موجود است، برنامه‌ای بنویسید که مساحت و محیط این مستطیل را محاسبه نماید و در خروجی نمایش دهد.

```
1 int main() {
2    float tol, arz, mohit, masahat;
3    tol=15;
4    arz=8;
5    mohit = 2*(tol + arz);
6    masahat= tol * arz;
7    printf("%f %f", mohit, masahat);
8    return 0;
9 }
```

■

فصل ۳

توابع ورودی و خروجی

۱.۳ تابع printf

تابع printf برای ارسال اطلاعات به خروجی استاندارد استفاده می‌شود. شکل کلی تابع printf به صورت زیر است.

```
1  printf(format string, arg1, arg2, ... );
```

format string حاوی اطلاعاتی ارسالی به خروجی است و رشته قالب بندی نامیده می‌شود. که ممکن است حاوی علامتهای کنترلی و فرمت باشد. و arg1,2,... سایر اطلاعات ارسالی است که با توجه به رشته قالب بندی به خروجی ارسال خواهند شد. علامتهای فرمت قابل استفاده در جدول ۱.۳ مشخص شده است. مثلا اگر قرار باشد یک عدد صحیح را در خروجی بنویسیم از علامت %d در رشته قالب بندی استفاده می‌کنیم. به ازای هر علامت قالب بندی که در رشته قالب بندی استفاده می‌کنیم باید یک مقدار متناظر در بخش arg داشته باشیم که می‌تواند یک مقدار ثابت و یا یک متغیر باشد.

برای روشن شدن بیشتر به مثالهای زیر توجه نمایید:

```
1  x=5; y=6;
2  printf("HELLO");
3  printf("the value of x is %d", x);
4  printf("sum of %d,%d is %d", x, y, x+y);
5  printf("PI value is %f", 3.1415925);
```

جدول ۱.۳: برخی از علامتهای قالب بندی قابل استفاده

علامت قالببندی	شرح
%c	کاراکتر
%d	عدد صحیح
%f	عدد اعشاری
%s	رشته کاراکتری

```
6 printf("My name is %s", "ALI");
7 printf("This is line1.\nThis is line2.");
```

۲.۳ تابع scanf

تابع scanf اطلاعات را از ورودی استاندارد دریافت می‌کند. شکل کلی تابع scanf به صورت زیر است.

```
1 scanf(format string, arg1, arg2, ...);
```

که format string رشته قالب بندی را مشخص می‌کند و حاوی علامتهای فرمت می‌باشد. به ازای هر علامت قالب بندی استفاده شده در رشته قالب بندی بایستی آدرس یک متغییر را در بخش arg مشخص نمایید. برای مثال برای دریافت یک عدد صحیح از ورودی و ذخیره آن در متغییر p از دستور زیر استفاده می‌کنیم. برای مشخص کردن آدرس یک متغییر از عملگر & استفاده می‌کنیم. یعنی &p آدرس متغییر p در حافظه را مشخص می‌کند.

```
1 scanf("%d", &p);
```

برای روشن شدن بیشتر مطلب به مثالهای زیر توجه نمایید.

```
1 int x;
2 int y;
3 float pi;
4 scanf("%d", &x);
5 scanf("%d", &y);
6 scanf("%f", &pi);
```

در خط یک تا سه متغییرهای x,y,pi تعریف شده‌اند. x از نوع صحیح، y از نوع صحیح و pi از نوع اعشاری تعریف شده‌اند. در خط چهارم مقدار متغییر x از ورودی خوانده خواهد شد. در خط پنجم یک عدد صحیح از ورودی دریافت می‌شود و در متغییر y ذخیره می‌شود. در خط ششم یک مقدار اعشاری از ورودی خوانده شده و در متغییر pi ذخیره خواهد شد.

