برنامه نویسی به زبان C

گردآورنده : محمد امیری

برگرفته از کتاب برنامه نویسی به زبان C

مهندس عین الله جعفرنژاد قمی

فصل اول

مقدمات زبان C

مقدمات زبان C

زبان C در سال ۱۹۷۲ توسط دنیس ریچی طراحی شد. این زبان تکامل‌یافتهٔ زبان BCPL می‌باشد که طراح آن مارتین ریچاردز است. زبان BCPL از زبان B که طراح آن کِن تامپسون می‌باشد، نتیجه شده است. علت نامگذاری C این‌است‌که بعد از B طراحی شد.

کسانی که تا حدودی با زبانهای برنامه‌سازی آشنایی دارند، می‌دانند که زبان دیگری به نام زبان C++ وجود دارد و آن از C ناشی شده است. C++ علاوه بر ویژگیهای C، ویژگیهای جدیدی دارد که در C موجود نیست. در کتاب حاضر، زبان برنامه‌نویسی C مورد بررسی قرار می‌گیرد. در این فصل، بعضی از عناصر زبان C را مورد بحث قرار می‌دهیم. بعضی از ویژگیهای زبان C عبارت‌انداز:

مقدمات زبان C

- زبان C یک زبان میانی است. زبانهای برنامه‌سازی را می‌توان به سه دسته تقسیم کرد: **زبانهای سطح بالا**، **زبانهای میانی**، **زبانهای سطح پایین** (جدول ۱-۱). علت میانی بودن زبان C این‌است‌که، از طرفی همانند زبان سطح پایینی مثل اسمبلی قادر است مستقیماً به حافظه دستیابی داشته باشد و با مفاهیم بیت، بایت و آدرس کار کند و از طرف دیگر، برنامه‌های این زبان، همچون زبانهای سطح بالایی مثل پاسکال، از قابلیت خوانایی بالایی برخوردارند. به عبارت دیگر، دستورالعملهای این زبان، به زبان محاوره‌ای انسان نزدیک است، که این ویژگی، مربوط به زبانهای سطح بالا است.

مقدمات زبان C

■ زبان C ، یک زبان ساخت یافته است. در این زبان با استفاده از حلقه‌های تکراری مثل while ، for و do while می‌توان برنامه‌هایی نوشت که قابلیت خوانایی و درک آنها بالا باشد. بعضی از زبانهای ساخت‌یافته در جدول ۲-۱ آمده‌اند.

جدول ۱-۱ سطوح زبانهای برنامه‌سازی.

زبانهای سطح بالا	زبانهای میانی	زبانهای سطح پایین
پاسکال اِدا ماجولا -۲ کوبول بیسیک	جاوا فورث C ، (C++)	ماکرواسمبلر اسمبلر

مقدمات زبان C

جدول ۲-۱ زبانها از نظر ساخت‌یافتگی.

زبانهای ساخت‌یافته	زبانهای غیرساخت‌یافته
پاسکال اِدا C، (C++) ماجولا -۲ جاوا	فرترن بیسیک کوبول

مقدمات زبان C

جدول ۳-۱ کلمات کلیدی زبان C.

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

مقدمات زبان C

- زبان C ، قابل انعطاف و بسیار قدرتمند است. در این زبان، هیچ محدودیتی برای برنامه‌نویس وجود ندارد. هر آنچه راکه فکر می‌کنید، می‌توانید در این زبان پیاده‌سازی کنید.
- C ، زبان برنامه‌نویسی سیستم است. برنامه‌های سیستم، برنامه‌هایی هستند که امکان بهره‌برداری از سخت‌افزار و سایر نرم‌افزارها را فراهم می‌کنند. بعضی از برنامه‌های سیستم عبارت‌انداز: سیستم عامل، مفسر [1]، کامپایلر، ویراستارها، واژه‌پردازها، مدیریت بانکهای اطلاعاتی و اسمبلر.
- ارتباط تنگاتنگی بین زبان C و اسمبلی وجود دارد و به‌این‌ترتیب می‌توان از تمام قابلیتهای اسمبلی در زبان C استفاده کرد. چگونگی برقراری ارتباط بین این دو زبان، در فصل ۲۱ به طور مفصل مورد بحث قرار می‌گیرد.
- C ، زبان قابل حمل است. معنای قابلیت حمل این‌است‌که برنامه‌هایی که به زبان C ، در یک نوع کامپیوتر (مثل آی.بی.اِم) نوشته شدند، بدون انجام تغییرات یا انجام تغییراتِ اندک، در کامپیوترهای دیگر (مثل VAX و DEC) قابل استفاده‌اند.

مقدمات زبان C

- C، زبان کوچکی است. تعداد **کلمات کلیدی**[۲] این زبان انگشت‌شمار است (۳۰کلمه کلیدی - جدول ۳-۱). تصور نشود که هرچه تعداد کلمات کلیدی زبان بیشتر باشد، آن زبان قدرتمند است. به عنوان مثال، زبان بیسیک در حدود ۱۵۰ کلمه کلیدی دارد ولی قدرت زبان C به مراتب بیشتر از زبان بیسیک است. توجه داشته باشید که بعضی از کامپایلرهای C، علاوه براین ۳۲کلمه کلیدی، کلمات دیگری را به زبان اضافه کرده‌اند (جدول ۴-۱).
- C نسبت به حروف حساس است[۳]. یعنی در این زبان، بین حروف کوچک و بزرگ تفاوت است و تمام کلمات کلیدی این زبان با حروف کوچک نوشته می‌شوند. به عنوان مثال، while یک کلمه کلیدی است ولی WHILE اینطور نیست. توصیه می‌شود که تمام برنامه‌های C با حروف کوچک نوشته شوند.

