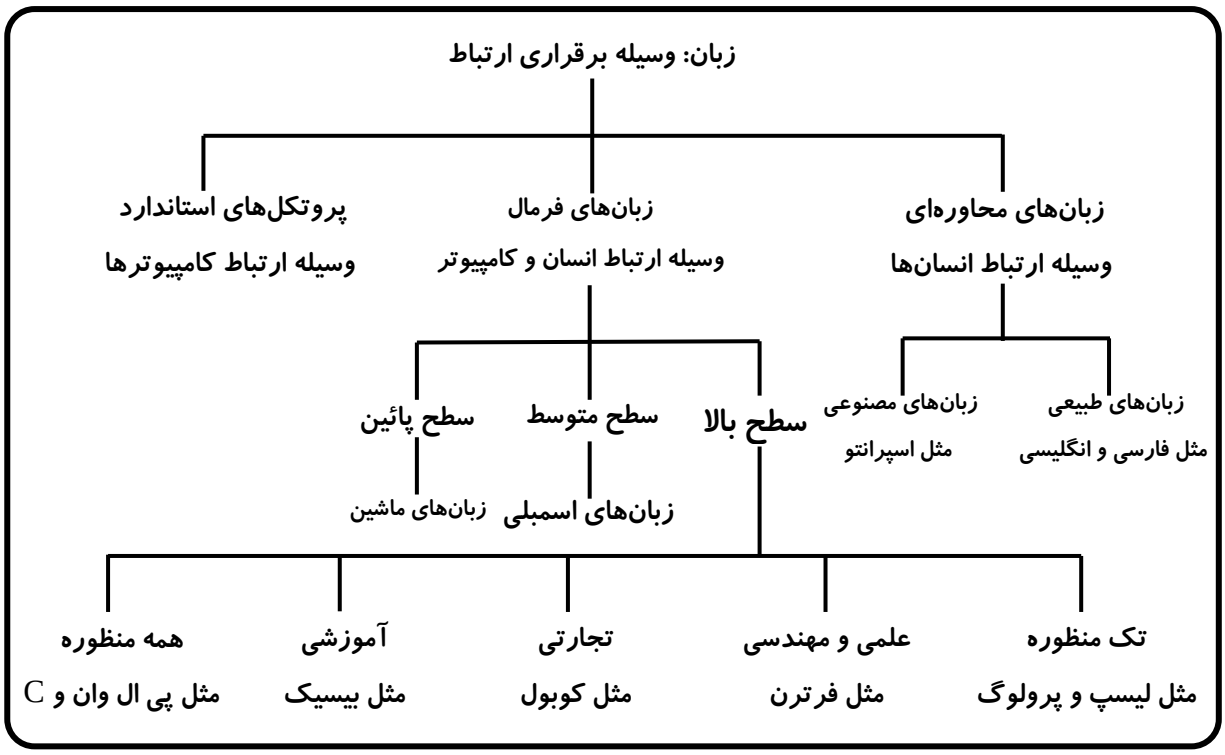
مبانی کامپیوتر و برنامه‌نویسی به زبان C

فصل سوم: تاریخچه و مقدمات زبان C

۳-۱ مقدمه



شکل ۳-۱: طبقه بندی زبان‌ها به عنوان وسیله ارتباط.

۳-۲ زبان‌های فرمال

زبان ماشین: قابل فهم برای کامپیوتر به عنوان مجری، عدم وجود ابهام در آن.

مشکل زبان ماشین: کلاً از صفر و یک، یادگیری آن مشکل، امکان بروز اشتباه خیلی زیاد، اِعمال اصلاحات در آنها کاری طاقت‌فرسا، نیاز به تخصص کامپیوتر.

زبان‌های اسمبلی: دستورالعمل‌های نمادی (تشکیل شده از حروف) و آدرس‌های نمادی، وابسته به ماشین .

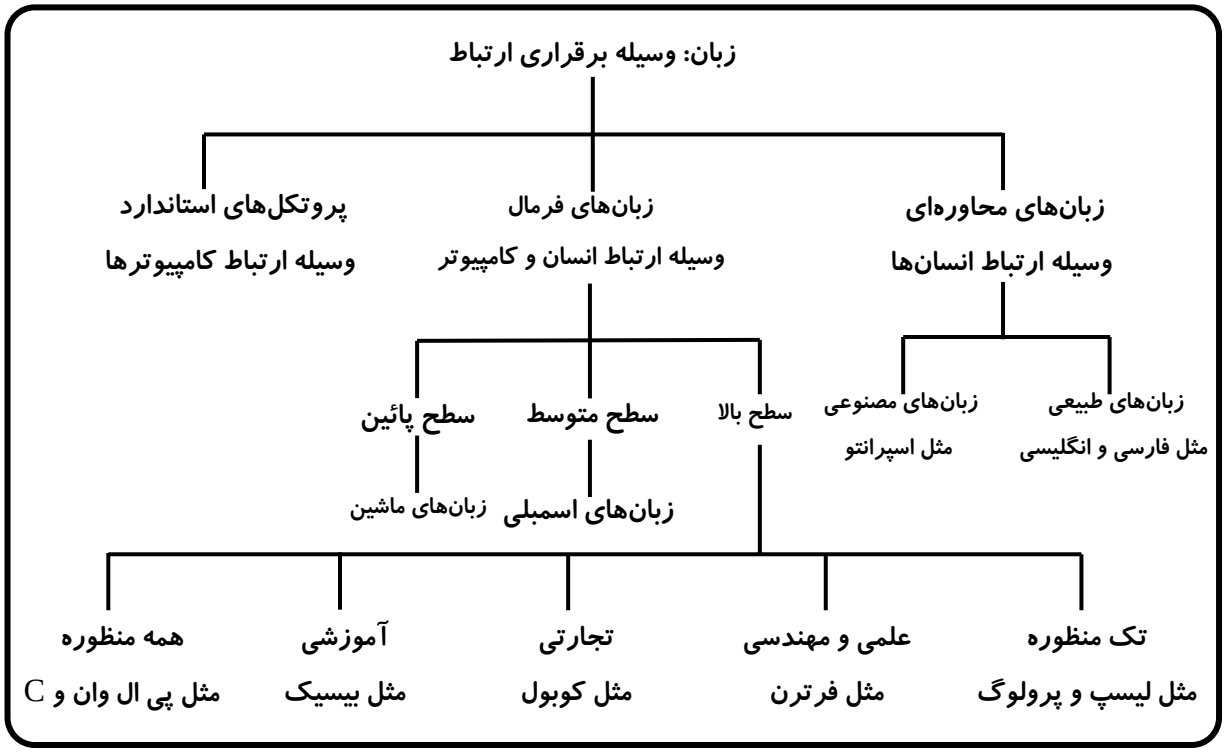
جدول ۱-۳: نمونه یک برنامه به زبان ماشین فرضی و معادل آن به یک زبان اسمبلی فرضی.

آدرس حافظه	محتویات (دستورالعمل‌ها یا داده‌ها)	معادل هرسطر به زبان اسمبلی		
۰۰۰۰۰		M	RES	
۰۰۰۰۱		N	RES	
۰۰۰۱۰		S	RES	
۰۰۰۱۱	۰۰۱۰۰۰۰۰		READ	M
۰۰۱۰۰	۰۰۱۰۰۰۰۱		READ	N
۰۰۱۰۱	۰۱۱۰۰۰۰۰		LOAD	M
۰۰۱۱۰	۱۰۰۰۰۰۰۱		ADD	N
۰۰۱۱۱	۱۰۱۰۰۰۱۰		STORE	S
۰۱۰۰۰	۰۱۰۰۰۰۰۰		PRINT	M
۰۱۰۰۱	۰۱۰۰۰۰۰۱		PRINT	N
۰۱۰۱۰	۰۱۰۰۰۰۱۰		PRINT	S
۰۱۰۱۱	۰۰۰۰۰۰۰۰		STOP	

۳-۲-۱ زبان‌های برنامه‌نویسی سطح بالا

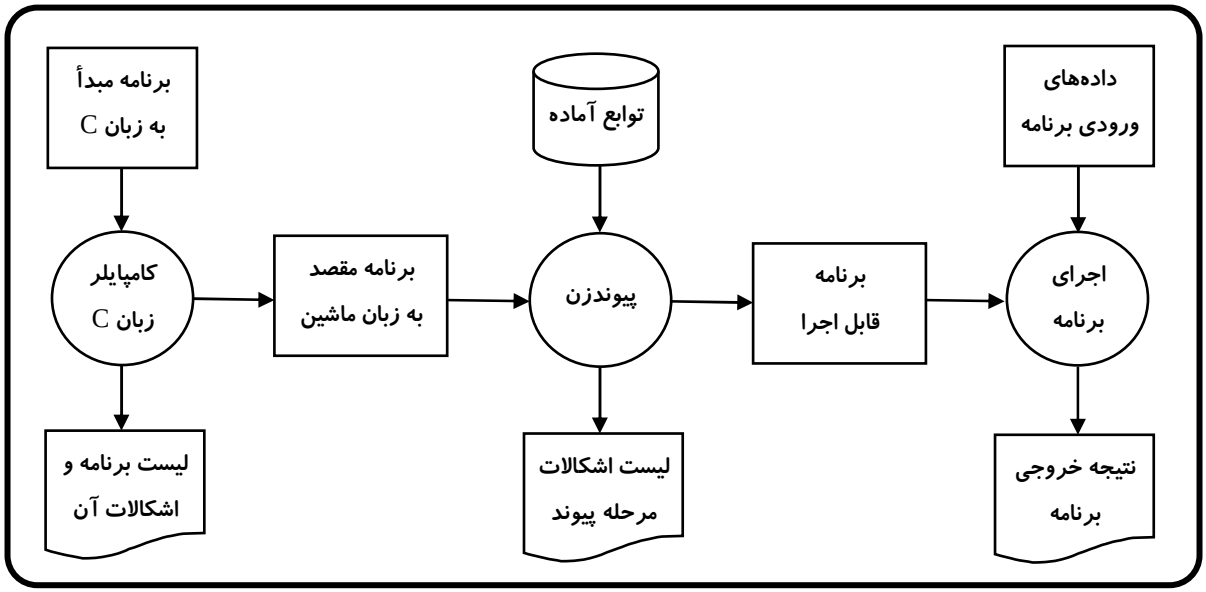
دستورالعمل‌ها تا حدی شبیه جملات زبان انگلیسی، حداقل وابستگی به ماشین، کاربرد ساده.

طبقه بندی زبانهای سطح بالا



شکل ۳-۱: طبقه بندی زبان‌ها به عنوان وسیله ارتباط.

۳-۲-۲ نحوه اجرای یک برنامه در زبان سطح بالا



شکل ۲-۳: مراحل مختلفی که برای اجرای یک برنامه به زبان C باید طی گردد.

