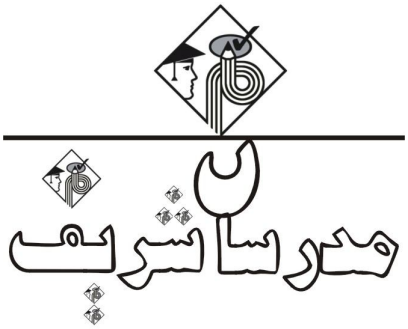


فصل اول

«تاریخچه زبان C»

تاریخچه زبان C

زبان C توسط دنیس ریچی در سال 1972 برای نوشتن سیستم عامل unix طراحی گردید. دنیس ریچی و تامپسون و هم‌چنین همکار آنها برایان کرینگان در سال 1973 سیستم عامل unix را بر روی سیستم DEC PDP 11/20 طراحی کردند. زبان C تنها از ابداعات ریچی نیست بلکه از زبان برنامه‌نویسی B تامپسون الهام گرفته است که خود زبان B نیز از روی زبان BCPL که در سال 1967 توسط مارتین ریچاردز ایجاد شده بود طراحی گردید.

هدف از طراحی زبان C آن بود که بوسیله آن سیستم عامل یونیکس طراحی شود. در آن زمان سیستم عامل‌ها به وسیله زبان اسمبلی طراحی می‌شد که از یک طرف برنامه‌نویسی با آن نسبتاً مشکل و هم‌چنین وقت زیادی صرف ایجاد و اشکال‌زدایی برنامه می‌شد همین امر سبب شد که شرکت بل با همکاری ریچی برای طراحی سیستم عامل خود زبان C را که از نظر قدرت تقریباً برابر اسمبلی بود و هم‌چنین برنامه‌نویسی با آن آسانتر بود را ایجاد کند. با به وجود آمدن زبان C و توانایی‌هایی که این زبان از خود نشان داد بزودی در بین تمام برنامه‌نویسان دنیا بسیار گسترش یافت و شرکت‌های زیادی اقدام به تولید نسخه‌های گوناگونی از این زبان به عمل آوردند که تا حدودی با هم اختلاف داشتند برای رفع این مشکل در سال 1983 برای زبان C یک استاندارد توسط مؤسسه استانداردهای ملی آمریکا (ANSI) بوجود آمده که به ANSI C معروف است. امروزه نسخه‌های متفاوتی از زبان C مانند Turbo C و Quick C و غیره وجود دارند که از استاندارد ANSI پیروی می‌کنند ما در این کتاب توجه خود را معطوف به ANSI C خواهیم کرد و در بعضی موارد به علت شباهت زیاد بین Turbo C و ANSI C مثال‌هایی از Turbo C نیز ارائه می‌دهیم روند تکامل زبان C را می‌توان در جدول زیر به طور خلاصه اینگونه بیان کرد.

نام زبان	سال ورودی
ALGOL	1960
CPL	1963
BCPL	1967
B	1970
C	1972
ANSI C	1983

ویژگی‌های زبان C

زبان C قابلیت برنامه‌نویسی بالا به پایین دارد. زبان C یک زبان ساخت یافته است در زبان‌های ساخت‌یافته سعی می‌شود از goto حدالامکان استفاده نشود. در این زبان می‌توان هر برنامه را به قسمت‌های مجزایی تقسیم نمود و سپس هر قسمت را توسط یک تابع نوشت و پس از تکمیل تمام قسمت‌ها می‌توان همه آنها را در یک تابع اصلی (main) برحسب نیاز فراخوانی کرد به این قابلیت برنامه‌نویس پیمانه‌ای(Modular Programming) یا برنامه‌نویس قسمتی(Compartmentalization) می‌گویند که هم باعث سهولت برنامه‌نویسی و آزمایش کردن برنامه می‌شود و هم‌چنین اشکال‌زدایی را راحت‌تر می‌کند. بعضی از زبان‌های ساخت یافته عبارتند از پاسکال، Algol،C و غیره ساخت‌یافته مثل فرترن ـ بیسیک ـ کوبول.