مثال ۳.۱. *دایره‌ای موجود است، برنامه‌ای بنویسید که اندازه شعاع دایره را از ورودی دریافت کرده و مساحت دایره را محاسبه نماید و در خروجی نمایش دهد.*

```
1 int main() {
2    float r,s;
3    scanf("%f", &r);
4    s=3.14 * r * r;
5    printf("%f", s);
6    return 0;
7 }
```

■

مثال ۳.۲. *یک مستطیل موجود است، برنامه‌ای بنویسید که اندازه طول و عرض مستطیل را از ورودی دریافت نموده و مساحت و محیط این مستطیل را محاسبه نماید و در خروجی نمایش دهد.*

```
1 int main() {
2    float tol, arz, mohit, masahat;
3    scanf("%f", &tol);
4    scanf("%f", &arz);
5    mohit = 2*(tol + arz);
6    masahat= tol * arz;
7    printf("%f %f", mohit, masahat);
8    return 0;
9 }
```

■

فصل ۴

دستورات شرطی

۱.۴ دستور if

دستور if به صورت زیر می‌باشد:

if (عبارت شرطی) دستور; else دستور;

توضیح: اگر عبارت شرطی درست باشد دستور بعد از آن اجرا می‌شود و اگر عبارت شرطی نادرست باشد دستور بعد از $else$ اجرا می‌شود. در شکل ۱.۴ نمودار گردشی دستور if به تصویر کشیده شده است. قسمت $else$ در دستور if اختیاری است.

مثال ۴.۱. *مثالی از دستور if*

```
1  if(x==0) y=1; else y=2;
```

اگر مقدار x برابر صفر باشد، عدد یک در متغییر y ذخیره می‌شود در غیر اینصورت عدد ۲ در متغییر y ذخیره می‌شود. ■

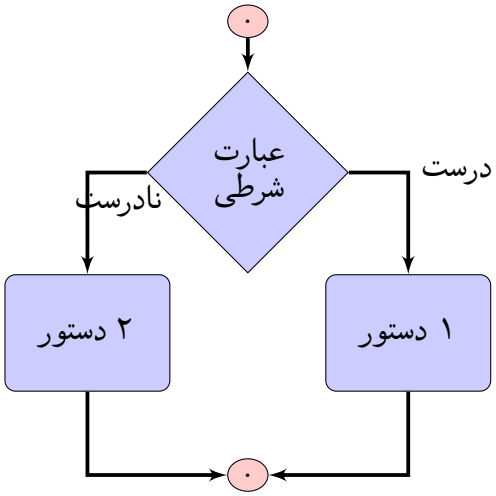
مثال ۴.۲. *برنامه‌ای بنویسید که زوج یا فرد بودن یک عدد را مشخص نماید.*
عددی زوج است که باقیمانده تقسیم آن بر عدد دو برابر صفر شود.

```
1  int main() {
2      int x=5;
3      r= x % 2;
4     if (r==0) printf("ZOJ"); else printf("FARD");
5     return 0;
6  }
```

■

مثال ۴.۳. *برنامه‌ای بنویسید که ریشه‌های یک معادله درجه ۲ را بیابد.*
یک معادله درجه دو به صورت $ax^2+bx+c=0$ تعریف می‌شود.که a,b,c ضرایب معادله می‌باشند و فرض می‌کنیم که $a \neq 0$. برای محاسبه ریشه های این معادله ابتدا δ(دلتا) را از معادله ۱.۴ محاسبه می‌کنیم.

$$\delta = b^2 - 4ac \qquad (۱.۴)$$



شکل ۱.۴: نمودار گردشی دستور if

حال سه حالت اتفاق می افتد.