۳-۳ تاریخچه زبان C

۳-۳-۱ سیر تکاملی

اول BCPL

بعد B

سپس NewB

و نهایتاً C

بعد C++، Java و C#

- **۱۹۸۳: ماموریت ANSIبرای کمیته X۳J۱۱، بررسی و ارائه تعریف فاقد ابهام و مستقل از ماشین زبان C.**
- **۳۱اکتبر ۱۹۸۸: پیشنهاد کمیته تحت عنـــوان**

American National Standard for Information Systems – Programming Laguage C, X۳.۱۵۹-۱۹۸۹

موسوم به ANSI C یا C استاندارد.

- **۱۹۹۰: پذیرفتن استاندارد ANSI/ISO C یا ANSI C یا C استاندارد توسط کمیته بین‌المللی استاندارد.**
- **۱۹۹۸ تا ۲۰۰۵: استفاده از استاندارد مزبور در کتاب حاضر به عنوان مبنای تعریف زبان C.**
- **ظهور روش‌های شئ‌گرا و زبان جدیدی بنام C++**
- **ارائه زبان جاوا برای برنامه‌نویسی اینترنت مستقل از ماشین و سیستم عامل.**
- **سهولت یادگیری هر دو زبان C++ و جاوا برای یک برنامه‌نویس C.**

۳-۳-۲ دلایل اهمیت زبان C

زبان منتخب: برای ایجاد مهمترین نرم‌افزارها روی مجموعه وسیعی از کامپیوترها.

واژه‌پردازها، نرم‌افزارهای عملیات بانکی، نرم‌افزارهای صفحه گسترده، آماری، سیستم‌های ارتباطی، شبکه‌ای و گرافیکی، کامپایلرها، نرم‌افزارهای آموزشی، برنامه‌ریزی و شبیه‌سازی، بخش‌های عمده سیستم‌عامل ، سیستم عامل ساده و محدود MS-DOS، سیستم عامل OS/۲، سیستم عامل مستقل از کامپیوتر Windows NT، نرم‌افزار مدیریت پایگاه داده‌های فاکس‌پرو . . . .

قابلیت حمل: امکان انتقال بدون تغییر برنامه C از یک کامپیوتر شخصی به یک کامپیوتر بزرگ و اجرا.

کارآیی: زبان اسمبلی قابل حمل!، ترجمه تولید شده توسط کامپایلر C هم کوچک از نظر اندازه، دارای سرعت پردازش زیاد در هنگام اجرا.

قدرت بیان: انواع ساختمان داده‌ها (آرایه‌ها و رکوردها)، نوع های داده اختصاصی و عملگرهای جدید.

تنوع در اندازه داده‌های محاسباتی: نوع داده‌های صحیح و اعشاری، وجود اندازه‌های مختلف و امکان حذف یا حفظ علامت، کنترل وسیع روی مقادیر عددی.

سرعت بالا: انجام عملیات با سرعت حداکثر در درجه اول برای نوشتن سیستم‌های عامل، کامپایلرها و سایر نرم‌افزارهای پایه‌ای، کنترل‌ها حتی‌الامکان در زمان ترجمه.

واحدمند بودن: تجزیه برنامه به تعدادی توابع مجزا، تشویق کار تیمی در پروژه‌های بزرگ.

شکل ظاهری ساده و قابلیت فهم آسان: عدم وجود قواعد پیچیده در نحوه نوشتن جملات، بیان همه عملیات در قالب توابع برمبنای تعریف ریاضی آنها.

کتابخانه‌های متنوع برای کابردهای گوناگون: فراهم‌کردن مجموعه(کتابخانه)های مختلف شامل توابع آماده برای عملیات مورد نیاز، قابل احضار در برنامه ها.

پشتیبانی‌های برنامه‌نویسی: اجرا بر روی انواع ماشین‌ها، امکانات متنوع نوشتن، آزمودن، خطایابی و اشکال‌زدایی برنامه‌ها، ویرایشگرهای واجد امکانات بررسی ساختارهای این زبان، کنگره‌بندی اتوماتیک و حتی خطایابی نحوی، اشکال‌زدای اتوماتیک.

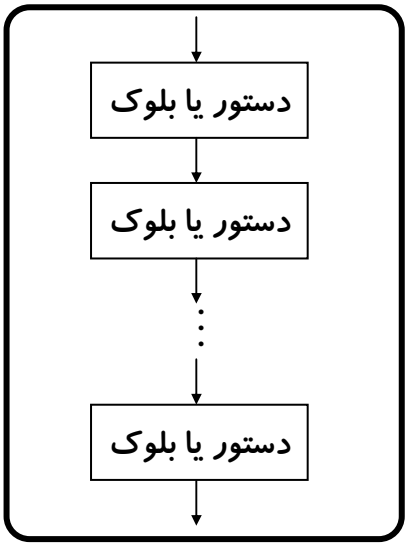
۳-۴ روش‌های برنامه‌نویسی

روشهای سنتی: کاسه اسپاگتی

۳-۴-۱ برنامه‌نویسی ساختیافته

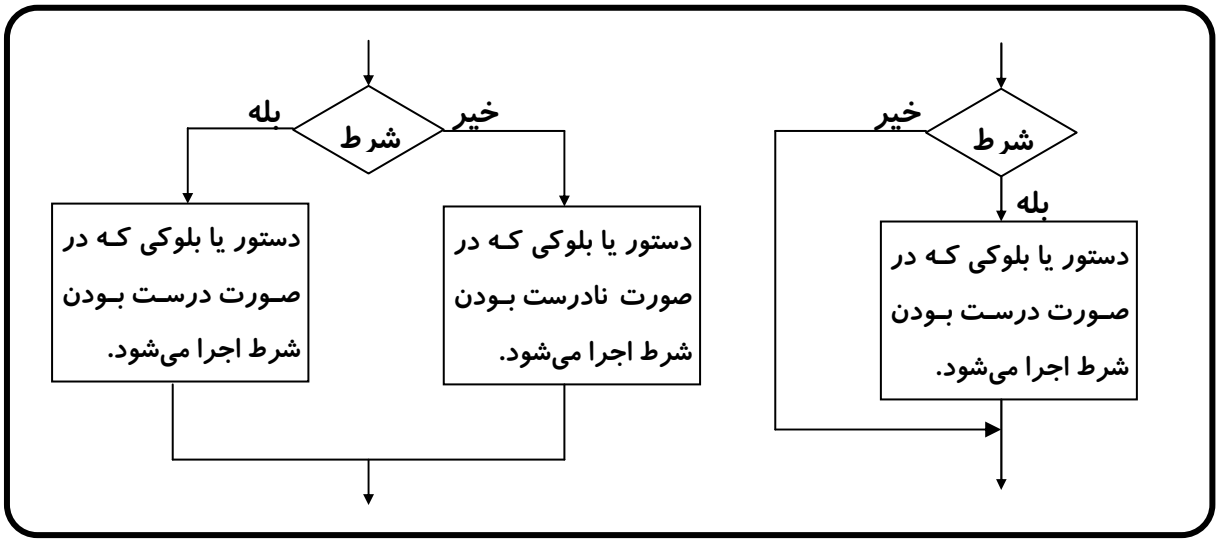
- ساختار ترتیب (مجموعه‌ای از دستورهای متوالی).
- ساختار تکرار (انواع حلقه‌های تکرار).
- ساختار انتخاب (تصمیم‌گیری در قالب جملات شرطی).
- یک برنامه ساختیافته: مجموعه‌ای از بلوک‌ها، شروع اجرا از اولین بلوک، خاتمه اجرا درآخرین بلوک.
- یک بلوک: یک دستور ساده یا مجموعه‌ای از دستورها در قالب یکی از ساختارهای بالا.

ساختار ترتیب:



شکل ۳-۳: قالب کلی ساختار ترتیب.

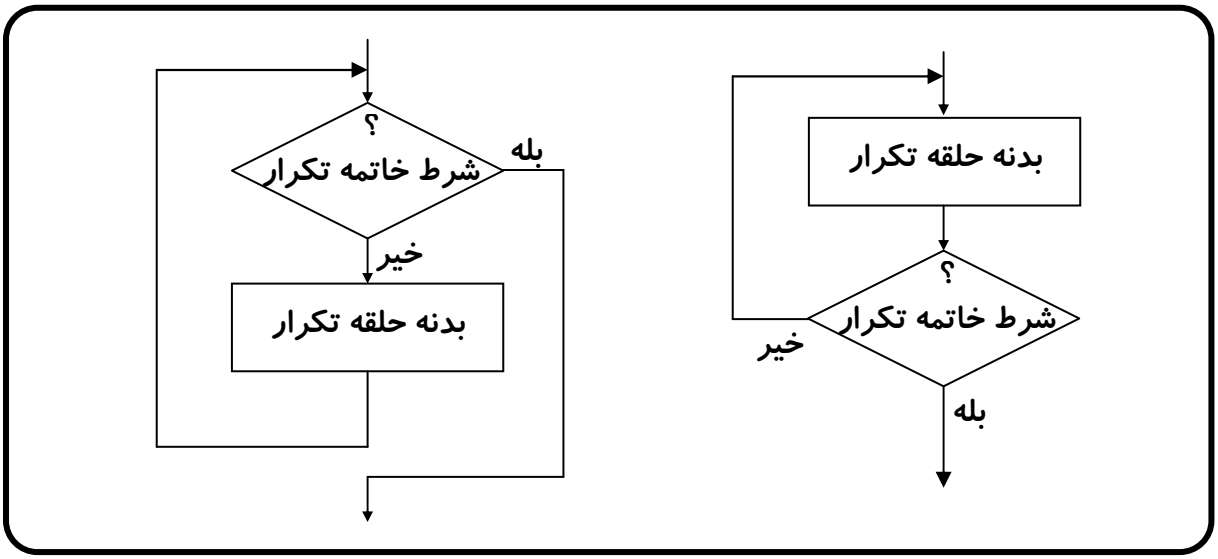
ساختار انتخاب یا تصمیم گیری:



شکل ۳-۴: قالب کلی دو حالت ساختار انتخاب یا تصمیم‌گیری.

ساختار تکرار: وجود در تقریباً همه برنامه‌ها.