مقدمات زبان C

- دستورالعملهای برنامهٔ C دارای ویژگیهای زیر هستند:
  1. هر دستور زبان C به ; ختم می‌شود.
  2. حداکثر طول یک دستور، ۲۵۵ کاراکتر است.
  3. هر دستور می‌تواند در یک یا چند سطر ادامه داشته باشد.
  4. در هر سطر می‌توان چند دستور را تایپ کرد (این کار، توصیه نمی‌شود).
  5. توضیحات می‌توانند در بین */و /* قرار گیرند و یا بعد از //ظاهر شوند:

```
/* This is a sample comment */
// This is another sample comment
```

مقدمات زبان C

انواع داده‌ها

هدف از برنامه‌نویسی، ورود داده‌ها به کامپیوتر، پردازش داده‌ها و استخراج نتایج است. لذا، داده‌ها نقش مهمی را در برنامه‌نویسی ایفا می‌کنند. یکی از جنبه‌های زبانهای برنامه‌سازی که باید دقیقاً مورد بررسی قرار گیرد، انواع داده‌هایی است که آن زبان با آنها سروکار دارد. در زبان C، پنج نوع داده وجود دارند که عبارتند از: **char**، **int**، **float**، **double** و **void**. نوع char برای ذخیره داده‌های کاراکتری مثل 'a' ، 'b' ، 'x' به کار می‌رود. نوع int برای ذخیره اعداد صحیح مثل 125 ، 430 ، 1650 به کار می‌رود. نوع float برای ذخیره اعداد اعشاری مثل 15.5 ، 175.5 و 1250.25 به کار می‌رود و نوع double برای ذخیره اعداد اعشاری که بزرگتر از float باشند مورد استفاده واقع می‌شود. نوع void را در جای مناسبی تشریح خواهیم کرد. هر یک از انواع داده‌های char ، int ، float و double مقادیری را می‌پذیرند که ممکن است از پردازنده‌ای (CPU) به پردازنده دیگر متفاوت باشد. به عنوان مثال، طول نوع int در محیطهای ۱۶ بیتی مثل DOS یا ویندوز ۳/۱، شانزده بیت و در محیطهای ۳۲ بیتی مثل ویندوز NT ، سی و دو بیت است. بنابراین، اگر برنامه‌هایی می‌نویسید که باید در محیطهای مختلف اجرا شوند، سعی کنید از کوچکترین مقدار انواع در C استفاده نمایید.

مقدمات زبان C

انواع داده‌ها

جدول ۵-۱ انواع داده‌ها و مقادیر قابل قبول آنها.

نوع	اندازه به بیت	بازه قابل قبول
char	۸	۱۲۷- تا ۱۲۷
unsigned char	۸	۰ تا ۲۵۵
signed char	۸	۱۲۷- تا ۱۲۷
int	۱۶ یا ۳۲	۳۲۷۶۷- تا ۳۲۷۶۷
unsigned int	۱۶ یا ۳۲	۰ تا ۶۵۵۳۵
signed int	۱۶ یا ۳۲	۳۲۷۶۷- تا ۳۲۷۶۷
short int	۱۶	۳۲۷۶۷- تا ۳۲۷۶۷
unsigned short int	۱۶	۰ تا ۶۵۵۳۵
signed short int	۱۶	۳۲۷۶۷- تا ۳۲۷۶۷
long int	۳۲	۲۱۴۷۴۸۳۶۴۷- تا ۲۱۴۷۴۸۳۶۴۷
signed long int	۳۲	۲۱۴۷۴۸۳۶۴۷- تا ۲۱۴۷۴۸۳۶۴۷
unsigned int	۳۲	۰ تا ۴۲۹۴۹۶۷۲۹۵
float	۳۲	۷ رقم دقت (ارقام بعد از اعشار) (تقریباً 10^{-38} تا 10^{38})
double	۶۴	۱۵ رقم دقت (تقریباً 10^{-308} تا 10^{308})
long double	۸۰	۱۹ رقم دقت (تقریباً 10^{-4932} تا 10^{4932})

مقدمات زبان C

انواع داده‌ها

با استفاده از کلماتی مثل signed (با علامت)، unsigned (بدون علامت)، long و short می‌توان انواع جدیدی را ایجاد کرد. کلمات signed ، short ، long و unsigned را می‌توان با انواع int به کار برد. نوع char را می‌توان با signed و unsigned به کار برد. long به همراه double نیز قابل استفاده است. چون داده‌های نوع int با علامت هستند، کاربرد signed با آنها، بی‌مورد است. انواع مختلف داده‌ها و مقادیری که هر یک از انواع پشتیبانی می‌کنند، در جدول ۵-۱ آمده است.

مقدمات زبان C

متغیرها

متغیر نامی برای کلمات حافظه است که داده‌ها در آنها قرار می‌گیرند و ممکن‌است در طول اجرای برنامه تغییر کنند. برای مراجعه به متغیرها از نامشان استفاده می‌شود. لذا متغیرها امکان نامگذاری برای کلمات حافظه را فراهم می‌کنند. برای نامگذاری متغیرها می‌توان از ترکیبی از حروف a تا z یا A تا Z، ارقام و خط ربط (_) استفاده کرد، به طوری که اولین کاراکتر آنها رقم نباشد. نام متغیر می‌تواند با هر طولی باشد ولی ۳۱کاراکتر اولِ آن مورد استفاده قرار می‌گیرد. بعضی از اسامی مجاز و غیرمجاز برای متغیرها در جدول ۶-۱ آمده‌اند.

تعریف متغیرها

همانطور که گفته شد،متغیرها محل ذخیره داده‌ها هستند و چون داده‌ها دارای نوع‌اند، متغیرها نیز باید دارای نوع باشند. به عبارت دیگر، متغیرهای فاقد نوع، در C شناخته شده نیستند. قبل از به کارگرفتن متغیرها، باید نوع آنها را مشخص کرد. نوع متغیر، مقادیری را که متغیر می‌تواند بپذیرد و اعمالی را که می‌توانند بر روی آن مقادیر انجام شوند، مشخص می‌کند. تعیین نوع متغیر را **تعریف متغیر** گویند. برای تعیین نوع متغیر، به صورت زیر عمل می‌شود:

```
; نام متغیر  نوع داده
```

در این شکل کلی، نوع داده، یکی از انواع موجود در جدول ۵-۱ است. برای تعیینِ نوع بیش از یک متغیر، باید آنها را با کاما از هم جدا کرد.

مثال ۱-۱

تعریف متغیرهای x و y از نوع int، متغیرهای m و n از نوع float، متغیرهای ch1 و ch2 از نوع char، متغیر d1 از نوع double و متغیر p1 از نوع long int.

```
int       x, y ;
float     m, n ;
char      ch1, ch2 ;
double    d1 ;
long int  p1 ;
```

جدول ۶-۱ بعضی از اسامی مجاز و غیرمجاز برای متغیرها.

اسامی مجاز	اسامی غیرمجاز
count	1test
test23	high!there
sum	grade.1
S_1	.pcx

مقداردادن به متغیرها

برای مقداردادن به متغیرها به سه روش می‌توان عمل کرد:

۱. هنگام تعریف (تعیین نوع) متغیر
۲. پس از تعریف نوع متغیر و با دستور انتساب (=)
۳. دستورات ورودی

مثال ۲-۱

مقداردادن به متغیرها در هنگام تعریف آنها.

```
int     x, y = 5 ;
char    ch1 = 'a' , ch2 = 'm';
```

دستور اول، دو متغیر x و y را از نوع int تعریف می‌کند و مقدار متغیر y را برابر با ۵ قرار می‌دهد. دستور دوم متغیرهای ch1 و ch2 را از نوع char تعریف می‌کند، مقدار ch1 را برابر با 'a'، و مقدار ch2 را برابر با 'm' تعیین می‌کند. کاراکترها در داخل کوتیشن یکانی (') قرار می‌گیرند.