- اگر δ کوچکتر از صفر باشد معادله دارای ریشه نمی باشد.
- اگر δ برابر صفر باشد، معادله دارای ریشه مضاعف است.
- اگر δ بزرگتر از صفر باشد، معادله دارای دو ریشه می باشد. هر یک از ریشه ها از معادله ۲.۴ قابل محاسبه است.

$$x_1 = \frac{-b+\sqrt{\delta}}{2a}, x_2 = \frac{-b-\sqrt{\delta}}{2a} \quad (۲.۴)$$

برنامه مربوط به محاسبه ریشه های معادله درجه ۲ در ادامه آورده شده است.

```
1  int main() {
2    float a=1;
3    float b=2;
4    float c=1;
5    float delta;
6    float x1, x2;
7
8    delta= b*b - 4*a*c;
9
10   if (delta<0) {
11     printf("\n Error: no solution.");
12     return 0;
13   }
```

```
14
15    if (delta==0) {
16      printf("\ndouble soulution.");
17      x1=-b/(2*a);
18      printf("\n x1=%f", x1);
19      return 0;
20    }
21
22    if (delta>0) {
23      x1=(-b+sqrt(delta))/(2*a);
24      x1=(-b-sqrt(delta))/(2*a);
25      printf("\n x1=%f", x1);
26      printf("\n x2=%f", x2);
27     return 0;
28    }
29  }
```

همانطور که در برنامه مثال ۴.۳ مشاهده می‌کنید. در خطوط ۲ الی ۶ برنامه، متغییرهای مورد استفاده تعریف شده است. در زبان Cقبل از استفاده از متغییر بایستی آن را تعریف کنیم. در خط ۸ مقدار محاسبه شده است. در خطوط ۱۵، ۱۰ و ۲۲ هر یک حالات سه‌گانه مطرح شده بررسی شده است. و در نهایت خروجی مناسب اعلام شده است. ■

۲.۴ دستور switch

۱.۲.۴ دستور break

فصل ۵

دستورات کنترل تکرار

این دستورات برای کنترل تکرار مجموعه‌ای از دستورات استفاده می‌شوند. دستورات تکرار در زبان C عبارتند از $do-while, while, for$.

۱.۵ دستور do-while

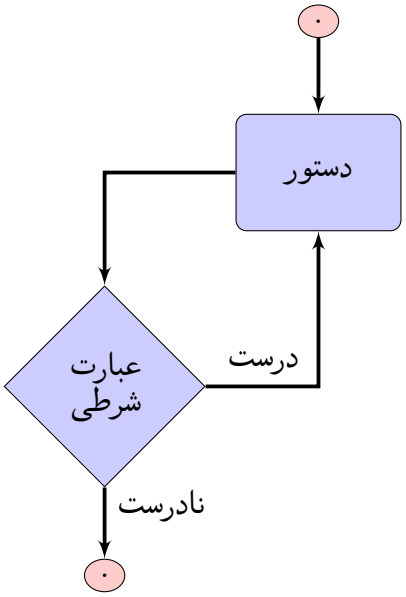
دستور $do-while$ زمانی استفاده می‌شود که تعداد تکرار دستورات نامشخص باشد. در این دستور کنترل اجرای دوباره دستورات در پایان دستور مشخص می‌شود. شکل کلی این ساختار در شکل ۱.۵ نشان داده شده است. دستور موجود در بدنه حلقه do حداقل یک بار اجرا می‌شود. در صورتی که شرط مشخص شده در انتهای ساختار ارزش درستی داشته باشد، دستور دوباره اجرا می‌شود و این کار تا زمان درست بودن شرط تکرار انجام می‌شود.

```
1 do {
2     statement;
3 } while (condition);
```

مثال ۵.۱. *محاسبه مجموع اعداد فرد کوچکتر از ۱۰۰*

```
1  #include <stdio.h>
2  int main() {
3     int s=0;
4     int k=1;
5     do {
6          s=s+k;
7          k=k+2;
8     } while (k<100);
9     printf("%d", s);
10    return 0;
11 }
```