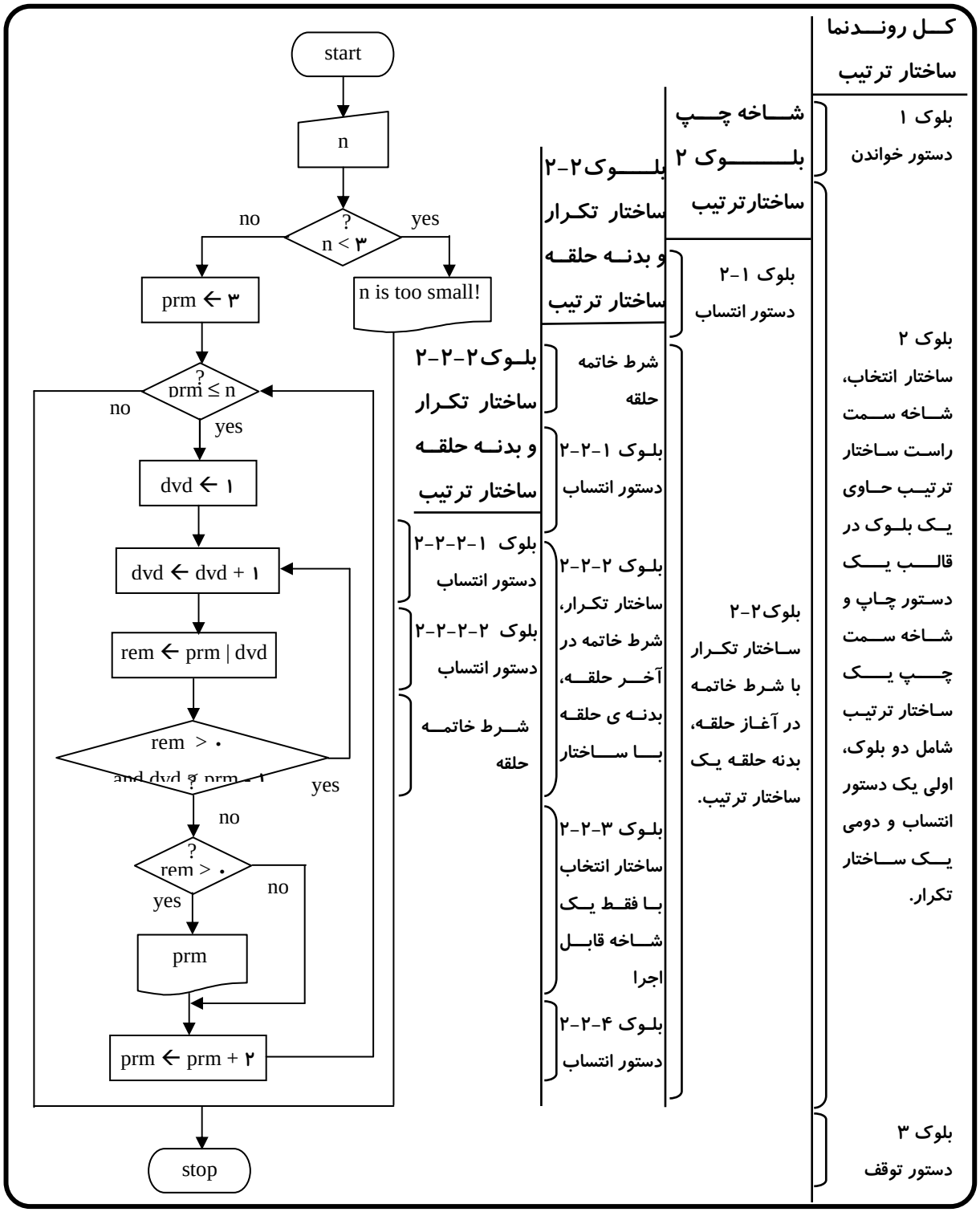
حلقه‌های تکرار با شمارنده، حلقه‌های تکرار با شرط خاص برای خاتمه و یا ترکیبی از آنها.



شکل ۳-۵: قالب کلی دو حالت ساختار تکرار.

نمونه عملی روندنمای ساختیافته: شکل ۳-۶

نوشتن برنامه به صورت ساختیافته: لزوم رعایت اصول مربوط به این روش از مرحله رسم روندنما



شکل ۳-۶: روندنمای چاپ اعداد اول از ۳ تا n به صورت ساختیافته همراه با تقسیم بندی ساختاری آن.

۳-۴-۲ برنامه‌نویسی واحدمند یا پیمانه‌ای

اصول ساختیافتگی: در متن برنامه‌ها، ساختار یکنواخت و به راحتی قابل فهم و دنبال‌کردن.

اصول واحدمندی: یک سطح بالاتر، مجموعه برنامه‌ها به صورت واحدهای مستقل.

یک واحد یا پیمانه: هر تابع یا برنامه مجزا در این روش.

- **هر واحد (تابع یا برنامه) دارای اندازه معقول ترجیحا قابل چاپ در یک صفحه A۴.**
- **وظیفه هر برنامه دقیقاً مشخص، عدم انجام کارهایی که به هم ارتباطی ندارند در یک برنامه.**
- **ارتباط درست بین برنامه‌ها و سهولت حذف و اضافه برنامه‌ها بدون آثار جانبی ناخواسته روی سایر برنامه‌ها**

۳-۴-۳ برنامه‌نویسی یک خطی

زبان اِی پی ال، وجود تعداد بسیار زیادی از عملگرها.

امکان نوشتن هر برنامه در یک سطر!

۳-۵ مفاهیم اولیه زبان C

۳-۵-۱ یک برنامه نمونه

```
#include <stdio.h>                    /* افزودن تعریف توابع مربوط به ورودی و خروجی به برنامه */
                                      /* برنامه‌ای برای خواندن n از ورودی و چاپ اعداد اول از ۳ تا n */
main()                                /* اعلام اسم برنامه */
{                                     /* علامت شروع برنامه */
   int prm, dvd, n, rem;              /* اعلام متغیرهای مورد استفاده */

   scanf("%d", &n);                   /* خواندن n از ورودی */
   if (n < ۳)                         /* تست درست بودن مقدار n */
      printf("n is too small!\n");    /* چاپ پیغام غلط بودن مقدار n */
   else
      for (prm = ۳; prm <= n; prm = prm + ۲)  /* حلقه تکرار ساختن اعداد */
      {                               /* شروع بدنه حلقه بیرونی */
         dvd = ۱;                     /* تعیین مقدار اولیه برای مقسوم‌علیه */
         do                           /* حلقه تکرار بررسی اول بودن */
         {                            /* شروع بدنه حلقه داخلی */
            dvd = dvd + ۱;            /* آماده کردن مقسوم علیه بعدی */
            rem = prm % dvd;          /* محاسبه باقیمانده */
         } while (rem > ۰ && dvd < prm - ۱);/* پایان حلقه بررسی اول بودن */
         if (rem > ۰)                 /* تست برای چاپ عددی که اول بوده */
            printf ("%d\n", prm);
      }                               /* پایان حلقه تکرار ساختن اعداد */
   return (۰);                        /* خاتمه طبیعی برنامه */
}                                     /* علامت پایان جملات برنامه */
```

شکل ۳-۷: برنامه چاپ اعداد اول از ۳ تا n.

نکات مهم در نوشتن یک برنامه به زبان C

فرمت آزاد: توصیه موکد در مورد رعایت کنگره‌بندی به عنوان یک قاعده برنامه‌نویسی ساختیافته.

ساختار تابعی: هر برنامه C متشکل از یک یا چند تابع، شامل تعدادی متغیر حاوی مقادیر و دستوراتی که باید روی آنها اجرا شود. لزوم وجود یک تابع با نام main در مجموعه توابع یک برنامه.

دستورهای مرحله پیش ترجمه: حاوی فرمان‌هایی جهت اطلاع مواردی به کامپایلر قبل از شروع ترجمه.

ساختار بلوکی: تعریف کل هر برنامه به صورت یک بلوک محصور بین، علامت { و علامت }، بعضی سطرها یک دستور مستقل، برخی از دستورها یک بلوک.

علامت پایان دستور: یک دستور مستقل روی یک یا چند سطر، پایان آن با علامت ; عدم نیاز بعد از }.

دستورهای کوتاه: نوشتن برنامه لاغر و قدبلند بهتر از نوشتن برنامه چاق و قدکوتاه!

جداسازی کلمات: لزوم فاصله‌گذاری در مواردی که هیچ جداکننده‌ای بین دو کلمه وجود نداشته باشد.

لزوم دستور بازگشت: پایان منطقی (آخرین دستوری که اجرا می‌شود) دستور return (۰); به معنی بازگشت یا خاتمه طبیعی برنامه.

توضیحات در برنامه: توضیحات محصور بین علائم /* و علائم */، هر جایی که امکان قرار دادن فاصله خالی، Tab یا Enter باشد.

۳-۵-۲ الفبای زبان C

الفبا، مجموعه علائم‌یا مجموعه کاراکترها: زیرمجموعه‌ای از کد ASCII **بر مبنای استاندارد ۶۴۶-۱۹۸۳** ISO Invariant Code Set**، شامل دو دسته و شش گروه.**

دسته اول شامل پنج گروه: حروف بزرگ (از A **تا** Z**)، حروف کوچک (از** a **تا** z**)، ارقام (از ۰ تا ۹)،**
عملگرها (مثل = > < % / * - + **) و جداکننده‌ها (مثل** " ' ; : **)**

نکته: وجود تفاوت بین حروف کوچک و بزرگ در این زبان.

مثال: غلط بودن IF**،** If **یا** iF **به جای کلمه** if **برای نوشتن جملات شرطی.**
تفاوت نام‌های sum**،** SUM **و** Sum **با یکدیگر در نام‌گذاری مقادیر.**

دسته دوم: علائم غیرقابل چاپ یا به سختی قابل چاپ در جدول ۳-۲
دلیل استفاده از این روش برای بیان این کاراکترها در راستای قابل حمل بودن یا مستقل از ماشین بودن

جدول ۳-۲: مجموعه علائم غیرقابل چاپ یا به سختی قابل چاپ.

نام علامت	نحوه نوشتن در زبان C	کد ASCII	معادل مبنای ده
صدای زنگ (beep)	\a	۰۰۰۰۰۱ ۱۱	۷
عقب‌گرد (backspace)	\b	۰۰۰۰۱۰ ۰۰	۸
اعلام شروع صفحه(form feed)	\f	۰۰۰۰۱۱ ۰۰	۱۲
اعلام شروع سطر(line feed)	\n	۰۰۰۰۱۰ ۱۰	۱۰
اعلام پایان سطر (carriage return)	\r	۰۰۰۰۱۱ ۰۱	۱۳
فاصله افقی (horizontal tab)	\t	۰۰۰۰۱۰ ۰۱	۹
فاصله عمودی (vertical tab)	\v	۰۰۰۰۱۰ ۱۱	۱۱

۳-۵-۳ مقادیر ثابت

اعداد صحیح

- دنباله‌ای از ارقام و در صورت لزوم یک علامت + یا – هم در سمت چپ قبل از همه رقم‌ها.
- ذخیره یک عدد صحیح در زبان C در تعداد مشخصی بایت‌های حافظه (معمولاً یک، دو یا چهار بایت).
- ساده‌ترین عدد صحیح در یک بایت ، یک بیت علامت و هفت بیت عدد، محدوده بین ۱۲۸- و ۱۲۷+.
- امکان ذخیره کد عددی معادل یک کاراکتر (ثابت کاراکتری) در یک بایت.
- ثابت کاراکتری 'f' دارای معادل عددی ۰۱۱۰۰۱۱۰ در مبنای دو یا ۱۰۲ در مبنای ده.
- ثابت کاراکتری '۵' عبارت است از ۰۰۱۱۰۱۰۱ یا ۵۳ (توجه به تفاوت آن با ارزش عدد ۵).
- '\a' که نشان‌دهنده کاراکتر صدای زنگ است دارای معادل عددی ۷.
- عدد صحیح دو بایتی (۱۵ بیت عدد و یک بیت علامت) بین ۳۲۷۶۸- و ۳۲۷۶۷+.
- عدد صحیح چهار بایتی (۳۱ بیت عدد و یک بیت علامت) بین ۲۱۴۷۴۸۳۶۴۸- و ۲۱۴۷۴۸۳۶۴۷+.
- اعلام کوچکترین و بزرگترین عدد صحیح قابل نمایش برای حالت‌های مختلف در فایل limits.h.
- مثال‌هایی از اعداد صحیح در زبان C: ۲۱۴۵۱-، ۵۴۳۲۱+، ۰، ۲۴۶۸، ۱۲۳۴۵۶۷۸۹۰ و ۱۱۲۲۳۳۴۴-.