مثال ۳-۱

مقدار دادن به متغیرها با دستور انتساب.

```
int       x, y, m ;
float     f1, f2 ;
char      ch1, ch2 ;
f1 = 15.5 ;
f2 = 20.25 ;
x = y = m = 0 ;
ch1 = ch2 = 'a';
```

سه دستور اول، متغیرها را تعریف می‌کنند. دستور چهارم، مقدار 15.5 را در f1 و دستور پنجم مقدار 20.25 را در متغیر f2 قرار می‌دهد. دستور ششم سه متغیر x و y و m را برابر با صفر قرار می‌دهد (انتساب چندگانه در C امکان‌پذیر است). دستور هفتم، حرف 'a' را در متغیرهای ch1 و ch2 قرار می‌دهد.

مثال ۴-۱

مقدار دادن به متغیرها با دستورات ورودی.

```
int     x, y ;
scanf ("%d%d",&x,&y) ;
```

توجه داشته باشید که در اینجا، متغیرهای x و y از نوع int تعریف شدند و دستور scanf مقادیر آنها را از طریق صفحه کلید دریافت می‌کند. فعلاً به چگونگی عملکرد آن کاری نداشته باشید، بلکه فقط این مطلب را یاد بگیرید که، متغیرها با دستورات ورودی، مقدار می‌گیرند.

در مورد تعیین نوع متغیرها این نکته را به خاطر داشته باشید: نوع متغیرها را برحسب نیاز تعریف کنید. به عنوان مثال، اگر تعریف متغیری مثل x از نوع int، جوابگوی نیاز شما است، آن را از نوع float، double یا long int تعریف نکنید.

تعریف ثوابت

ثوابت مقادیری هستند که در برنامه وجود دارند ولی قابل تغییر نیستند. برای تعریف ثوابت به دو روش عمل می‌شود: ۱. استفاده از دستور #define و ۲. استفاده از دستور const. برای تعریف ثوابت از طریق دستور #define به صورت زیر عمل می‌شود:

#define <نام ثابت> <مقدار>

نامگذاری برای ثوابت از قانون نامگذاری برای متغیرها تبعیت می‌کند. مقداری که برای ثابت تعیین می‌شود، نوع ثابت را نیز مشخص می‌کند. دقت داشته باشید که در انتهای دستور #define علامت ; قرار نمی‌گیرد. علتش این‌است‌که این دستور، از دستورات پیش پردازنده [1] است، نه دستور زبان C. **پیش پردازنده** یک برنامه سیستم است که قبل از ترجمهٔ برنامه توسط کامپایلر، تغییراتی در آن ایجاد می‌کند. پیش پردازنده مقدار ثابت را که در دستور #define آمده است، به جای نام ثابت در برنامه قرار می‌دهد و این دستور در زمان اجرا وجود ندارد. نام دیگر ثوابتی که به‌این‌صورت تعریف می‌شوند، ماکرو [2] است که در ادامه کتاب مورد بحث قرار می‌گیرد. برای تفکیک اینگونه ثوابت از متغیرهای برنامه، بهتر است نام آنها با حروف بزرگ انتخاب شود.

مقدمات زبان C

ثوابت

مثال ۵-۱

تعریف ثوابت PI و M با دستور #define . دستور اول، مقدار M را برابر با ۱۰۰ و دستور دوم مقدار PI را برابر با ۳/۱۴ تعیین می‌کند.

```
#define  M   100
#define  PI  3.14
```

برای تعریف ثوابت با دستور const به صورت زیر عمل می‌شود:

const <نوع داده> <نام ثابت> = <مقدار>

در این شکل کلی، نوع داده، یکی از انواع موجود در جدول ۵-۱ است. نام ثابت مثل نام متغیرها انتخاب می‌شود که مقدار ثابت با علامت = در آن قرار می‌گیرد.

مثال ۶-۱

تعیین ثوابت n و count از نوع int و با مقادیر 100 و 50، و ثابت x از نوع signed char و با مقدار 'x'.

```
const   int   n = 100 , int     count = 50 ;
const signed   char      x = 'a' ;
```

اگر پس از تعریف ثوابت، در ادامه برنامه سعی کنید مقادیر آنها را عوض کنید، کامپایلر خطایی را به شما اعلان خواهد کرد. لذا با توجه به تعاریف مثال ۶-۱، دستورات زیر نادرست‌اند:

```
n = 150 ;
x = 'b' ;
count = 200 ;
```

ثوابت

کاربرد ثوابت در برنامه، از امتیازات خاصی برخوردار است. به عنوان مثال، اگر بخواهید در هر بار اجرای برنامه، مقدار n را که در مثال ۶-۱ آمده است عوض کنید، لازم نیست، کلیه دستورات برنامه را که با n سروکار دارند تغییر دهید، بلکه کافی است، مقدار n را در دستور const به مقدار مورد نظر تغییر دهید.

مقدمات زبان C

عملگرها

عملگرها[1] نمادهایی هستند که اعمال خاصی را انجام می‌دهند. به عنوان مثال، نماد '+' عملگری است که دو مقدار را با هم جمع می‌کند (عمل جمع را انجام می‌دهد). پس از تعریف متغیرها و مقداردادن به آنها باید بتوان عملیاتی را روی آنها انجام داد. برای انجام این عملیات باید از عملگرها استفاده کرد. عملگرها در زبان C به چند دسته تقسیم می‌شوند: **۱.** عملگرهای محاسباتی **۲.** عملگرهای رابطه‌ای **۳.** عملگر های منطقی **۴.** عملگرهای بیتی عملگرها بر روی یک یا دو مقدار عمل می‌کنند. مقادیری را که عملگرها بر روی آنها عمل می‌کنند، **عملوند**[2] گویند. به عنوان مثال، در a + 5، متغیر a و مقدار 5 را عملوندهای عملگرِ + گویند.