■



شکل ۱.۵: نمودار گردشی دستور do-while

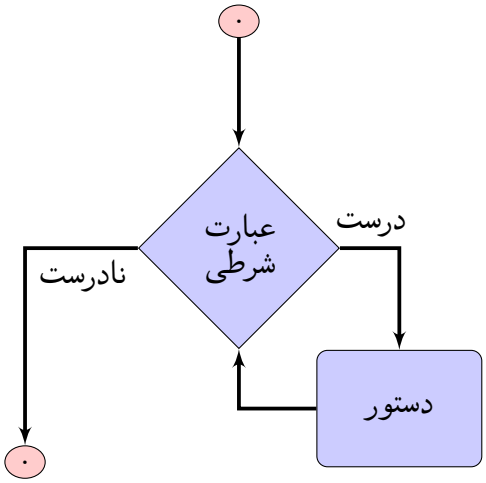
مثال ۵.۲. *برنامه‌ای که یک عدد از ورودی دریافت می‌کند و مقسوم علیه‌های آن را محاسبه می‌کند. می‌گوییم عدد k مقسوم علیه عدد n است اگر باقیمانده تقسیم n بر عدد k برابر صفر شود.*

```
1  #include <stdio.h>
2  int main() {
3      int n;
4      scanf("%d", &n);
5      int k=1;
6      do {
7           if (n % k== 0) {
8          printf("%d", k);
9          if (n/k != k) printf(" %d", n/k);
10          }
11          k=k+1;
12     } while (k*k <= n);
13     return 0;
14 }
```

■

۲.۵ دستور while

دستور *while* هم زمانی استفاده می‌شود که تعداد تکرار دستورات نامشخص باشد. در این دستور کنترل اجرای دوباره دستورات در ابتدا مشخص می‌شود. نمودار گردشی دستور while در شکل ۳.۵



شکل ۲.۵: نمودار گردشی دستور while

نشان داده شده است. در این ساختار ممکن است دستورات اصلا اجرا نشوند. در صورتی که شرط مشخص شده در ابتدای ساختار ارزش درستی داشته باشد، اجرای دستورات از ابتدای بلاک مجددا تکرار می‌شود.

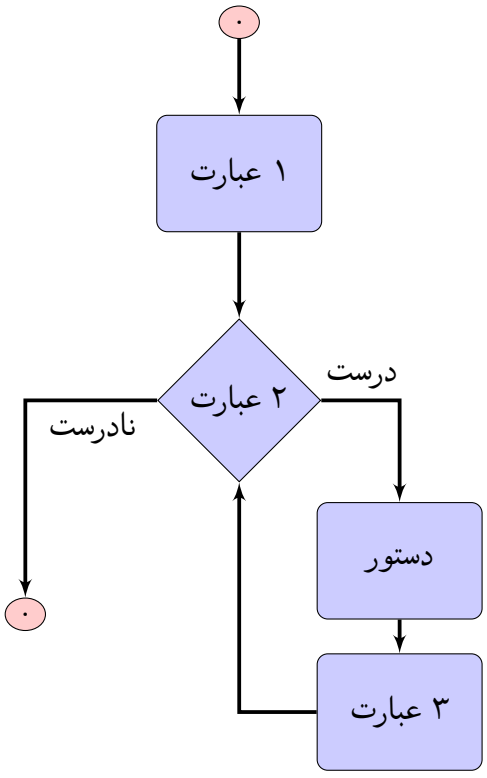
```
1   while (condition)
2      statement;
```

۳.۵ دستور for

```
1     for( expresion ; expresion ; expresion )
2        statement;
```

۴.۵ دستور break

۵.۵ دستور continue



شکل ۳.۵: نمودار گردشی دستور for

فصل ۶

توابع

فصل ۷

آرایه، اشاره‌گر و رشته‌ها

۱.۷ آرایه

آرایه، یک لیست از عناصر هم نوع است که دسترسی به هر عنصر از طریق اندیس(های) عددی امکان پذیر است. آرایه ممکن است یک‌بعدی یا چند بعدی باشد. به آرایه یک‌بعدی، بردار و به آرایه دوبعدی ماتریس گفته می‌شود. برای تعریف آرایه کافی است نوع، نام و تعداد عناصر آرایه را مشخص کنیم. متغییر اندیس‌دار، نام دیگری برای آرایه است.