اعداد اعشاری معمولی

- دو قسمت صحیح و اعشار جدا شده با علامت . (نقطه) به معنی ممیز اعشاری،
- هر قسمت دنباله‌ای از رقم‌ها، امکان قرار دادن یک علامت + یا – هم در سمت چپ قبل از همه رقم‌ها، وجود ممیز الزامی، امکان حذف یکی از قسمت‌های صحیح یا اعشار.
- در تعداد مشخصی بایت‌های حافظه (معمولاً چهار یا هشت بایت) ذخیره می‌شود
- عدد اعشاری چهار بایتی، دقت عدد شش رقم، بزرگی آن بین $۱۰^{-۳۸}$ تا $۱۰^{+۳۸}$
- عدد اعشاری هشت بایتی، دقت ۱۵ رقم و بزرگی بین $۱۰^{-۳۰۸}$ و $۱۰^{+۳۰۸}$
- اعلام کوچکترین و بزرگترین عدد اعشاری قابل نمایش در زبان C در فایلی با نام float.h.
- مثال‌هایی از عدد اعشاری معمولی: ۱۲۳.۴۵۶، ۱۱.۲۲۳۳-، ۰.۳۲۱ ، ۹۸۷۶.۶۵۴+، ۰.۷۴۳- و ۰.۲۳۴۵+.

اعداد اعشاری با توان علمی

- شامل دو بخش پایه یا مانتیس و توان جدا شده با حرف E یا e.
- قسمت پایه یک عدد اعشاری معمولی یا یک عدد صحیح، امکان قرار دادن علامت + یا –در سمت چپ.
- قسمت توان حتماً یک عدد صحیح، امکان قرار دادن علامت + یا –در سمت چپ توان.
- نحوه ذخیره در داخل حافظه هر دو نوع یکسان.
- مثال‌های این‌گونه اعداد: ۱۲۳e۱۲+، ۴۵۶E-۱۲، ۱۲۳.e۱۵–، ۱.۳۵۶۷E-۲۳، ۶۷۸۹.۵E-۳– و .۰E۰

۳-۵-۴ نام‌گذاری متغیرها

- ذخیره شدن هر مقدار در برنامه تحت یک نام (متغیر یا شناسه).
- استفاده از متغیر به عنوان محلی برای ذخیره و بازیابی داده‌ها.
- وجود مؤلفه‌هایی برای هر متغیر جهت تعریف آن از جنبه‌های مختلف.
- برخی از این مؤلفه‌ها: نام، نوع، طول، مقدار اولیه، طول عمر، محل و مقدار جاری.

قواعد نام‌گذاری متغیرها در زبان

- کاراکتر اول نام متغیر باید یکی از حروف الفبای انگلیسی (کوچک یا بزرگ) یا علامت _ باشد.
- سایر کاراکترهای نام می‌تواند از حروف، ارقام یا علامت _ باشد.
- تعداد کاراکترهای یک نام محدودیتی ندارد ولی هر اسمی از روی ۳۱ کاراکتر اولش شناسایی می‌شود.
- در نام‌گذاری متغیرها کاراکتر _ علامت خط فاصله یا تفریق نیست بلکه علامت underscore است.
- تفاوت حروف کوچک و بزرگ در زبان C، مثلاً A و a دو اسم متفاوت.

قاعده: توصیه عدم استفاده از علامت _ برای شروع اسامی.

قرارداد: حروف کوچک برای نام‌گذاری متغیرها و حروف بزرگ برای نامیدن مقادیر ثابت و نوعهای جدید.

توجه: رزرو بودن کلمات کلیدی برای ساختن جملات زبان C شامل مجموعه زیر:

```
auto      do        for       return    switch
break     double    goto      short     typedef
```

```
case      else      if        signed    union
char      enum      int       sizeof    unsigned
const     extern    long      static    void
continue  float     register  struct    volatile
default                                 while
```

- **لزوم وجود تفاوت در ۳۱ کاراکتر اول در اسامی با طول بیشتر از ۳۱ کاراکتر.**
- **رعایت قوانین هر دو محیط برای اسامی بین یک برنامه C و محیط خارج (زبان اسمبلی یا سیستم عامل)**

اسامی قابل قبول:

```
number  tedad  no_of_students  standard_deviation_of_marks  st۲۵_x۷
```

اسامی غیرقابل قبول:

```
Std-no  name$of$car  while  ۴tran_programming  MY-NAME-is  _student#
```

۳-۶ دستورهای تعریف نوع متغیرها

قالب کلی یک دستور تعریف نوع یا اعلان: ‹نوع متغیرها› ‹لیست متغیرها› ;

```
int number, tx۷۵_f۸, m, sum_of_students;        /* تعریف متغیرهای صحیح معمولی */
float max, average۵, standard_deviation, a, b;  /* تعریف متغیرهای اعشاری معمولی */
char tab, quest, ff_۱, initial;                 /* تعریف متغیرهای صحیح یک بایتی */
```

مشخص شدن مؤلفه‌های نام، نوع و طول متغیرهای ذکر شده در دستورات فوق.

۳-۷ عملگرها، عملوندها و عبارات

عملگرها یا اپراتورها: علائمی که وقتی به شکل تعریف شده در کنار یا بین مقادیر قرار می‌گیرند باعث انجام یک عمل و حصول یک نتیجه می‌شود.

مثال: ۳۲ + ۱۵ با مفهوم جمع زدن.

عملوند: یک عدد، یک متغیر، احضار یک تابع ویا یک ترکیب درست از عملگر و عملوندها

آثار جانبی: وجود آثار جانبی روی عملوندها در بعضی عملگرها علاوه بر نتیجه عمل.

عملگرهای خاص: امکان اعمال بعضی عملگرها فقط روی نوع خاصی از مقادیر.

عبارات: ترکیب درستی از عملگرها و عملوندها.

- وجود یک مجموعه غنی از عملگرها (محاسبه‌ای، مقایسه‌ای، منطقی)
- نتیجه حاصل از همه عملگرها فقط مقدار عددی.
- عبارات زبان C فقط محاسبه‌ای، بیان تحت عنوان عبارات.

۳-۷-۱ عملگرهای محاسبه‌ای و تخصیص (تخصیص به عنوان عملگر در زبان C و در انواع مختلف)

جدول ۳-۳: تعدادی از عملگرهای محاسبه‌ای و تخصیص همراه با نحوه عمل آنها.

ردیف	شکل عملگر	نوع عملگر	نحوه کار
۱	+	دوتایی	جمع زدن دو عملوند سمت راست و سمت چپ عملگر و ارائه حاصل‌جمع به عنوان نتیجه.
۲	-	دوتایی	تفریق عملوند سمت راست از عملوند سمت چپ و ارائه حاصل تفریق به عنوان نتیجه.
۳	-	یکتایی	منفی‌کردن عملوند موجود در سمت راست عملگر، و ارائه حاصل عمل به عنوان نتیجه.
۴	*	دوتایی	ضرب‌کردن دو عملوند سمت راست و چپ در یکدیگر و ارائه حاصل‌ضرب به عنوان نتیجه.
۵	/	دوتایی	تقسیم عملوند سمت چپ بر عملوند سمت راست و ارائه خارج قسمت به عنوان نتیجه.
۶	%	دوتایی	محاسبه باقیمانده تقسیم عملوند سمت چپ بر عملوند سمت راست (هر دو عملوند باید مقدار صحیح باشند) و ارائه این باقیمانده به عنوان نتیجه.
۷	=	دوتایی	تخصیص مقدار عملگر سمت راست به عملگر سمت چپ و ارائه آن مقدار به عنوان نتیجه.

۳-۷-۲ عبارات

- هر ترکیب درستی از عملگرها با مقادیر ثابت، متغیرها و احضار توابع به عنوان عملوند آنها.
- استفاده از زوج پرانتز برای دسته‌بندی عملگرها و تعیین ترتیب اجرای آنها.

یک تعریف ساده و قابل فهم از عبارت.

- هر مقدار ثابت، اسم متغیر و احضار تابع به تنهایی یک عبارت درست است.
- از قرار دادن یک عملگر یکتایی در کنار یک عملوند یک عبارت درست حاصل می شود (عملوند باید یک عبارت درست باشد، محل عملگر در سمت چپ یا راست با توجه به نوع عملگر).
- ترکیب یک عملگر دوتایی و دو عملوند در دوطرف آن که هرکدام باید یک عبارت درست باشد، خود یک عبارت درست خواهد بود.
- در نتیجه قراردادن یک عبارت درست در داخل یک زوج پرانتز، یک عبارت درست حاصل می‌شود

مثالهایی از اولین قاعده:

```
۰  -۱۱۲  ۳۲۲۱۱  +۱.۷  ۱۲e-۱۵  't'  m  number  standard_dev  sin(x)
```

مثالهایی از قاعده دوم:

```
-+۱۵.۷  -  -۱۲۳  -  -۱۲e-۵  -'t'  -  -k  -  -  -m  -standard_dev  -sin(x)
```

مثالهایی از قاعده سوم:

```
۱.۷+۳۲  ۱۲e-۵*-۱۲  m+n  m٪۱۵  t='T'-'A'  n=m+'x'
'm'+-۵  x*-  -  -y  a*b+c/-d  alpha+sin(betta)  d=b*b-۴*a*c
```

مثالهایی از تعدادی عبارت کلی:

```
(a + b) * (c + d) / m - n
-sum*(۵٪prm+(s=n/۵۷.۲)*۱۲۱)
res = (-b + sqrt(b * b - ۴ * a * c)) / (۲ * a)
s=t=(a+b*(m-۵)/(-(n+۵.۶)/(m=n+۱۲*-k)+-s*t)-۴.۱۲)/(-a+b)
m = n = k + ۱ / (j = k * ۲)
```

تقدم و ترتیب اجرای عملگرها

- تعریف اولویت اِعمال عملگرها و ترتیب انجام عملگرهای مشابه در حدامکان.
- وجود موارد خاص دقیقاً تعریف نشده به دلیل محدودیت‌های مربوط به کامپیوتر مورد استفاده.
- لزوم رجوع به کامپایلر مورد استفاده روی کامپیوتر مربوطه برای روشن شدن آنها.