عملگرهای محاسباتی

عملگرهای محاسباتی، عملگرهایی هستند که اعمال محاسباتی را روی عملوندها انجام می‌دهند. این عملگرها در جدول ۷-۱ مشاهده می‌شوند. هر یک از عملگرهای - ، + ، * ، / تقریباً در همه زبانها وجود دارد. عملگر % برای محاسبه باقیمانده تقسیم به کار می‌رود. این عملگر، عملوند اول را بر عملوند دوم تقسیم می‌کند (تقسیم صحیح) و باقیمانده را برمی‌گرداند. عملگر **کاهش** (--) یک واحد از عملوندش کم می‌کند و نتیجه را در آن عملوند قرار می‌دهد و عملگر **افزایش** (++) یک واحد به عملوندش اضافه کرده نتیجه را در آن عملوند قرار می‌دهد. دستورات زیر را در نظر بگیرید:

```
int  x = 10,  m = 10 ;
x ++ ;
++ m ;
```

متغیرهای x و m با مقدار اولیه ۱۰ تعریف می‌شوند. دستور ; x ++ یک واحد به x اضافه می‌کند و نتیجه را در x قرار می‌دهد و دستور ; m ++ یک واحد به m اضافه کرده نتیجه را در m قرار می‌دهد. بنابراین در این نوع دستورات، عملگر افزایش (یا کاهش) چه قبل از عملوند باشند و چه بعد از آن، عملکرد آنها یکسان است. اما عملکرد آنها در عبارات محاسباتی با هم متفاوت است.

مقدمات زبان C

عملگرها

جدول ۷-۱ عملگرهای محاسباتی.

عملگر	نام	مثال
-	تفریق و منهای یکانی	x - y یا -x
+	جمع	x + y
*	ضرب	x * y
/	تقسیم	x / y
%	باقیمانده تقسیم	x % y
--	کاهش (decrement)	x -- یا -- x
++	افزایش (increment)	x ++ یا ++ x

عبارات محاسباتی

عبارات[1] ترکیبی از متغیرها، ثوابت و عملگرها هستند. به عنوان مثال، x + y، 4 + x / 5، 6 * 2 / m + 4 همگی عبارات محاسباتی‌اند. اگر عملگرهای ++ و -- در عبارات محاسباتی، قبل از عملوند قرار گیرند، ابتدا این عملگرها عمل کرده، نتیجه آن در محاسبات شرکت می‌کند ولی اگر بعد از عملوند ظاهر شوند، مقدار فعلی عملوند مورد استفاده قرار گرفته سپس عملگر بر روی عملوند عمل می‌کند. دستورات زیر را در نظر بگیرید:

```
int   x, y ;
x = 10 ;
y = ++ x ;
```

با اجرای دستور دوم مقدار ۱۰ در x قرار می‌گیرد و با اجرای دستور سوم، ابتدا یک واحد به x اضافه می‌شود و سپس مقدار حاصل در متغیر y قرار می‌گیرد. لذا، y برابر با ۱۱ خواهد شد. اکنون دستورات زیر را در نظر بگیرید:

```
int   x, y ;
x = 10 ;
y = x ++ ;
```

با اجرای دستور دوم، مقدار ۱۰ در x قرار می‌گیرد و با اجرای دستور سوم، مقدار فعلی x در y قرار می‌گیرد و سپس یک واحد به x اضافه می‌شود. لذا، مقدار y برابر با ۱۰ ولی مقدار x برابر با ۱۱ خواهد شد. برای آشنایی بیشتر با این عملگرها، دستورات زیر را در نظر بگیرید:

```
int  x, y, m ;
x = 10 ;
y = 15 ;
m = ++x + y++ ;
```

در اجرای دستور چهارم، ابتدا به مقدار فعلی x (۱۰) یک واحد اضافه می‌شود تا به ۱۱ تبدیل شود. مقدار جدید x با مقدار فعلی y (۱۵) جمع می‌شود و نتیجه آن، یعنی ۲۶ در m قرار می‌گیرد. در پایان، مقدار y برابر با ۱۶ خواهد شد.

تقدم عملگرها

وقتی در عبارتی، چندین عملگر وجود داشته باشند، ترتیب اجرای عملگرها چگونه خواهد بود؟ برای پاسخ به این سئوال، باید **تقدم عملگرها** را مشخص کرد. برای پی بردن به این مسئله، دستورات زیر را در نظر بگیرید:

```
int   m, x = 6 , y = 10 ;
m = x + y / 2 * 3 ;
```

به نظر شما، حاصل این عبارت چه خواهد بود؟ اگر ابتدا x با y جمع شود و حاصل آن بر ۲ تقسیم شود و نتیجه آن در ۳ ضرب شود، حاصل عبارت **۲۴** خواهد بود. اگر ابتدا y بر ۲ تقسیم شود و حاصل آن با x جمع شود و نتیجه آن در ۳ ضرب شود، حاصل آن **۳۳** خواهد بود. اگر ابتدا y بر ۲ تقسیم شود و نتیجه آن در ۳ ضرب شود، حاصل آن با x جمع

شود، حاصل عبارت، ۲۱ خواهد بود. کدام مقدار درست است؟ برای یافتن پاسخ درست، باید تقدم عملگرها را بدانیم. تقدم عملگرهای محاسباتی در جدول ۸-۱ آمده است. همانطور که در این جدول مشاهده می‌کنید، بالاترین تقدم مربوط به عملگرهای افزایش و کاهش و کمترین تقدم مربوط به عملگرهای + و - است. عملگرهایی که تقدم آنها یکسان است، مثل + ، - یا % ، * و / نسبت به هم **تقدم مکانی** دارند. یعنی، هر کدام که زودتر ظاهر شد، همان عملگر زودتر انجام می‌شود.

با توجه به مطالب گفته شده، در عبارت m = x + y / 2 * 3 ابتدا عملگر /و سپس عملگر * و در انتها عملگر + انجام می‌شود. به‌این ترتیب حاصل این عبارت برابر با ۲۱ است.

جدول ۸-۱ تقدم عملگرهای محاسباتی.

بالاترین تقدم	++ --
	-(منهای یکانی)
	* / %
پایین‌ترین تقدم	+ -

عملگرهای رابطه‌ای

عملگرهای رابطه‌ای، ارتباط بین عملوندها را مشخص می‌کنند. اعمالی مثل تساوی دو مقدار، کوچکتر یا بزرگتر بودن، مقایسه با صفر، و غیره، توسط عملگرهای رابطه‌ای مشخص می‌شود. عملگرهای رابطه‌ای در جدول ۹-۱ آمده‌اند. در مورد عملگرهای رابطه‌ای، شاید با عملگر == آشنایی نداشته باشید. این عملگر در دستورات شرطی برای مقایسه دو مقدار مورد استفاده قرار می‌گیرد. به عنوان مثال، دستور مقایسه دو مقدار x و y، باید به صورت آیا x==y است نوشته شود.

جدول ۹-۱ عملگرهای رابطه‌ای.