ـ قابلیت حمل (Portability)

زبان C قابل حمل است معنای قابلیت حمل این است که برنامه‌های به زبان C، که در یک نوع کامپیوتر نوشته می‌شوند، بدون انجام تغییرات یا انجام تغییرات اندک در کامپیوترهای دیگر قابل استفاده است. به عبارت دیگر می‌توان متن برنامه (Source Code) را روی انواع کامپیوترها با سخت‌افزارهای متفاوت و سیستم عامل‌ها کامپایل کرده و برنامه را روی آن سیستم اجرا کرد.

ـ قابلیت انعطاف (Flexibility)

زبان C قابل انعطاف و بسیار قدرتمند است در این زبان هیچ محدودیتی برای برنامه‌نویس وجود ندارد.

ـ کم بودن تعداد کلمات کلیدی

زبان C زبان کوچکی است تعداد کلمات کلیدی این زبان حدود 32 کلمه است برخلاف تصور بعضی‌ها که فکر می‌کنند هر چه تعداد کلمات کلیدی زبان بیشتر باشد آن زبان قدرتمندتر است به عنوان مثال زبان بیسیک 150 کلمه کلیدی دارد ولی قدرت زبان C به مراتب بیشتر از بیسیک است. توجه داشته باشید که بعضی از کامپایلرهای C علاوه بر 32 کلمه کلیدی استاندارد، کلمات دیگری به این زبان افزوده‌اند.

ـ ارتباط با زبان اسمبلی

ارتباط تنگاتنگی بین زبان C و زبان اسمبلی وجود دارد بدین ترتیب که می‌توان از قابلیت‌های اسمبلی در C استفاده کرد.

ـ هدف از طراحی C

زبان C یک زبان همه‌منظوره (General Purpose) است و برای کلیه موارد تجاری، بانکی، سیستمی، گرافیک و مهندسی و غیره قابل استفاده است گرچه بیشتر قدرت زبان C در نوشتن برنامه‌های سیستمی مثل سیستم‌عامل‌ها و کامپایلرها می‌باشد.

ـ زبان C کامپایلری می‌باشد!

زبان C به صورت کامپایلری است برعکس بیسیک که مفسری است، در زبان‌های برنامه‌نویسی مفسری هر خط ابتدا خوانده می‌شود و اگر خطای گرامری در آن وجود نداشته باشد تبدیل به زبان ماشین شده و اجرا می‌گردد ولی در زبان‌های کامپایلری کل برنامه خوانده می‌شود و اگر خطای گرامری در برنامه وجود نداشته باشد تبدیل به کد ماشین می‌شود و بعداً اجرا می‌گردد به عبارت دیگر در کامپایلر یک بار برنامه کامپایل شده و در نهایت به کد قابل اجرا (Executable File) تبدیل می‌شود و پس از آن بارها می‌تواند اجرا شود ولی در مفسری با هر بار اجرای برنامه هر خط برنامه خوانده و به کد ماشین ترجمه و اجرا می‌شود.

ـ C یک زبان میانی

زبان‌های برنامه‌نویسی را می‌توان به ۳ دسته تقسیم کرد:

۱ـ زبان‌های سطح بالا که به زبان عادی شباهت دارد.

۲ـ زبان‌های سطح میانی.

۳ـ زبان‌های سطح پایین که به زبان ماشین شباهت دارند.

علت میانی بودن زبان C این است که از طرفی همانند زبان سطح پایین اسمبلی قادر است مستقیماً به حافظه دستیابی پیدا کند و با مفاهیم بیت و بایت کار کند و از طرف دیگر برنامه‌های این زبان هم‌چون زبان‌های سطح بالایی مانند پاسکال از قابلیت خوانایی بالایی برخوردارند.

زبان C یک زبان حساس به حروف (Case Sensitive) است یعنی زبان C بین حروف بزرگ و کوچک تفاوت قائل است برعکس پاسکال مثلاً عبارت SUM و Sum و sum در C با هم متفاوت است.