اولویت و ترتیب اجرای عملگرها

- اول: عملگرهای منفی‌کردن از راست به چپ.
- دوم: عملگرهای ضرب، تقسیم و باقیمانده با تقدم یکسان از چپ به راست.
- سوم: عملگرهای جمع و تفریق با تقدم یکسان از چپ به راست.
- چهارم: عملگر تخصیص از راست به چپ.
- شروع محاسبات از عبارت‌های داخل زوج پرانتزها (داخلی ترین زوج پرانتز اول) طبق همین اولویت‌ها.

توجه: عدم وجود قانونی برای این که کدام عملوند از یک عملگر دوتایی اول مورد دستیابی و محاسبه قرار می‌گیرد،مگر موارد ارائه شده در اولویت‌های فوق.

مثال: در عبارت (a+b)*(c-d) اول زوج پرانتز سمت چپی یا زوج پرانتز سمت راستی مشخص نیست.

امکان وجود اثر جانبی

مثال: با فرض اینکه متغیر صحیح a حاوی عدد ۴ باشد، حاصل b=(a=a-۲)*(a+۶) برابر ۱۶ می شود.

در عبارت معادل مفصل‌تر b=(a=a-۲)*((a+۶)*(a+۶)/(a+۶))، حاصل برابر ۲۰ خواهد بود!

جدول ۳-۴: نمونه‌هایی از عبارات دارای ابهام در ترتیب محاسبه همراه با نتیجه محاسبه در دو کامپایلر مختلف.

عبارت با فرض این‌که a حاوی یک باشد	نتیجه اجرا		توضیح
b=(a=۳)+(a=۲)	Turbo C	a:۳, b:۵	داخل زوج پرانتز سمت راست اول انجام شده است.
b=(a=۳)+(a=۲)	Borland C	a:۲, b:۵	داخل زوج پرانتز سمت چپ اول انجام شده است.
b=((a=۲)+۱)+(a=۳)	هر دو کامپایلر	a:۳, b:۶	داخل زوج پرانتز سمت چپ اول انجام شده است.
b=(a=۳)+((a=۲)+۱)	هر دو کامپایلر	a:۳, b:۶	داخل زوج پرانتز سمت راست اول انجام شده است.
b=(a+۱)+(a)+(a)*(a=۵)	هر دو کامپایلر	a:۵, b:۳۶	داخل زوج پرانتزها از راست به چپ محاسبه شده است.
b=(a)*(a=۵)+(a+۱)+(a)	هر دو کامپایلر	a:۵, b:۳۶	نخست زوج پرانتز دوم از سمت چپ، بعد زوج پرانتز اول از سمت چپ و سپس دو زوج پرانتز بعدی محاسبه شده است
b=(a=۲)+(a+۱)+(a)*(a=۳)	هر دو کامپایلر	a:۲, b:۱۴	از راست به چپ نخست زوج پرانتز اول، بعد زوج پرانتز دوم، سپس زوج پرانتز چهارم و نهایتاً زوج پرانتز سوم محاسبه شده است
b=(a)*(a=۳)+(a=۲)+(a+۱)	هر دو کامپایلر	a:۲, b:۱۴	از راست به چپ نخست زوج پرانتز سوم، بعد زوج پرانتز چهارم، سپس زوج پرانتز دوم و نهایتاً زوج پرانتز اول محاسبه شده است

- امکان ایجاد پیچیدگی‌های فراوان، بهترین راه تفکیک عبارت‌های دارای اثر جانبی

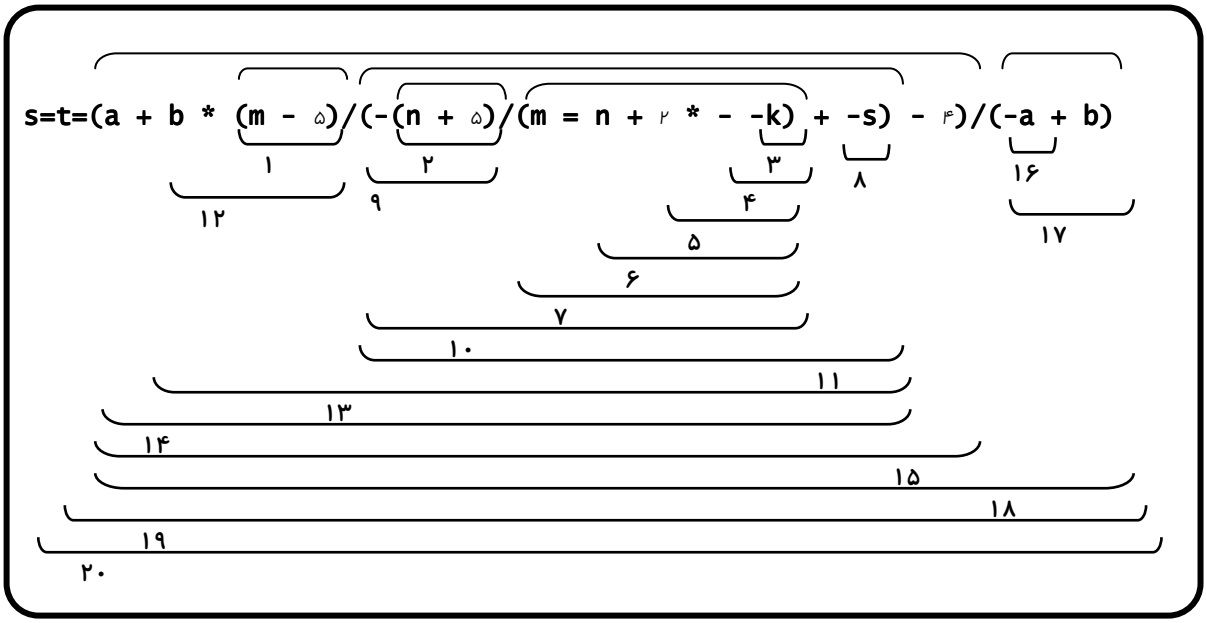
مثلاً اگر در عبارت b=(a=a-۲)*(a+۶) منظور محاسبه زوج پرانتزهای هم‌سطح از راست به چپ باشد باید به صورت زیر نوشته شود.

```
t=a+۶;
b=(a=a-۲)*t;
```

- در احضار توابع مثلاً در عبارت y=f(x)+g(x) مشخص نیست که آیا اول احضار تابع f انجام می‌شود یا احضار تابع g.
- در محاسبه یک عبارت با آثار جانبی، در استاندارد زبان C هیچ قاعده‌ای ارائه نشده است
- اِعمال این آثار به ساختار سخت‌افزار کامپیوتر بستگی دارد که هر کامپایلری با توجه به آن ساختار و شرایط لحظه‌ای زمان ترجمه عمل می‌نماید.

- از نظر اصول برنامه‌نویسی نوشتن یک عبارت به شکلی که ترتیب انجام مواردی که تعریف نشده روی نتیجه آن اثر داشته باشد کار درستی نیست.
- بهتر است از نوشتن این‌گونه عبارت‌ها به طور کلی اجتناب شده و نحوه دقیق محاسبه با جداسازی عبارت‌های مبهم دقیقاً مشخص گردد.

محاسبه یک عبارت مفصل در شکل ۳-۸، پرانتزهای هم‌سطح از چپ به راست



شکل ۳-۸: ترتیب و تقدم اجرای عملگرها در یک عبارت نمونه.

نوع عبارات

- دو مقدار از یک نوع ، نتیجه از همان نوع، دو مقدار با نوع‌های متفاوت، نتیجه از نوع پیچیده، بنابراین:
- نتیجه هر عمل محاسبه‌ای روی دو مقدار صحیح همواره صحیح است.
- نتیجه هر عمل محاسبه‌ای روی دو مقدار که حداقل یکی از آنها اعشاری باشد همواره اعشاری است.
- نتیجه هر عمل محاسبه‌ای روی دو مقدار همنوع با اندازه‌های متفاوت، از نوع با اندازه بزرگتر است.
- کاربرد بعضی از عملگرهای محاسبه‌ای فقط روی نوع خاصی از عملوندها، عملگر % (باقیمانده).

مثال متغیرهای m و n از نوع صحیح و به ترتیب حاوی مقادیر ۵ و ۲

- x و y متغیرهای اعشاری و به ترتیب حاوی مقادیر ۵.۰ و ۲.۰
- m+n برابر ۷، m/n برابر ۲ و m%n برابر ۱ خواهد بود
- x*y برابر ۱۰.۰، x/y برابر ۲.۵، m/y و x/n هم برابر ۲.۵
- m%x، x%n و x%y همه غلط هستند (چرا؟).

در عملگر تخصیص

- نوع نتیجه به نوع عملوند سمت چپ بستگی دارد که باید یک عملوند قابل تغییر باشد.
- عملگر تخصیص در متن یک عبارت، معمولاً در داخل یک زوج پرانتز.
- داشتن یک یا چند تخصیص متوالی به‌عنوان آخرین عملگرها.

مثال: m و n صحیح و به ترتیب حاوی مقادیر ۵ و ۲، x و y اعشاری و به ترتیب حاوی مقادیر ۵.۰ و ۲.۰.