مثال ۷.۱. *تعریف آرایه یک‌بعدی با ۱۰ عنصر از نوع صحیح به نام a*

```
1  int a[10];
```

در تعریف فوق آرایه به نام a با پنج عنصر ایجاد می‌شود. اولین عنصر آرایه در اندیس صفر قرار دارد و به از طریق $a[0]$ در دسترس است. همچنین آخرین عنصر آرایه در اندیس چهارم قرار دارد. ■

مثال ۷.۲. *تعریف آرایه دوبعدی با پنج سطر و شش ستون از نوع اعشاری به نام f*

```
1  float f[5][6];
```

در تعریف فوق آرایه f با ۳۰ عنصر ایجاد می‌شود، آرایه f دارای پنج سطر است و در هر سطر آن شش عدد اعشاری قرار دارد. اولین عنصر آن در سطر صفر و ستون صفر قرار دارد و به صورت $f[0][0]$ در دسترس است. ■

مثال ۷.۳. *برنامه‌ای بنویسید که ده عدد صحیح از ورودی دریافت کرده و دریک آرایه ذخیره نماید.*

```
1  int a[10];
2  int i,x;
3  int main() {
4     for (i=0; i<10; i++) {
5        scanf("%d", &x);
6        a[i]=x;
7     }
8  }
```

در برنامه فوق در خط ۱ آرایه a با ده عنصر تعریف شد. در خط ۲ متغییرهای کمکی x,i تعریف شده‌اند. سپس در یک حلقه تکرار در خط ۵ یک عدد صحیح از ورودی دریافت شد و در خط ۶ این عدد در اندیس iام آرایه a یعنی در $a[i]$ ذخیره می‌شود. ■

مثال ۷.۴. *برنامه‌ای بنویسید که ده عدد صحیح از ورودی دریافت کند و در یک آرایه ذخیره نماید، سپس مجموع اعداد آرایه را محاسبه و چاپ نماید.*

```
1  int a[10];
2  int i,x,sum;
3  int main() {
4     for (i=0; i<10; i++) {
5        scanf("%d", &x);
6        a[i]=x;
7     }
8
9     sum=0;
10    for (i=0; i<10; i++) {
11       sum = sum + a[i];
12    }
13
14    printf("%d", sum);
15 }
```

در برنامه فوق مانند ابتدا عمل خواندن عناصر آرایه از خط ۴ الی ۷ انجام می‌شود. سپس محاسبه مجموع عناصر آرایه از خط ۹ الی ۱۲ انجام شده است.

■

۲.۷ یافتن بزرگترین عنصر

مثال ۷.۵. *برنامه‌ای بنویسید که ده عدد صحیح از ورودی دریافت کند و در یک آرایه ذخیره نماید، سپس بزرگترین عنصر آرایه را یافته و در خروجی نمایش دهد.*

```
1  int a[10];
2  int i,x,max;
3  int main() {
4     for (i=0; i<10; i++) {
5        scanf("%d", &x);
6        a[i]=x;
7     }
8
9     max=a[0];
10    for (i=1; i<10; i++) {
11       if (a[i] > max) {
```

```
12            max=a[i];
13         }
14      }
15
16      printf("%d", max);
17   }
```

برای محاسبه بزرگترین عنصر ابتدا فرض می‌کنیم که اولین عنصر، بزرگترن عنصر باشد (خط ۹)، سپس در یک حلقه تکرار بقیه عناصر را با بزرگترین عنصر مقایسه می‌کنیم. هر عنصری که از بزرگترین عنصر بزرگتر باشد به عنوان بزرگترین عنصر در نظر گرفته می‌شود. در پایان تکرار در خط ۱۶ مقدار بزرگترین عنصر محاسبه شده نمایش داده می‌شود. ■

۳.۷ ارسال آرایه به تابع

در بیشتر موارد لازم است پردازش آرایه از طریق یک تابع انجام شود لذا بایستی آرایه را به تابع ارسال نماییم. از طرفی در زبان C آدرس شروع آرایه به تابع منتقل می‌شود و همچنین تابع تعریف شده از تعداد عناصر آرایه باخبر نیست بنابراین تعداد عناصر آرایه نیز بایستی به تابع اعلام شود.