انواع خطاها در زبان C

۱ـ خطای زمان کامپایل (Compile Time Error)

خطاهای زبان کامپایل یا خطاهای syntax به علت خطای گرامری موجود در دستورات می‌باشد و تا زمانی که برنامه خطای گرامری داشته باشد تبدیل به کد ماشین نمی‌شود.

۲ـ خطای زمان اجرا (Run Time Error)

خطای زمان اجرا به خطایی اطلاق می‌شود که در هنگام اجرای برنامه به وقوع بپیوندد مثلاً اگر در هنگام اجرای برنامه دستور تقسیم عدد بر صفر انجام شود با خطای تقسیم بر صفر Division by Zero مواجه می‌شویم.

۳ـ خطای منطقی (Logical Error)

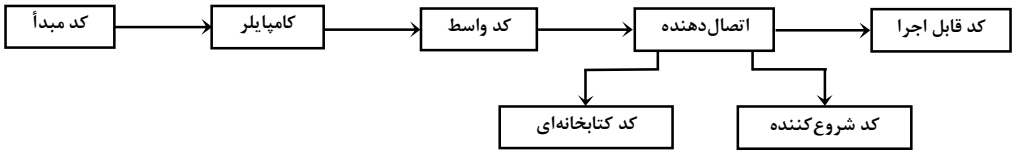
در این نوع خطا در هنگام کامپایل و اجرا شدن برنامه پیامی صادر نمی‌شود ولی آن جوابی که موردنظر شما است و از برنامه انتظار داریم به دست نمی‌آید به این خطاها، خطای مفهومی نیز گفته می‌شود مثلاً فرض کنید که در یک خط برنامه قرار بود عبارت y=10+5; نوشته شود ولی به طور اشتباه y=10-5; تایپ شده است. بدیهی است که این خطوط از نظر گرامری درست هستند ولی ازنظر مفهوم با یکدیگر تفاوت دارند.

مراحل ساخت برنامه در زبان C

پس از نوشتن برنامه در محیط ویرایشگر C برای این که بتوانیم برنامه‌ای قابل اجرا به دست آوریم باید آن را به یک فایل اجرایی (Executable File) تبدیل کنیم از آنجایی که C زبانی کامپایلری می‌باشد مراحل تولید فایل اجرایی بدین طریق انجام می‌شود که ابتدا برنامه

را در یک ویراستار C نوشته و سپس آن را کامپایل می‌کنیم، کامپایلر C برنامه را ترجمه کرده و در صورتی که برنامه از نظر دستوری (Syntax) درست باشد فایل واسط (Object) با پسوند obj را تولید می‌کند. درون این فایل ماشین کدهای برنامه قرار دارد. سپس اتصال‌دهنده (linker) فایل با پسوند obj را گرفته و آن را با کدهای دیگر که شامل توابع کتابخانه‌ای و کدهای شروع‌کننده می‌باشد به منظور ایجاد فایل اجرایی ترکیب کرده و فایل قابل اجرا (Exe) را به وجود می‌آورد.

فایل obj و exe هر دو شامل ماشین کد برنامه می‌باشند با این تفاوت که فایل exe علاوه بر ماشین کد خود برنامه، ماشین کد توابع به کار رفته در برنامه و هم‌چنین کدهای شروع‌کننده که بین برنامه و سیستم عامل عمل می‌کنند را شامل می‌باشد با یک مثال موضوع فوق روشن‌تر می‌شود. مثلاً اگر در فایل منبع مثلاً prog.c از دستور clrscr استفاده کرده باشیم فایل prog.obj دارای ماشین کد clrscr نخواهد بود بلکه تنها شامل روتین‌هایی برای فراخوانی تابع clrscr می‌باشد ماشین کد مربوط به clrscr در یکی از کتابخانه‌های C موجود می‌باشد که توسط اتصال‌دهنده (Linker) به فایل obj افزوده شده و نهایتاً فایل exe را می‌سازد لازم به ذکر است امروزه برخلاف گذشته کامپایلرها و اتصال‌دهنده‌ها از هم مجزا نیستند و در یک واحد پیاده‌سازی شده‌اند.