- در هر دو عبارت m=m/n و m=x/y مقدار جدید m برابر ۲ خواهد شد.
- حاصل نهایی m=m/y و m=x/n هم برابر ۲ خواهد شد که در متغیر m ذخیره می‌گردد.
- در x=m=x/y نتیجه x/y برابر ۲.۵ می‌باشد، m حاوی مقدار صحیح ۲ و x حاوی مقدار اعشاری ۲.۰

عبارت به عنوان یک دستور مستقل در برنامه

قالب کلی ‹ عبارت ›;

نحوه اجرای این دستور

```
main()                                  /*  برنامه حاوی عبارات نمونه  */
{
   int m, n, k, j;                      /*  تعریف متغیرهای صحیح  */
   float x, y, z;                       /*  تعریف متغیرهای اعشاری  */

   m = 5;                               /*  مقدار m برابر۵ است  */
   n = 13;                              /*  مقدار n برابر۱۳ است  */
   j = k = ۲;                   /*  مقدار j برابر۲ و مقدار k هم برابر۲ است  */
   x = 5.0;                             /*  مقدار x برابر  ۵.۰ است  */
   y = 2.5; z = 15;             /*  دو دستور در یک سطر، مقدار z برابر۱۵.۰ و مقدار y برابر  ۲.۵ است  */
   k = (n + ۴) / m;             /*  مقدار k برابر۳ است  */
   k = (n + ۴) % m;             /*  مقدار k  برابر۲ است  */
   k = n + ۴ / m;                       /*  مقدار k برابر۱۳ است  */
   k = n + ۱۰ / m + ۱۰;         /*  مقدار k برابر۲۵ است  */
   k = n * ۱۰ / m * ۱۰;         /*  مقدار k  برابر۲۶۰ است  */
   k = (m + ۲) / j;             /*  مقدار k برابر۳ است  */
   k = (m + ۲.۰) / j;                   /*  مقدار عبارت سمت راست تقصیص ۳.۵ ولی مقدار k برابر۳ است  */
   k = (x + ۲) / j;             /*  مقدار عبارت سمت راست تقصیص ۳.۵ ولی مقدار k برابر۳ است  */
   z = (m + ۲) / j;             /*  مقدار عبارت سمت راست تقصیص ۳ ولی مقدار z برابر۳.۰ است  */
   z = m + ۲.۰ / j;             /*  مقدار عبارت سمت راست تقصیص ۶.۰ و مقدار z هم۶.۰ است  */
   z = (m + ۲.۰) / j;                   /*  مقدار عبارت سمت راست تقصیص ۳.۵ و مقدار z هم۳.۵ است  */
   z = ۲.۰ + m / j;             /*  مقدار عبارت سمت راست تقصیص ۴.۰ و مقدار z هم۴.۰ است  */
                                        /*  مقدار عبارت سمت راست تقصیص ۳.۵ و مقدار جدید yهم۳.۵ است  */
                                        /*  مقدار kبرابر ۳ و مقدار جدید zبرابر۳.۰ است  */
   z = k = y = (y * ۲.۰ + j) / j;
                                /*  مقدار عبارت سمت راست تقصیص به  y  ۳.۵ و مقدار جدید xبرابر۷.۰ است  */
                                        /*  مقدار j , nبرابر ۲ و مقدار جدید yبرابر۳.۵ است  */
                                        /*  مقدار kبرابر ۳ و مقدار جدید zبرابر۳.۰ است  */
   z = k = y = (x = x + ۲.۰)/(j = n = m / j);
   return (۰);
}
```

شکل ۳-۹: نمونه‌هایی از عبارات به عنوان دستورهای برنامه.

۳-۸ چاپ یا نمایش در خروجی

- عدم وجود دستورات خاص برای عملیات ورودی و خروجی در زبان C برخلاف خیلی زبانها.
- اجتناب ناپذیر بودن انجام این‌گونه عملیات در هر زبان از جمله زبان C.
- انجام این عملیات از طریق مجموعه‌ای از توابع آماده، تعاریف آنها در فایل سرآمد `stdio.h` .
- لزوم `#include` کردن این فایل به برنامه.
- وجود روش‌های متنوع و منابع مختلف برای انجام عملیات ورودی و خروجی در زبان C.
- تخصیص ورودی و خروجی استاندارد به هر برنامه، صفحه کلید و صفحه نمایش.
- نوشتن با فرمت روی خروجی استاندارد.
- امکان تغییر مسیر ورودی و خروجی استاندارد به فایل‌های داده‌ای روی حافظه فرعی.

۳-۸-۱ چاپ با فرمت

- آماده‌سازی داده‌های مورد نظر طبق فرمت یا الگوی دلخواه و سپس نمایش روی صفحه یا چاپ
- استفاده از تابع `printf` با قالب کلی زیر.

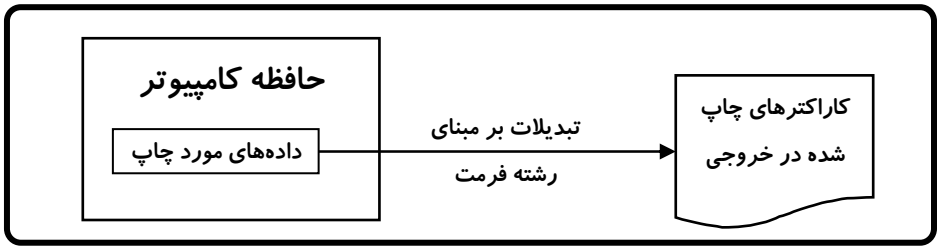
`printf(`‹رشته فرمت› , ‹لیست مقادیر مورد چاپ›`)`

چند نکته کلی در مورد توابع

- تشابه معنای توابع در زبان C با معنای آن در ریاضی.
- ارسال مقادیری به عنوان آرگومان در هنگام احضار.
- انجام عملیاتِ از قبل تعریف شده توسط تابع.
- برگرداندن مقداری به عنوان نتیجه تابع.

- آرگومانهای ارائه شده در احضار این تابع رشته فرمت و لیست مقادیر مورد چاپ.
- عمل از قبل تعریف شده چاپ مقادیر داده شده طبق فرمت تعیین شده.
- نتیجه‌ای که برگردانده می‌شود تعداد کاراکتر چاپ شده.
- استفاده از احضار مانند نام متغیر به عنوان یک عملوند در عبارت‌ها، مشروط بر این‌که هیچ اثر جانبی بر آن مترتب نباشد.

- قرار دادن احضار تابع به تنهایی در یک دستور (عبارت به عنوان یک دستور مستقل).
- آماده نمودن مقادیر داده شده در لیست طبق الگوی اعلام شده در رشته فرمت و نمایش در خروجی.
- برگرداندن تعداد کاراکترهای چاپ شده به عنوان نتیجه یا ۱- در صورت عدم موفقیت.
- اعلام مقادیر مورد چاپ در قالب اسامی متغیرها یا عبارات در لیست مقادیر، جدا سازی با علامت کاما.
- تعیین نحوه چاپ هر مقدار در رشته فرمت از طریق مشخصه‌های تبدیل فرمت.
- قرار دادن پیغام‌ها و توضیحات مورد چاپ و علائم مربوط به سطربندی و صفحه‌بندی در رشته فرمت.



شکل ۱۰-۳: نحوه کار تابع printf (چاپ با فرمت).

مثال:

```
int m;
char c;
float x;
m = ۱۲۳;
c = 't';
x = ۱۵.۴;
printf("the value of m:%d\tthe value of x:%f\tthe value of
c:%c\n",m,x,c);
```

- رشته فرمت مشخص کننده نوع مقادیر، طول مقادیر و محل چاپ مقادیر به همراه توضیحات بین آنها.
- m و x و c نام متغیرهایی (عبارت‌هایی) که باید مقادیر آنها چاپ شود.
- %d ، %f و %c مشخصه‌های تبدیل $\frac{1}{b}$ فاصله، به ترتیب از چپ به راست مربوط به عبارت‌های مورد چاپ.
- اولین مشخصه چاپ یک عدد صحیح مبنای ده، دومی چاپ یک عدد اعشاری مبنای ده و سومی چاپ کاراکتر معادل یک عدد صحیح.
- کاراکترهای کنترلی: \t رها کردن تعدادی فاصله خالی و \n باعث انتقال به اول سطر بعدی.

- سایر مطالب در رشته فرمت: توضیحات که عینا چاپ می شود.

نتیجه اجرای دستور فوق

```
the value of m:۱۲۳    the value of x:۱۵.۴۰۰۰۰۰    the value of c:t
```

مثال دیگر: ذخیره و استفاده از نتیجه این تابع.

```
int m, cnt;
char c;
float x;
m = ۱۲۳;
x = ۱۵.۴;
c = ‘t’;
cnt = printf
     (“the value of m:%d\tthe value of x:%f\tthe value of
c:%c\n”,m,x,c);
printf(“number of characters printed in the previous line is:%d\n”,
cnt);
```

نتیجه اجرای دستور فوق

```
the value of m:۱۲۳ the value of x:۱۵.۴۰۰۰۰۰  the value of c:t
number of characters printed in the previous line is:۶۱
```

۳-۸-۲ رشته فرمت و مشخصه‌های تبدیل

- رشته فرمت: یک ثابت رشته ای در قالب دنباله‌ای از کاراکترها در داخل یک زوج علامت ‘‘.
- روش ذخیره ثابتهای رشته‌ای در زبان C، حالت پویا و قابل تعریف و تغییر در زمان اجرا.
- لزوم درج ثابت رشته‌ای به طور کامل روی یک سطر، غلط نحوی درصورت شکستن آن به دو سطر.
- رشته فرمت مبنای عمل چاپ بر اساس مشخصه های فرمت.
- شروع هر مشخصه فرمت با علامت %، بعد از آن یک علامت تبدیل (مثل d و f و c).
- تبدیل داده‌های مبنای دو موجود در داخل حافظه (صفر و یک‌ها) به شکل خواسته شده برای چاپ.
- موارد دیگر بین % و علامت تبدیل به ترتیب زیر در صورت نیاز (وجود پیش‌فرض مناسب).

- طول میدان: عدد صحیح مبین حداقل تعداد ستون مورد استفاده برای چاپ.

- طول میدان در مقابل طول مقدار مورد چاپ (تعداد کاراکتر)، احتساب علامت، طول میدان پیش‌فرض.
- جداکننده: علامت . (نقطه) برای جداکردن طول میدان از دقت مقدار.

- دقت مقدار: عدد صحیح، برای مقادیر اعشاری تعداد ارقام اعشار، برای مقادیر صحیح حداقل تعداد رقم.
- گرد شدن و صفر اضافه در ارتباط با دقت، حذف ممیز اعشار در دقت صفر، دقت پیش‌فرض.

- رشته فرمت مبنای عمل چاپ، نیاز به یک مقدار برای هر مشخصه تبدیل فرمت.
- تعداد عبارت‌ها کم‌تر از تعداد مشخصه‌های تبدیل، خطرناک!
- تعداد عبارت‌های بیشتر از تعداد مشخصه‌ها، توجه به آثار جانبی.
- عدم وجود هیچ‌گونه تضمین برای ترتیب محاسبه مقادیر مورد چاپ (چپ به راست یا راست بـه چـپ) استاندارد زبان C (ترتیب محاسبه آرگومانهای توابع قبل از احضار)، توجه به آثار جانبی.

- مثال: دستور زیر با فرض k متغیر صحیح و حاوی عدد ۵.

```
printf("first k=٪٣d,second k=٪٣d,last k=٪٣d\n", k, k=k*٢, k);
```

نتیجه چاپ در صورت محاسبه مقادیر از چپ به راست:

```
first k=  ۵, second k = ۱۰, last k= ۱۰
```

نتیجه چاپ در صورت محاسبه مقادیر از راست به چپ (Borland C Version ۳.۱).

```
first k= ۱۰, second k = ۱۰, last k=  ۵
```

جدول ۳-۵: مثالهایی از فرمت چاپ برای مشخصه‌های تبدیل %d و %c با متغیرهای صحیح m و n به ترتیب حاوی مقادیر ۱۲۳۴ و ۴۳۲۱-، متغیر صحیح یک بایتی t حاوی ثابت کاراکتری 'p' و متغیر اعشاری x حاوی مقدار ۱۲۳.۴۵۶ .