مثال ۷.۶. *برنامه‌ای بنویسید که ده عدد صحیح از ورودی دریافت کند و در یک آرایه ذخیره نماید، سپس به کمک یک تابع بزرگترین عنصر آرایه را یافته و در خروجی نمایش دهد.*

```
1  int maximum(int z[], int n) {
2     int max, i;
3     max=z[0];
4     for (i=1; i<10; i++) {
5        if (z[i] > max) {
6           max=z[i];
7        }
8     }
9     return max;
10 }
11
12 int main() {
13    int a[10];
14    int i,x,m;
15
16    for (i=0; i<10; i++) {
17       scanf("%d", &x);
18       a[i]=x;
19    }
20
```

```
21      m=maximum(a, 10);
22
23      printf("%d", m);
24  }
```

■

۴.۷ جستجو در آرایه

هدف از جستجویافتن یک کلید مانند x در یک لیست از اطلاعات است که در آرایه ذخیره شده است. این لیست ممکن است مرتب و یا نامرتب باشد. دو روش برای جستجو در آرایه وجود دارد که عبارتند از: جستجوی خطی و جستجوی دودویی. در صورتی که لیست داده شده مرتب باشد ما مجبوریم از روش جستجوی خطی استفاده نماییم و اگر لیست ورودی مرتب شده باشد استفاده از روش جستجوی دودویی بسیار کارآمد خواهد بود. در ادامه هریک از این روشهای جستجو معرفی شده‌اند.

۱.۴.۷ جستجوی خطی

الگوریتم جستجوی خطی یک کلید مانند x را در یک آرایه نامرتب جستجو می‌کند. چون آرایه نامرتب است مجبوریم از اولین عنصر آرایه عمل جستجو را شروع کنیم و با کلید x مقایسه نماییم. در صورتی که به کلید مورد نظر برسیم الگوریتم تمام شده و اندیس یافت شده به عنوان خروجی الگوریتم است. در غیر اینصورت بایستی به ترتیب خانه‌های بعدی آرایه را بررسی نماییم تا به عنصر مورد نظر برسیم. اگر پس از پایان مراحل جستجو کلید مورد نظر یافت نشده باشد خروجی الگوریتم با مقدار منفی یک مشخص می‌شود که نشان می‌دهد کلید در آرایه وجود ندارد. در قطعه برنامه زیر تابع جستجوی خطی به نام linearSearch پیاده سازی شده است.

```
1  void linearSearch(int a[], int n, int x) {
2     int i;
3     for(i=0;i<n; i++) {
4        if (a[i]==x) return i;
5     }
6     return -1;
7  }
```

۲.۴.۷ جستجوی دودویی

```
1  void binarySearch(int a[], int n, int x) {
2     int l,u,m;
3     l=0; u=n-1;
4     while(l<=u) {
```

```
5       m=(l+u)/2;
6       if(x==a[m]) return m;
7       if (x<a[m]) u=m-1;
8       else l=m+1;
9     }
10    return -1;
11 }
```

۵.۷ مرتب‌سازی آرایه

TODO

۱.۵.۷ مرتب سازی درجی

```
1  void insertionSort(int a[], int n) {
2    int i,j,t;
3   for(i=1;i<n; i++) {
4     t=a[i];
5     j=i-1;
6     while (j>=0 && a[j]>t) {
7       a[j+1]=a[j];
8       j--;
9     }
10    a[j+1]=t;
11  }
12 }
```