ساختار یک برنامه در C

ساختار کلی یک برنامه ساده در زبان C را با یک مثال شرح می‌دهیم:

مثال ۱: برنامه‌ای که یک عدد صحیح گرفته و آن را به توان 2 می‌رساند.

پاسخ:

```
# include < stdio.h>
# include <conio.h>
int main ( )
{
  int i,j;
  printf ("enter a number:");
  scanf ("%d", & i);
  j=i*i;
  printf ("%d",j);
  return 0;
}
```

برنامه‌های زبان C، از مجموعه‌ای از دستورات و تعدادی توابع تشکیل می‌شود هر تابع دارای یک اسم منحصر به فرد است مثل scanf که برای گرفتن ورودی از صفحه‌کلید استفاده می‌شود. بدنه اصلی هر برنامه به زبان C شامل تابعی به نام main می‌باشد.

نکته ۱: هر برنامه به زبان C حداقل از یک تابع تشکیل شده است (تابع main)

نکته ۲: هر برنامه C از تابع با نام main شروع می‌شود.

علاوه بر main توابع دیگری از قبل نوشته شده‌اند و همراه کامپایلر زبان C ارائه می‌گردد. مثل scanf، بنابراین قبل از پرداختن به برنامه‌نویسی باید اطلاع داشته باشیم که این توابع در کجا قرار دارند و چگونه می‌توان از آنها در برنامه استفاده کرد. این توابع در تعدادی از فایل‌ها به نام فایل‌های هدر قرار دارند. هدر فایل‌ها، فایل‌های متنی بوده و معرفی توابع و ثابت‌ها در آن نوشته شده است. پسوند این فایل‌ها h. می‌باشند و معمولاً در شاخه Tc\ Include قرار دارند برای استفاده کردن از توابعی که درون این فایل‌های هدر قرار دارند باید فایل هدری که آن توابع در آن قرار دارند را از طریق دستور #include به کار ببریم.

نکته ۳: دستوراتی که با علامت # شروع می‌شوند دستورات پیش‌پردازنده می‌باشند، پیش‌پردازنده مترجمی است که با مشاهده دستوراتی که با # شروع می‌شوند اجرا می‌شود و آن را به دستورات زبان C تبدیل می‌کند.

نکته ۴: توجه داشته باشید که در انتهای دستورات پیش‌پردازنده ; قرار نمی‌گیرد صورت کلی استفاده از دستور #include می‌تواند به یکی از اشکال زیر باشد:

```
# include <نام فایل هدر>
# include "نام فایل هدر"
```

تفاوت بین فرم اول و دوم این است که اگر نام فایل هدر در بین علامت < > باشد، فقط مسیرهای تعیین شده در منوی directories option→ بررسی می‌گردد.
معمولاً رسم است آن دسته از فایل‌های هدر را که خود برنامه‌نویس خودش ایجاد کرده داخل علامت " " قرار دهد و فایل‌های هدر خود سیستم را در علامت < > قرار دهد.

نکته ۵: برای اضافه کردن هر فایل هدر به برنامه باید از یک دستور #include استفاده کرد یعنی مثلاً برای اضافه کردن دو فایل هدر stdio.h , conio.h به برنامه بدین شکل عمل می‌کنیم.
include <stdio.h>
include <conio.h>

نکته ۶: بین # و include نباید فاصله‌ای وجود داشته باشد.