ردیف	فرمت	عبارت	نتیجهٔ چاپ	توضیح
۱	%d	m	"۱۲۳۴"	طول میدان ۴ به صورت پیش‌فرض استفاده شده است.
۲	%d	n	"-۴۳۲۱"	طول میدان ۵ به صورت پیش‌فرض استفاده شده است.
۳	%۴d	m	"۱۲۳۴"	طول میدان ۴ با اندازهٔ مقدار مطابقت دارد.
۴	%۴d	m+n	"-۳۰۸۷"	طول میدان ۴ کافی نیست بنابراین ۵ به جای آن استفاده شده است.
۵	%۶d	m	" ۱۲۳۴"	با توجه به طول میدان ۶ دو فاصلهٔ خالی در سمت چپ اضافه شده است.
۶	%۸.۶d	m	"۰۰۱۲۳۴"	با توجه به طول میدان ۸ و دقت ۶ دو فاصلهٔ خالی و دو صفر بی‌ارزش در سمت چپ اضافه شده است.
۷	%d	t	"۱۱۲"	معادل عددی کد ASCII برای p عدد ۱۱۲ است که با طول میدان پیش‌فرض ۳ چاپ شده است.
۸	%۲d	t	"۱۱۲"	طول میدان ۲ کافی نیست بنابراین ۳ به جای آن استفاده شده است.
۹	%۳d	t-'a'	" ۱۵"	تفاضل معادلهای عددی 'p' و 'a' برابر ۱۵ است که با طول میدان ۳ نیاز به یک فاصلهٔ خالی دارد.
۱۰	%۷d	-(t+۷)	"-۱۱۹"	حاصل ۱۱۹- است که با توجه به طول میدان ۷ با سه فاصلهٔ خالی چاپ می‌شود.
۱۱	%c	t	"p"	طول میدان ۱ به صورت پیش‌فرض استفاده شده است.
۱۲	%۳c	t	" p"	با توجه به طول میدان ۳ دو فاصلهٔ خالی در سمت چپ اضافه شده است.
۱۳	%۵c	t+۲	" r"	حاصل عبارت معادل عددی کاراکتر 'r' است که با توجه به طول میدان ۵ نیاز به چهار فاصلهٔ خالی دارد.
۱۴	%c	m-۱۱۱۲	"z"	حاصل عبارت ۱۲۲ است که کد ASCII معادل حرف z است.
۱۵	%d	x	"."	نوع متغیر با نوع مشخصهٔ تبدیل فرمت تطبیق ندارد (مقدار اعشاری با مشخصهٔ تبدیل فرمت عدد صحیح)، نتیجهٔ چاپ قابل پیش‌بینی نیست.

جدول ۳-۶: مثالهایی از فرمت چاپ برای مشخصهٔ تبدیل فرمت %f، با متغیرهای اعشاری x و y به ترتیب حاوی ۱۲۳.۴۵۶ و ۳۱.۲۳۴- و متغیر صحیح m حاوی عدد ۱۲۳۴ می‌باشند.

ردیف	فرمت عبارت	نتیجهٔ چاپ	توضیح
۱	%f x	"۱۲۳.۴۵۶۰۰۰"	طول میدان ۱۰ و دقت ۶ به صورت پیش‌فرض استفاده شده است.
۲	%f y	"-۳۱.۲۳۴۰۰۰"	طول میدان ۱۰ و دقت ۶ به صورت پیش‌فرض استفاده شده است.
۳	٪۶f x+y	"۹۲.۲۲۲۰۰۰"	چون دقت ۶ فرض می‌شود، طول میدان ۶ کافی نیست طول میدان ۹ به صورت پیش‌فرض استفاده می‌شود.
۴	٪۱۰f x+y	" ۹۲.۲۲۲۰۰۰"	با توجه به طول میدان ۱۰ و دقت پیش فرض ۶ یک فاصلهٔ خالی در سمت چپ اضافه شده است.
۵	%.۲f x	"۱۲۳.۴۶"	با توجه به دقت ۲ عدد تا دو رقم اعشار گرد شده، طول میدان ۶ به صورت پیش‌فرض استفاده شده است.
۶	٪۸.۲f x	" ۱۲۳.۴۶"	با توجه به دقت ۲ عدد تا دو رقم اعشار گرد شده و با توجه به طول میدان ۸ دو فاصلهٔ خالی در سمت چپ اضافه شده است.
۷	٪۶.۵f x	"۱۲۳.۴۵۶۰۰"	با توجه به دقت ۵ دو صفر به سمت راست اضافه شده ولی طول میدان ۶ کافی نیست بنابراین طول میدان ۹ به جای آن استفاده شده است.
۸	٪۱۱.۵f y	-۳۱.۲۳۴۰۰"	" با توجه به دقت ۵ دو صفر به سمت راست اضافه شده و با توجه به طول میدان ۱۱ دو فاصلهٔ خالی به سمت چپ اضافه شده است..
۹	%f m	"۱۲۳۴۰۰۰۰"	نوع عدد با نوع مشخصهٔ تبدیل فرمت تطبیق ندارد (مقدار صحیح با مشخصهٔ تبدیل فرمت عدد اعشاری)، نتیجهٔ چاپ قابل پیش‌بینی نیست.

۳-۹ برنامه‌های نمونه

- وجود تعدادی برنامه نمونه از این فصل به بعد متناسب با مطالب هر فصل.
- توصیه اکید در مورد مطالعه و فهم آنها.
- اجرای این برنامه‌ها روی کامپیوتر برای یادگیری نحوه استفاده از محیط برنامه‌نویسی C

برنامه ۳-۱: برنامه ای بنویسید که در آن یک عدد صحیح و مثبت پنج رقمی را مشخص کرده و پس از چاپ آن عدد در وسط یک سطر (عرض سطر را ۸۰ ستون فرض کنید)، ارقام عدد را جدا کرده و آنها را با فواصل مساوی روی سطر بعد چاپ نماید. نهایتاً مجموع ارقام عدد را پس از رها کردن دو سطر خالی در وسط سطر پنجم چاپ کند.

- توجه به نحوه جداسازی رقم‌های عدد.
- نحوه استفاده از طول میدان برای تنظیم محل چاپ عددها.
- کاراکتر کنترلی \n برای رها نمودن سطر.

```
#include <stdio.h>

main()                                          /* برنامهٔ جداسازی و محاسبهٔ جمع رقمها */
{  int adad, jam_e_argham;
   int r۱, r۲, r۳, r۴, r۵;

   adad = ۲۴۷۶۳;
   r۱ = adad % ۱۰;                              /* جداکردن رقم اول */
   r۲ = adad / ۱۰ % ۱۰;                         /* جداکردن رقم دوم */
   r۳ = adad % ۱۰۰۰ / ۱۰۰;                      /* جداکردن رقم سوم */
   r۴ = adad / ۱۰۰۰ % ۱۰;                       /* جداکردن رقم چهارم */
   r۵ = adad / ۱۰۰۰۰;                          /* جداکردن رقم پنجم */
   jam_e_argham = r۱ + r۲ + r۳ + r۴ + r۵;/* محاسبهٔ مجموع ارقام */
   printf("%۴۳d\n", adad);                      /* چاپ عدد در وسط سطر */
                                                /* چاپ رقمها با فواصل مساوی */
   printf("%۱۶d%۱۶d%۱۶d%۱۶d%۱۶d\n", r۵, r۴, r۳, r۲, r۱);
                                                /* چاپ جمع رقمها در وسط سطر پس از رها کردن دو سطر خالی */
   printf("\n\n%۴۱d\n", jam_e_argham);
   return (۰);                                  /* خاتمهٔ طبیعی برنامه */
```

شکل ۳-۱۱: متن برنامهٔ ۳-۱، جداسازی و محاسبهٔ جمع رقمهای یک عدد صحیح.

برنامه ۳-۲: برنامه‌ای بنویسید که در آن نخست ضرایب یک معادله درجه دوم که ریشه حقیقی دارد تعریف شود و پس از چاپ ضرایب از اول یک صفحه (هر ضریب روی یک سطر با توضیح مناسب)، معادله حل شده و جواب‌های آن روی یک سطر مجزا با یک سطر فاصله از سطر قبلی با توضیحات مربوطه چاپ گردد.

- **توجه به نحوه پرانتزگذاری در دستورهای مربوط به محاسبه ریشه‌ها.**
- **محاسبه جذر از طریق تابع `sqrt` از توابع ریاضی.**
- **اضافه کردن فایل سرآمد `math.h` حاوی تعاریف توابع ریاضی قبل از شروع برنامه.**

```
#include <stdio.h>
#include <math.h>                          /* اضافه‌کردن فایل سرآمد حاوی توابع ریاضی */

main()                                     /* برنامهٔ حل معادلهٔ درجه دوم */
{  float a, b, c;
   float x۱, x۲;

   a = ۲;
   b = ۵.۵;
   c = ۱.۷;
   printf("\fa=%f\n", a);                  /* چاپ ضرایب از اول صفحه در سه سطر */
   printf("b=%f\n", b);
   printf("c=%f\n", c);
   x۱ = (-b + sqrt(b * b - ۴ * a * c)) / (۲ * a);   /* محاسبهٔ یک ریشه */
   x۲ = (-b - sqrt(b * b - ۴ * a * c)) / (۲ * a);   /* محاسبهٔ ریشهٔ دیگر */
   printf("\nx۱=%۶.۲f\tx۲=%۶.۲f\n", x۱, x۲);       /* چاپ ریشه‌ها */
   return (۰);
```

شکل ۳-۱۲: متن برنامهٔ ۳-۲، حل معادلهٔ درجه دوم با ریشهٔ حقیقی.

برنامه ۳-۳: برنامه‌ای بنویسید که در آن یک عدد اعشاری مثبت با سه رقم صحیح و دو رقم اعشار مشخص شود. سپس نخست قسمت‌های صحیح و اعشاری عدد مزبور به دو عدد صحیح مستقل تبدیل گردد و بعد از روی این دو عدد یک عدد صحیح جدید ساخته شود که ارقام اعشاری عدد اولیه در بین ارقام صحیح آن قرارگرفته باشد. مثلاً اگر عدد اعشاری اولیه ۱۴۵.۷۶ باشد نخست دو عدد صحیح ۱۴۵ و ۷۶ و سپس عدد ۱۷۴۶۵ ساخته می‌شود. در پایان عدد اولیه را همراه با توضیح مناسب در اول یک سطر، دو عدد صحیح بعدی را پس از رهاکردن یک سطر خالی با فواصل مساوی روی یک سطر بدون توضیح (عرض سطر ۸۰ فرض شود) و نهایتاً عدد صحیح ترکیب شده را پس از رهاکردن سه سطر خالی همراه با توضیح در اول یک سطر چاپ نماید.