عملگر	نام	مثال
>	بزرگتر	x > y
>=	بزرگتر یا مساوی	x >= y
<	کوچکتر	x < y
<=	کوچکتر یا مساوی	x <= y
==	متساوی	x == y
!=	نامساوی	x != y

عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل می‌کنند. عبارات منطقی دارای دو ارزش **درستی** و **نادرستی** اند. در زبان C، ارزش **نادرستی** با مقدار صفر و ارزش **درستی** با مقادیر غیر صفر مشخص می‌شود. عملگرهای منطقی در جدول ۱۰-۱ آمده‌اند. ترتیب قرار گرفتن آنها در این جدول از تقدم بالا به پایین است.

نتیجه عملگر ! وقتی درست است که عملوند آن دارای ارزش نادرستی باشد. نتیجه عملگر && وقتی درست است که هر دو عملوند ارزش درستی داشته باشند و نتیجه عملگر || وقتی نادرست است که هر دو عملوند ارزش نادرستی داشته باشند (در بقیه موارد نتیجه آن ارزش درستی دارد). به عنوان مثال، دستورات زیر را در نظر بگیرید:

```
int  x, y, m, p, q ;
x = 0 ;
y = 1 ;
m = x && y ;
p = x || y ;
q = ! x ;
```

با اجرای دستور چهارم، ارزش دارای m برابر با نادرستی خواهد بود، زیرا x دارای ارزش نادرستی و y دارای ارزش درستی است و حاصل بر && بر روی آنها، نادرستی است. با اجرای دستور پنجم، p ارزش درستی خواهد شد، زیرا قرار x دارای ارزش نادرستی است و y دارای ارزش درستی است و عملگر || روی آنها عمل می‌کند و ارزش درستی را برمی‌گرداند. چون x ارزش نادرستی دارد، x! دارای ارزش درستی خواهد بود که نتیجه آن در q قرار می‌گیرد.

جدول ۱۰-۱ عملگرهای منطقی به ترتیب تقدم.

عملگر	نام	مثال
!	نقیض (not)	!x
&&	و (and)	x > y && m <p
||	یا (or)	x > y || m < p

چون اغلب، عملگرهای منطقی و رابطه‌ای با هم مورد استفاده قرار می‌گیرند درک تقدم آنها از اهمیت ویژه‌ای برخوردار است. تقدم آنها را در جدول ۱۱-۱ مشاهده کنید.

عملگرهای ترکیبی

از ترکیب عملگرهای محاسباتی و علامت =، مجموعه دیگری از عملگرها ایجاد می‌شود که عمل محاسباتی و انتساب را انجام دهند. این عملگرها در جدول ۱۲-۱ آمده‌اند. تقدم این عملگرها پایین‌تر از سایر عملگرها است.

جدول ۱۱-۱ تقدم عملگرهای منطقی و رابطه‌ای .

بالاترین	!
	> >= < <=
	== !=
	&&
پایین‌ترین	||

مقدمات زبان C

عملگرها

جدول ۱۲-۱ عملگرهای ترکیبی.

عملگر	نام	مثال	معادل
+=	انتساب جمع	x += y	x = x + y
-=	انتساب تفریق	x -= y	x = x - y
*=	انتساب ضرب	x *= y	x = x * y
/=	انتساب تقسیم	x /= y	x = x / y
%=	انتساب باقیمانده تقسیم	x %= y	x = x % y

مقدمات زبان C

عملگر ()

پرانتزها عملگرهایی هستند که تقدم عملگرهای داخل خود را بالا می‌برند. به عنوان مثال، عبارت زیر را در نظر بگیرید:

```
y = 4 * 2 / (3+1) + (6 + (7-2))
```

برای ارزیابی این عبارت، باید ابتدا عبارت موجود در داخلی‌ترین پرانتز را ارزیابی کرد. در این عبارت، ترتیب انجام عملیات به‌این صورت است : ابتدا عدد ۴ در ۲ ضرب می‌شود که حاصل آن ۸ است. بعد از ۲ ، عملگر تقسیم قرار دارد که عملوند بعدی آن در داخل پرانتز است. لذا تقدم عملگر + موجود در آن، از تقدم عملگر تقسیم بیشتر می‌شود و در نتیجه ۳ با یک جمع شده، حاصل آن ۴ است. اکنون، مقدار ۸ (که قبلاً محاسبه شد) بر ۴ تقسیم می‌شود و حاصل آن ۲ است. عملگر بعدی، + است ولی بعد از آن پرانتز آمده است. چون تقدم عملگرهای موجود در پرانتز بالاتر است، ابتدا ۲ از ۷ کم می‌شود که مقدار آن ۵ است و ۵ با ۶ جمع می‌شود که حاصل آن ۱۱ است و با مقدار ۲ که از قبل محاسبه شد جمع می‌شود و حاصل عبارت ۱۳ است که در y قرار می‌گیرد.

تقدم عملگرها در حالت کلی

جدول ۱۶-۱ تقدم عملگرها در حالت کلی.

0	بالا ترین تقدم
! ~ ++ -- sizeof	
* / %	
+ -	
<< >>	
< <= > >=	
== !=	
&	
^	
|	
&&	
||	
?	
= += -= *= /= %=	پایین‌ترین تقدم

مقدمات زبان C

تبدیل انواع

در زبان C انواع مختلف داده‌ها می‌توانند به یکدیگر تبدیل شوند. تبدیل انواع از دو جهت قابل بررسی است:
۱. تبدیل انواع در عبارات محاسباتی و ۲. تبدیل انواع در احکام انتساب . در مورد تبدیل انواع در عبارات محاسباتی، این قانون حاکم است که انواع کوچکتر به انواع بزرگتر تبدیل می‌شوند:

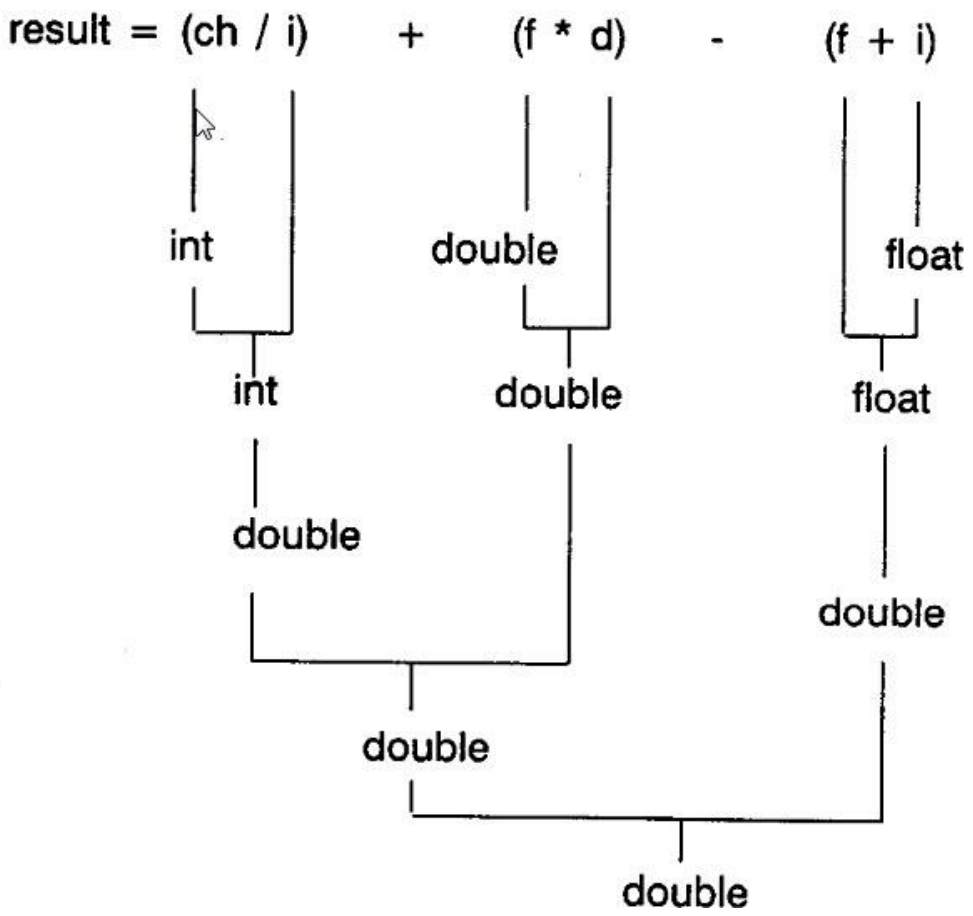
- اگر یکی از عملوندها long double باشد، عملوند دیگر به long double تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها double باشد، عملوند دیگر به double تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها float باشد، عملوند دیگر به float تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها unsigned long باشد، عملوند دیگر به unsigned long تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها long باشد، عملوند دیگر به long تبدیل می‌شود.
- وگرنه، اگر یکی از عملوندها unsigned int باشد، عملوند دیگر به unsigned int تبدیل می‌شود.

تبدیل انواع

توجه داشته باشید که اگر یکی از عملوندها از نوع long و دیگری از نوع unsigned int باشد، ولی مقدار unsigned int نتواند توسط long نمایش داده شود، هر دو عملوند به unsigned long تبدیل می‌شوند. دستورات زیر چگونگی ارزیابی عبارت و تبدیل انواع در عبارات محاسباتی را تشریح می‌کنند. همان‌طور که در ارزیابی این عبارت مشاهده می‌کنید، نوع نتیجه عبارت، double است.

```
char    ch ;
int     i ;
float   f ;
double  d ;
```

تبدیل انواع


result = (ch / i) + (f * d) - (f + i)
int
double
float
int
double
float
double
double
double
double

تبدیل انواع

تبدیل انواع در احکام انتساب وقتی اتفاق می‌افتد که دو متغیر از انواع مختلف به یکدیگر نسبت داده شوند. به عنوان مثال، اگر متغیری از نوع char به متغیری از نوع int نسبت داده شوند، با تبدیل انواع در احکام انتساب سروکار داریم. دستورات زیر را در نظر بگیرید:

```
int x ;
char ch ;
float f ;
...
ch = x ;
x = f ;
f = ch ;
f = x ;
```

همان‌طوری که در این احکام انتساب می‌بینید، انواع کاراکتری، اعشاری و دقت مضاعف به یکدیگر نسبت داده می‌شوند. هر کدام از این انواع به انواع دیگری قابل تبدیل است ولی باید دقت داشته باشید که در تبدیل انواع، بخشی از اطلاعات ممکن است از بین برود. به عنوان مثال، در انتساب یک مقدار اعشاری به یک مقدار صحیح، حداقل مقداری که از بین می‌رود، بخش اعشاری عدد است. تبدیل انواع مختلف به یکدیگر و اطلاعاتی که ممکن است از بین برود، در جدول ۱۷-۱ آمده‌اند.

مقدمات زبان C

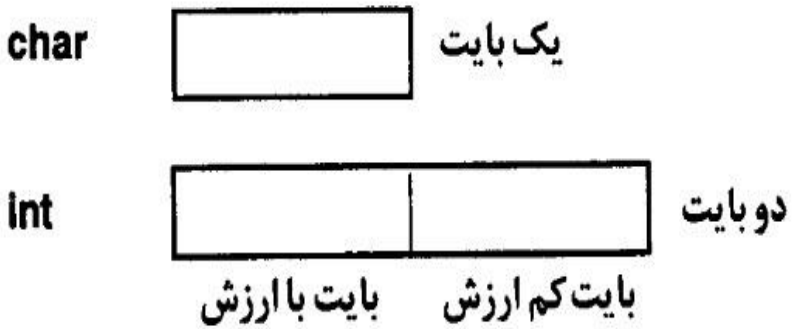
تبدیل انواع

جدول ۱۷-۱ تبدیل انواع در احکام انتساب.

نوع منبع	نوع مقصد	اطلاعاتی که ممکن است از بین برود
char	signed char	اگر مقدار بیش از ۱۲۷ باشد، مقصد منفی خواهد شد
short int	char	۸ بیت با ارزش
int (۱۶ بیتی)	char	۸ بیت با ارزش
int (۳۲ بیتی)	char	۲۴ بیت با ارزش
long int	char	۲۴ بیت با ارزش
int (۱۶ بیتی)	short int	اطلاعات از بین نمی‌رود
int (۳۲ بیتی)	short int	۱۶ بیت با ارزش
long int	int (۱۶ بیتی)	۱۶ بیت با ارزش
long int	int (۳۲ بیتی)	اطلاعات از بین نمی‌رود
float	int	بخش کسری و یا بیشتر
double	float	دقت علائم کم می‌شود و نتیجه گرد می‌شود
long double	double	دقت علائم کم می‌شود و نتیجه گرد می‌شود

تبدیل انواع

همان‌طور که در این جدول مشاهده می‌کنید، تبدیل نوع double به int وجود ندارد. برای چنین تبدیلی، ابتدا باید double را به float و سپس float را به int تبدیل کنید. برای مشاهده چگونگی از بین رفتن اطلاعات در تبدیل انواع، انتساب int به char را در نظر می‌گیریم:



داده‌های کاراکتری در یک بایت و داده‌های صحیح در دو بایت ذخیره می‌شوند. اگر داده‌های ذخیره شده در متغیر صحیح، به اندازه‌ای کوچک باشد که در بایت کم ارزش قرار گیرد، در انتقال مقادیر صحیح به کاراکتری، اطلاعاتی را از دست نخواهیم داد. ولی اگر مقدار موجود در متغیر صحیح، در دو بایت ذخیره شده باشد، بخشی از عدد که در بایت باارزش آن قرار دارد، از بین می‌رود. لذا برنامه‌نویس باید در تبدیل انواع دقت کافی به خرج دهد تا اطلاعات مفید و ارزشمند خود را از دست ندهد.