نکته ۷: در دستور #include می‌توان آدرس محلی که فایل هدر در آن قرار دارد را نوشت مثلاً می‌توانیم بنویسیم:
#include "a:\tc\include\my-header.h"
در زبان C استاندارد، توضیحات (comment) برنامه داخل دوعلامت /*,*/ نوشته می‌شود توضیحات می‌تواند شامل یک خط یا چند خط باشند علامت توضیحات می‌تواند در ابتدای یک خط و یا حتی بعد از یک دستور یا تعریف یک متغیر باشد.

```
/*this is a comment*/
int i; /*this is a comment*/
```

در Turbo C به غیر از علامت فوق می‌توان از نماد // نیز برای ارائه توضیحات استفاده کرد تفاوت این دو در این است که با علامت // فقط می‌توان در یک خط، توضیحات ارائه کنید در صورتی که بخواهیم با علامت // چند خطا را به عنوان توضیحات درج کنیم باید برای هر خط علامت // را به کار ببریم. ولی با نماد / * *// می توانیم هر تعدادخط را به طور یکجا یعنی با به کاربردن یکبار علامت *// به عنوان شروع توضیحات و علامت / * به عنوان پایان توضیحات را به عنوان توضیحات مشخص کنیم.
دستورالعمل‌های برنامه با علامت } شروع و با علامت { پایان می‌یابد ولی به طور کلی می‌توان گفت تعریف تمام توابع و بلوک‌ها در C مانند پاسکال که بین Begin و End قرار می‌گیرد، در C بین دوعلامت } و { به ترتیب به عنوان شروع و پایان بلوک یا تابع بکار می‌رود. مثلاً اگر دستور if بیشتر از یک دستور داشته باشد تشکیل یک بلاک می‌دهد که باید برای این که دستورات به طور صحیح انجام گیرند بین دوعلامت } { قرار گیرد.

```
if (i>5)
{
  i=i*2;
  i=i*3;
  i+ +;
}
```

نکته ۸: در C آخرین دستور یعنی علامت { که به عنوان پایان بلاک main محسوب می‌شود علامت سمی‌کولن (;) ندارد البته در صورتی که ; هم قرار گیرد کامپایلر C خطایی اعلام نمی‌کند در زبان C می‌بایست متغیرها را قبل از استفاده تعریف کرد.

مثال ۲: طرز تعریف کردن دو متغیر از نوع صحیح می‌تواند به شکل زیر باشد.

پاسخ:

```
int i;
int j;
```

البته دو متغیر را می‌توان در یک خط هم تعریف کرد مثلاً int i,j;

نکته ۹: طول هر خط برنامه در C می‌تواند حداکثر 255 کاراکتر باشد.

نکته ۱۰: در هر سطر برنامه در C می‌توان چند دستور نوشت.

مثال ۳:

```
main ( ) {int i,j;}
```

از تابع printf (print format) برای چاپ خروجی استاندارد استفاده می‌شود. الگوی تابع printf در فایل stdio.h قرار دارد. شکل دستور printf به صورت زیر است.
printf ("<عبارت اول>" و < عبارت دوم >);
که در عبارت اول هر چیزی بین دو علامت " " عیناً در خروجی ظاهر می‌شود و همچنین نوع اطلاعاتی که توسط تابع printf چاپ می‌شود را مشخص می‌کند.
این کدها با علامت ٪ شروع می‌شود و به آنها کدهای تعیین‌کننده فرمت خروجی می‌گویند مثلاً %d برای چاپ عدد صحیح استفاده می‌شود. در

قسمت دوم printf نیز نام متغیرهایی که می‌خواهیم چاپ شود مشخص می‌شود.

مثال ۴: برای این که بتوانیم مقدار متغیر x که برابر 10 است را چاپ کنیم به صورت زیر عمل می‌کنیم.

پاسخ:

```
int x=10;
printf ("the value of x is %d",x);
```

هنگام چاپ کردن، عبارت the value of x is دقیقاً در خروجی چاپ می‌شود و بجای %d که جزو کاراکترهای تعیین کننده فرمت خروجی می‌باشد مقدار x قرار می‌گیرد.

می‌توانیم قسمت دوم printf را نیز حذف کنیم و فقط قسمت اول را بنویسیم در این صورت هر عبارتی که بین علامت " " وجود داشته باشد عیناً در خروجی چاپ می‌شود مثلاً printf ("this is a test"); که در خروجی عبارت this is a test چاپ می‌شود.