- **توجه به نحوه جداسازی دو قسمت یک عدد اعشاری در قالب عدد صحیح.**
- **تلفیق و تبدیل آنها به یک عدد صحیح جدید.**

```
#include <stdio.h>

main()                                          /* برنامهٔ جداسازی و تلفیق ارقام */
{  float first;
   int second, third, fourth;

   first = ۱۴۲.۷۵;                              /* تعیین عدد اولیه */
   second = first;                              /* جداسازی قسمت صحیح */
   third = (first - second) * ۱۰۰;              /* جداسازی قسمت اعشار */
   fourth =                                     /* ساختن عدد آخر در این سطر و دو سطر بعدی */
     (second / ۱۰۰ * ۱۰ + third / ۱۰);
   fourth = (fourth * ۱۰ + second / ۱۰ % ۱۰) * ۱۰ + third % ۱۰;
   fourth = fourth * ۱۰ + second % ۱۰;
   printf("The original number is: ٪۶.۲f\n",first);/* چاپ عدد اولیه */
   printf("\n٪۲۷d٪۲۷d\n", second, third);/* چاپ قسمتهای صحیح و اعشار */
                                                /* چاپ عدد نهایی */
   printf("\n\n\nThe final result with combined digits: ٪۵d\n",
          fourth);
   return (۰);
```

شکل ۳-۱۳: متن برنامهٔ ۳-۳، جداسازی و تلفیق ارقام یک عدد اعشاری.

۱۰-۳ اشتباهات متداول برنامه‌نویسی

جمع آوری اشکالات و اشتباهات متداول در ارتباط با مطالب هر فصل در پایان.

لزوم توجه به این موارد برای عدم انجام آنها.

- **برای نمایش ثابت‌های کاراکتری باید از علامت ‘ در دو طرف آن استفاده کرد که فقط یک کاراکتر در آن قرار می‌گیرد، استفاده از علامت “ در دو طرف آن اشتباه است.**
- **کلیه کلمات کلیدی باید با حروف کوچک نوشته شود وگرنه غلط است.**
- **نام متغیرها نباید از کلمات کلیدی انتخاب شود.**
- **کلیه متغیرهای مورد استفاده در یک برنامه باید در آغاز برنامه تعریف شوند وگرنه توسط کامپایلر غلط گرفته می‌شود.**
- **در تخصیص مقادیر به متغیرها چه مستقیم و چه در اثر انجام محاسبات، باید حدود تعریف‌شده را با توجه به نوع متغیر رعایت نمود و گرنه مقادیر به صورت ناقص ذخیره شده و نتایج غلط تولید خواهد گردید.**
- **در تخصیص ثابت‌های کاراکتری به متغیرهای صحیح یک بایتی باید دقت نمود که مثلاً ۹ با ’۹‘ تفاوت دارد، اگرچه هر دو می‌توانند در محاسبات شرکت کنند، اولی عدد ۹ است ولی دومی ثابت کاراکتری ۹ است که ارزش عددی آن در کد ASCII برابر ۵۷ می‌باشد.**
- **باتوجه به دستورهای تعریف نوع که تاکنون گفته شده است، متغیرهایی که تعریف شده‌اند ولی مقداری به آنها تخصیص داده نشده حاوی مقادیر نامشخصی هستند و استفاده از مقدار آنها در عبارات، باعث تولید نتایج ناخواسته خواهد گردید.**
- **عملوندهای عملگر % (باقیمانده) باید از نوع صحیح باشد وگرنه غلط گرفته می‌شود.**
- **اگر هر دو عملوند مورد استفاده در یک عمل تقسیم صحیح باشند، خارج قسمت صحیح محاسبه می‌شود.**
- **بعضی از عملگرها دارای اثر جانبی روی یک عملوند خود هستند یعنی مقدار آن‌را تغییر می‌دهند مثل عملگر تخصیص (=) که مقدار عملوند سمت راست را که نتیجه عمل است در عملوند سمت چپ قرار می‌دهد. در مورد این‌گونه عملگرها باید توجه داشت که عملوندی که اثر جانبی بر آن مترتب**

است نمی‌تواند یک مقدار ثابت، یک عبارت یا یک احضار تابع باشد بلکه باید یک عملوند قابل تغییر (lvalue) مثل اسم یک متغیر باشد که تغییر روی آن اعمال گردد.

- چون معمولاً هر برنامه‌ای نیاز به عمل ورودی و یا خروجی دارد در همه برنامه‌ها فایل سرآمد <stdio.h> باید اضافه شود مگر برنامه‌ای نوشته شود که ابداً عمل ورودی و خروجی انجام ندهد! اضافه کردن این فایل سرآمد با افزودن فرمان #include <stdio.h> در شروع فایل حاوی برنامه انجام می‌گردد. در غیراین صورت اگر از توابع ورودی و یا خروجی استفاده شود، دستور حاوی احضار آن تابع توسط کامپایلر C غلط گرفته می‌شود.
- برای استفاده از توابع ریاضی از قبیل جذر، توان، خطوط مثلثاتی و لگاریتم لازم است قبل از شروع برنامه فرمان #include <math.h> اضافه گردد وگرنه در صورت استفاده از توابع مزبور، روی دستور حاوی آن توسط کامپایلر C غلط گرفته می‌شود. این مطلب درمورد سایر توابع کتابخانه‌ای از قبل نوشته شده همراه کامپایلر C نیز صادق است.
- در احضار تابع printf تعداد مشخصه‌های تبدیل در رشته فرمت نباید بیشتر از تعداد مقادیر مورد چاپ باشد وگرنه مقادیر ناخواسته و غیرقابل استفاده‌ای چاپ خواهد شد.
- در احضار تابع printf نباید هیچ‌گونه فرضی در مورد ترتیب محاسبه عبارت‌هایی که مقادیرشان چاپ می‌گردد در نظر گرفت زیرا این موضوع روی کامپیوترهای مختلف فرق می کند.
- هر احضار تابع printf باید رشته فرمت داشته باشد و گرنه توسط کامپایلر C غلط گرفته می‌شود، ولی می‌تواند لیست مقادیر نداشته باشد و فقط پیغامی را چاپ نماید.
- وقتی رشته فرمت به صورت ثابت رشته‌ای داده می‌شود باید به طور کامل روی یک سطر قرار گیرد وگرنه توسط کامپایلر غلط گرفته می‌شود. این مطلب به طور کلی در مورد نوشتن ثابت‌های رشته‌ای باید رعایت گردد.
- نوع مشخصه‌های فرمت و نوع حاصل عبارت متناظر آن در احضار تابع printf باید با هم مطابقت داشته باشد وگرنه مقادیر چاپ شده غلط بوده و قابل استفاده نیست. مثلاً یک مقدار صحیح را نباید با مشخصه %f چاپ کرد وهمینطور نباید یک مقدار اعشاری را با مشخصه %d چاپ نمود.
- سطر اول برنامه که حاوی اسم آن است به علامت ; ختم نمی‌گردد و همینطور بعد از علامت { که پایان جملات بلوک برنامه یا سایر بلوک‌ها را مشخص می‌کند، نیازی به علامت ; نیست. سایر دستورهای برنامه باید به علامت ; ختم شوند.

- سطر اول برنامه که حاوی اسم آن است خارج از بلوک برنامه نوشته می‌شود و بلوک برنامه که حاوی کلیه جملات برنامه مگر سطر اول است، بلافاصله بعد از آن شروع می‌شود.
- اگر عملیات مورد نظر برای یک برنامه فقط توسط یک تابع انجام شود نام آن تابع باید `main` باشد و اگر توسط مجموعه‌ای از توابع انجام می‌شود، یکی و فقط یکی از آن توابع باید نامش `main` باشد و بقیه توابع به طور مستقیم یا غیرمستقیم توسط آن احضار شوند.

۳-۱۱ پرسش ها

- توصیه اکید در مورد انجام پرسشها پس از مطالعه و فهم برنامه های نمونه.

۳-۱۰- برنامه ای بنویسید که یک عدد صحیح و مثبت پنج رقمی (کوچک تر از ۳۲۷۶۷) در آن مشخص شود. نخست از روی آن دو عدد جدید بسازد که یکی حاوی رقمهای اول و سوم و پنجم عدد اولیه و دیگری حاوی رقمهای دوم و چهارم آن عدد باشد. سپس از روی این دو عدد یک عدد اعشاری بسازد که رقمهای صحیح آن از اولین عدد ساخته شده فوق و رقمهای اعشار آن از روی دومین عدد فوق گرفته شود. در پایان عدد اولیه را همراه با توضیح مناسب روی اولین سطر از صفحه، دو عدد صحیح ساخته شده را بعد از رها نمودن سه سطر خالی با فواصل مساوی روی یک سطر دیگر و عدد اعشاری نهایی را با توضیح مناسب در آخر همان سطر چاپ نماید (عرض سطر را ۸۰ ستون فرض کنید).

*۳-۱۱- برنامه ارائه شده در شکل ۳-۱۴ را دنبال کرده، نتیجه اجرا (تغییرات اعمال شده روی متغیرهای برنامه) و حاصل چاپ آن را به طور جداگانه مشخص نمایید. سپس برنامه را روی کامپیوتر اجرا کرده، پاسخ خود را با حاصل اجرا توسط کامپیوتر مقایسه کنید.

۳-۱۲ تکلیف شماره سه، مهلت یک هفته

انجام پرسشهای ۳-۱۰ و ۳-۱۱

```
#include <stdio.h>

main()
{
    int m, n, k, j, n۱, n۲;
   float x, y, z, t۱, t۲;
   char p, q;

   m=۵;
   n=۱۳;
   j=k=۲;
   x=۵.۰;
   y=۲.۵; z=۱۵;
   p='g'; q='h';
   printf("\fm=%d\tn=٪۱۰d\tm+n=٪۱۰.۵d\n" , m, n, m+n);
   k=(n+۴) / m;
   n۱=(n+۴) % m;
   n۲=n+۴ / m;
   printf("\n٪۲۰d٪۱۰d٪۸.۶d" , k, -n۱, n۲-۱۰۰);
   k=n + ۱۰ / m + ۱۰;
   n۱= n * ۱۰ / m * ۱۰;
   n۲= (m + ۲) / j;
   printf("%c % d %d\n", k * ۲, k= n۱+n۲, k * ۲);
   k=(m + ۲.۰) / j + (k = (x+۲) / j);
   z = (m+۲) / j;
```

شکل ۳-۱۴: متن برنامهٔ مربوط به پرسش ۳-۱۱.