روش ایجاد برنامه

مقدمات زبان C

روش ایجاد برنامه

تاکنون با مقدمات زبان C آشنا شدید و آماده می‌شوید که در فصل ۲ برنامه‌های ساده ای را بنویسید. اما قبل از آن، خوب است که با روش ایجاد برنامه در C آشناشوید. برنامه نویسی، نوعی حل مسئله است. اگر مسئله‌ها را به‌راحتی حل می‌کنید، برنامه‌نویس موفقی می‌شوید. یکی از اهداف کتاب این‌است‌که توانایی شما را در حل مسئله بالا ببرد. در C می‌توان برنامه‌های **ساخت‌یافته** نوشت. برنامه‌های ساخت‌یافته برنامه‌هایی هستند که قابلیت خوانایی آنها بالا است، درک آنها آسان است، و نگهداری آنها مشکل نیست. برنامه‌نویسی ساخت‌یافته، برنامه‌نویسی استانداردی است. برای نوشتن برنامه در C، باید موارد زیر را در نظر بگیرید:

۱. تعیین نیازمندیهای مسئله
۲. تحلیل مسئله
۳. طراحی الگوریتم حل مسئله
۴. پیاده‌سازی الگوریتم
۵. تست و کنترل برنامه
۶. نگهداری و نوسازی برنامه

تعیین نیازمندیهای مسئله

تعیین نیازمندیهای مسئله موجب می‌شود تا مسئله را به وضوح و بدون هیچ ابهامی بشناسید و چیزهایی را که برای حل مسئله لازم است، درک کنید. هدف این‌است‌که جنبه‌های بی‌اهمیت مسئله را نادیده بگیرید و به موارد اصلی بپردازید. این کار، چندان ساده نیست. ممکن‌است لازم باشد با کسی که مسئله را طرح کرد، مذاکراتی داشته باشید.

تحلیل مسئله

تحلیل مسئله، شامل تعیین ورودیها و خروجیها و سایر نیازمندیها یا محدودیتهای حل مسئله است. ورودیها، داده‌هایی هستند که مسئله باید بر روی آنها کار کند و خروجیها، نتایج مورد انتظارِ مسئله‌اند. در این مرحله، فرمت خروجی اطلاعات باید مشخص شود، متغیرها باید تعریف شوند و رابطه بین آنها باید مشخص گردد. ارتباط بین متغیرها ممکن‌است با فرمول خاصی بیان شود.

اگر مراحل ۱ و ۲ به خوبی انجام نشوند، مسئله به درستی حل نخواهد شد. صورت مسئله را به‌دقت بخوانید تا اولاً ایدهٔ روشنی از مسئله پیدا کنید و ثانیاً ورودیها و خروجیها را تعیین نمایید. ممکن‌است با خواندن دقیقِ صورت مسئله، به این مطلب پی ببرید که بسیاری از ورودیها و خروجیها، در صورتِ مسئله وجود دارند. جملات زیر را که مسئله‌ای را بیان می‌کنند در نظر بگیرید:

دانشجویی می‌خواهد با دانستن **طول و عرض** مستطیل، **مساحت** آن را حساب کند.

ورودی‌ها و خروجی‌های مسئله با استفاده از کلماتی که پررنگ شده‌اند مشخص می‌گردد:

ورودی‌های مسئله

- طول مستطیل.
- عرض مستطیل.

خروجی مسئله

- مساحت مستطیل.

وقتی ورودی‌ها و خروجی‌های مسئله مشخص شد، باید فرمول‌هایی تدوین شود که رابطه بین آن‌ها را مشخص کند.
فرمول محاسبه مساحت مستطیل به صورت زیر است:

عرض مستطیل × طول مستطیل = مساحت مستطیل

در بعضی از موارد ممکن‌است رابطه بین ورودیها و خروجیها به این سادگی مشخص نشود و نیاز به فرضها و تسهیلات خاصی باشد. فرایند استخراج متغیرهای مسئله و تعیین روابط بین آنها از طریق صورت مسئله، **انتزاع** [1] نام دارد.

طراحی الگوریتم

در طراحی الگوریتم برای حل مسئله، لازم است قدم‌به‌قدم **رویه‌هایی**[۲] نوشته شوند-**الگوریتم**- و سپس بررسی شود که آیا الگوریتم، مسئله را به درستی حل می‌کند یا خیر. نوشتن الگوریتم، مشکل‌ترین بخش حل مسئله است. سعی نکنید تمام جزئیات مسئله را حل کنید، بلکه سعی کنید شیوه طراحی **بالا به پایین**[۳] را به کار ببرید. در روش طراحی بالا به پایین، ابتدا مراحل اصلی مسئله که باید حل شوند، مشخص می‌گردند و سپس با حل هر مرحله اصلی، کل مسئله حل می‌شود. اغلب الگوریتمها معمولاً مراحل زیر را دارا هستند:

۱. خواندن داده‌ها ۲. انجام محاسبات ۳. چاپ نتایج

وقتی مراحل مهم مسئله مشخص شدند، می‌توانید هر مرحله را به‌طور جداگانه حل کنید. به عنوان مثال، مرحله انجام محاسبات ممکن‌است به بخشهای کوچکتری تقسیم شود. این عمل را **بهینه‌سازی الگوریتم** گویند.

پیاده‌سازی الگوریتم

در پیاده‌سازی الگوریتم، باید الگوریتم را به برنامه تبدیل کرد. هر مرحله از الگوریتم باید به یک یا چند دستور زبان برنامه‌سازی تبدیل شود. یکی از کارهای مهم این مرحله مشخص کردن فایل‌های سرآیند[۴] است که باید به برنامه اضافه شوند. با این فایل‌ها در فصل ۲ آشنا می‌شوید.

تست برنامه

در تست و بررسی برنامه، باید کل برنامه را تست کنید تا مشخص شود که آیا خواسته شما را برآورده می‌کند یا خیر. در تست برنامه، باید آن را برای داده‌های مختلفی چند بار اجرا کنید و خروجی‌های برنامه را بررسی کنید.

نگهداری برنامه

نگهداری و **نوسازی** [1] برنامه شامل اصلاح برنامه جهت حذف خطاهای قبلی و نوسازی آن جهت پاسخگویی به نیازهای فعلی است. بعضی از سازمانها بعد از اینکه نویسنده برنامه‌ای به جایی دیگر منتقل شد، برنامه‌های آن را تا ۵ سال یا بیشتر نگهداری می‌کنند، ولی به تدریج آن را از دور خارج می‌کنند.