۲.۵.۷ مرتب سازی انتخابی

```
1  #define SWAP(a,b,x) {x=a; a=b; b=x;}
2
3  void selectionSort(int a[], int n) {
4    int i,j,t,min;
5    for(i=0;i<n;i++) {
6      min=i;
7      for(j=i+1;j<n;j++)
8          if (a[j]<a[min]) min=j;
9      if(i!=min) SWAP(a[i],a[min],t);
10   }
11 }
```

۳.۵.۷ مرتب سازی حبابی

TODO

۴.۵.۷ مرتب‌سازی پایدار

یک الگوریتم مرتب‌سازی پایدار است اگر ترتیب عناصر با کلیدهای برابر در حالت مرتب شده همان ترتیبی باشد که در حالت نامرتب بود. مثلا اگر در یک لیست نامرتب دو عنصر aو b دارای کلیدهای یکسان باشند و در حالت نامرتب a قبل از b باشد، آنگاه الگوریتم مرتب سازی پایدار است اگر در حالت مرتب شده هم a قبل از b باشد.
به طور مثال فرض کنید بخواهیم لیست زیر را بر اساس کلیدهای عددی مرتب نماییم. به همراه هر کلید یک داده الفبایی همراه شده است. همانطور که مشخص است، مقدار کلید ۳ تکرار شده است یک بار به همراه c و یکبار به همراه f.

```
1 4 3 2 8 3 9
a b c d e f g
```

اگر مرتب سازی پایدار باشد نتیجه مرتب شده به صورت زیر خواهد بود. در اینصورت ترتیب داده‌های همراه همانند ترتیب در وضعیت ورودی می‌باشد.

```
1 2 3 3 4 8 9
a d c f b e g
```

و اگر مرتب سازی ناپایدار باشد نتیجه مرتب سازی به صورت زیر ممکن است باشد. در اینصورت ترتیب داده‌های همراه همانند ترتیب در وضعیت ورودی نمی‌باشد.

```
1 2 3 3 4 8 9
a d f c b e g
```

هر چند داده‌ها بر اساس کلید در هر دو حالت مرتب شده هستند ولی ترتیب داده‌های همراه کلیدها در دو حالت فوق متفاوت است.

۵.۵.۷ مرتب سازی ادغامی

TODO

۶.۵.۷ مرتب سازی سریع

TODO

۷.۵.۷ مرتب سازی شمارشی

TODO

۸.۵.۷ مرتب سازی مبنایی

TODO

۶.۷ اشاره‌گر

۷.۷ رشته‌ها

۱.۷.۷ طول رشته

```
1  int strlen(const char *s) {
2     char *p=s;
3     while (*p) p++;
4     return p-s;
5  }
```

۲.۷.۷ کپی رشته

```
1  char *strcpy(const char *s, const char *t) {
2    char *p=s;
3    char *q=t;
4    while (*q) {
5     *p=*q;
6     p++; q++;
7    }
8    return s;
9  }
```

۳.۷.۷ الحاق دو رشته

```
1  char *strcat(const char *s, const char *t) {
2    char *p=s;
3    while (*p) p++;  // find end of s
4    strcpy(p,t);     // copy t at the end of s
5    return s;
6  }
```

۴.۷.۷ مقایسه دو رشته

```
1  int strcmp(const char *s, const char *t) {
2    char *p=s;
3    char *q=t;
4    while (*p && *q && *p==*q) {
5     p++; q++;
6    }
7    return *p-*q;
8  }
```

۵.۷.۷ جستجوی زیر رشته

```
1  char *strstr(const char *s, const char *t) {
2    char *p,*q,*z;
```

```
 3    p=s;
 4    while (*p) {
 5       if (*p==*t) {
 6          q=t; z=p;
 7           while(*q && *z && *q==*z) {
 8              q++; z++;
 9           }
10           if (!*q) return p;
11       }
12      p++;
13    }
14    return NULL;
15  }
```

فصل ۸

فایل

اجرای برنامه TODO