فرآیند آماده‌سازی و اجرای برنامه

پس از اینکه برنامه‌تان را نوشتید، باید آن را وارد کامپیوتر کرده، ترجمه، پیوند و اجرا نمایید. لذا فرآیند آماده‌سازی و اجرای برنامه شامل موارد زیر است:

۱. واردکردن برنامه در یک محیط ویراستاری و ذخیره کردن بر روی دیسک. فایل برنامه بر روی دیسک، در کامپایلرهای C++ با پسوند cpp. و در کامپایلرهای توربو C با پسوند c. ذخیره می‌شود.

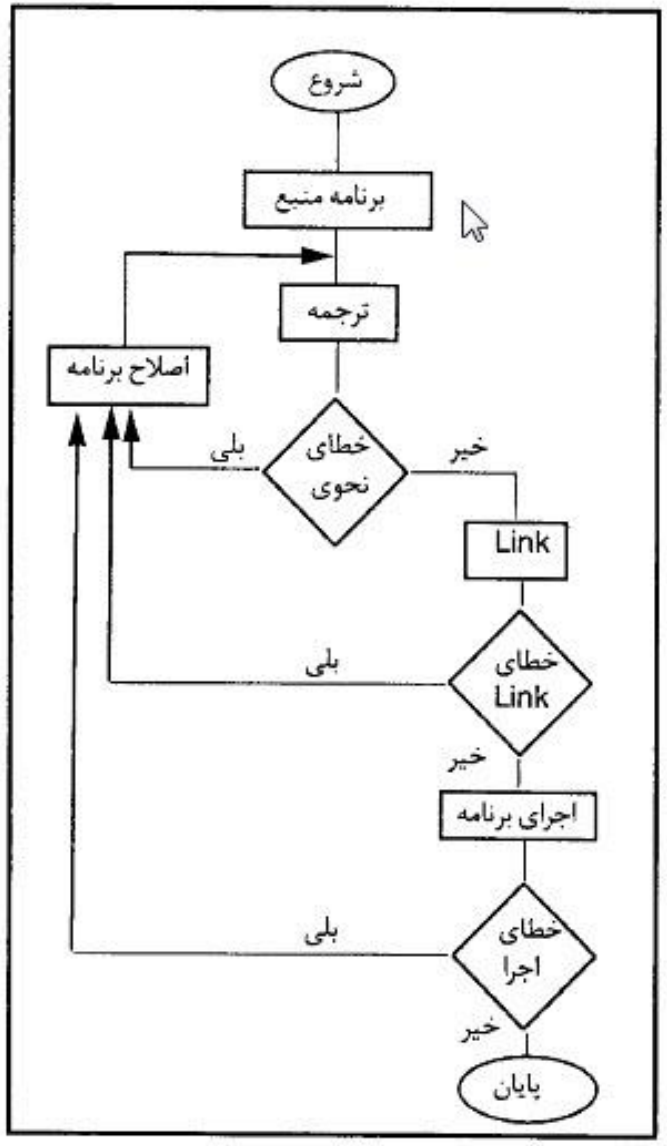
۲. ترجمه برنامه جهت اشکالزدایی آن. اگر برنامه **اشکالات نحوی**[۲] داشته باشد، باید آنها را برطرف کرده و برنامه را دوباره ترجمه کنید. این مرحله را آنقدر انجام دهید تا کلیه اشکالات برنامه برطرف شوند.

۳. پس از ترجمه برنامه، فایلی با پسوند obj ایجاد می‌گردد که به زبان ماشین است ولی قابل اجرا نیست. علتش این است که هنوز بخش‌های مختلف برنامه به هم پیوند نخورده‌اند و آدرس‌دهی آنها کامل نیست.

۴. پس از ترجمه برنامه، باید عمل **پیوند**[۱] را انجام دهید. در این مرحله، فایلی با پسوند exe تشکیل می‌شود که قابل اجرا است.

مقدمات زبان C

روش ایجاد برنامه



شکل ۱-۱ روند آماده‌سازی و اجرای برنامه

تمرینات

مقدمات زبان C

تمرینات

۱. ویژگیهای مهم زبان C را توصیف کنید.
۲. ضرورت وجود توضیحات را در برنامه بیان کرده، روش اعلان توضیحات را در C تشریح کنید.
۳. هدف از انواع داده چیست؟ انواع داده‌ها را در C تشریح کنید.
۴. تعریف متغیرها و چگونگی مقداردهی به آنها را بیان کنید.
۵. ثوابت را تعریف کرده، چگونگی تعیین آنها را در C بیان کنید. به نظر شما استفاده از ثوابت در برنامه چه امتیازاتی دارد؟
۶. دستوراتی بنویسید که متغیرهای x و y را از نوع int، ch و d را از نوع double و ثابت k را از نوع unsigned int و با مقدار اولیه ۲۰ تعریف کند.
۷. منظور از عملگر چیست؟ به نظر شما کاربرد عملگرهای ++ و -- چه امتیازاتی دارد؟
۸. منظور از تقدم عملگرها چیست؟ مثال بزنید.
۹. مثالی بزنید که در آن از عملگرهای بیتی استفاده شده باشد؟
۱۰. نقش عملگر ? را بیان کنید.
۱۱. مثالهایی از تبدیل انواع در احکام انتساب و تبدیل انواع در عبارات محاسباتی ارائه دهید.
۱۲. قوانین حاکم بر تبدیل انواع در عبارات محاسباتی را بیان کنید.
۱۳. تشریح کنید که در چه مواقعی ممکن است در تبدیل انواع در احکام انتساب، اطلاعات از بین برود؟

مقدمات زبان C

تمرینات

۱۴. فرمولهای زیر را به صورت عباراتی در C بنویسید.

1. $x = \frac{y * 2}{m + p}$

2. $y = x + m^2 - \frac{k}{r + 2}$

۱۵. با توجه به مقادیر تعیین شده، هر یک از عبارات زیر را ارزیابی کنید:

```
x = 8   ,   y = 10   ,   m = 6
k = x / 4 * (y / 2) * m
k = x / y ++ + y / -- m
```

۱۶. با توجه به مقادیر زیر مقدار y چند است؟

```
x = 8 ,  m = 6
y = x * 2 < m + 4   ?   4 * m : 8 *  m
```

۱۷. مراحل ایجاد برنامه را تشریح کنید.

۱۸. برای مسئلهٔ زیر مراحل ایجاد برنامه را طی کنید:

دانش آموزی می‌خواهد **معدل ۵ نمره** خود را محاسبه کند.