از تابع scanf (scan format) که الگوی آن در فایل stdio.h قرار دارد برای ورود اطلاعات از صفحه کلید مورد استفاده قرار می‌گیرد شکل دستور scanf به صورت زیر می‌باشد:

```
scanf ("<عبارت اول>" و "< عبارت دوم >");
```

<عبارت۱> مشخص می‌کند که مقادیر ورودی چگونه باید خوانده شود به کاراکترهایی که در این مکان قرار می‌گیرد کاراکتر فرمت می‌گویند. کاراکترهای فرمت با علامت ٪ شروع می‌شود مثلاً %d برای ورود اعداد صحیح استفاده می‌شود و در قسمت دوم آدرس متغیرهایی که مقدار ورودی در آنها قرار می‌گیرد را مشخص می‌کند به جز رشته‌ها که علت آن را در فصل اشاره‌گرها بیان خواهیم کرد. برای بدست آوردن آدرس متغیرها قبل از نام متغیر از علامت & استفاده می‌کنیم.

مثال ۵: برای این که بتوانیم یک متغیر صحیح از ورودی بگیریم تابع scanf را به صورت زیر به کار می‌بریم.

پاسخ:

```
int i;
scanf ("%d",&i);
```

لازم به ذکر است بررسی دقیق توابع ورودی خروجی در فصول آینده انجام می‌شود در این قسمت فقط برای آشنایی با شکل ساده یک برنامه C این نکات ذکر گردید.

اگر چند برنامه را اجرا کنیم و هر برنامه خروجی خودش را روی صفحه نمایش چاپ کند بعد از مدتی دیدن خروجی مشکل خواهد بود برای این که بتوانیم صفحه نمایش را پاک کنیم از تابع clrscr که در فایل هدر conio.h وجود دارد استفاده می‌کنیم.

نکته دیگری که درباره برنامه به زبان C باید بدانیم این است که برای این که به سیستم عامل اجرا کننده تابع main اطلاع دهیم که برنامه با موفقیت به پایان رسیده معمولاً آخرین دستور برنامه C عبارت return 0; است این دستور مقدار صفر را به سیستم عامل برمی‌گرداند و سیستم عامل متوجه می‌شود برنامه با موفقیت به پایان رسیده است. البته بکار بردن دستور فوق در بیشتر کامپایلرها از جمله Turbo C اختیاری می‌باشد.

مقداری که به سیستم عامل برگشت داده می‌شود از نوع int است لذا تابع main که می‌خواهد مقداری از نوع int را برگرداند، خودش باید از نوع int باشد به همین دلیل تابع main را به صورت زیر در برنامه به کار می‌بریم. int main ( )

نکته ۱۱: اگر عبارت int را قبل از main ننویسیم باز هم خروجی از نوع int درنظر گرفته خواهد شد.

نکته ۱۲: در صورتی که نخواهیم تابع main مقداری به سیستم عامل برگرداند قبل از main عبارت void را به کار می‌بریم و دستور return 0; را دیگر نمی‌نویسیم.

کلمات کلیدی زبان C

در زبان C کلماتی وجود دارند که برای زبان C رزرو شده‌اند لذا برنامه‌نویس نمی تواند از این کلمات برای نامگذاری شناسه‌های خود اعم از متغیر، ثابت، توابع و غیره استفاده کند. کلمات رزرو شده که توسط ANSI C تعریف شده‌اند شامل جدول زیر است.

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

در تورب C کلمات کلیدی زیر به لیست فوق اضافه شده است.

-es	-ds	-cs	asm
huge	far	cdecl	-ss
pascal	near	interrupt	

تعریف ثابت با پیش‌پردازنده #define (ثابت حقیقی)

ماکرو نامی برای یک رشته است که این رشته می‌تواند ترکیبی از حروف، ارقام، مقادیر ثابت، توابع و غیره باشد.
دستور پیش‌پردازنده #define برای تعریف ماکرو استفاده می‌شود این دستور به دو شکل به کار می‌رود که هر کدام را با ارائه مثال شرح می‌دهیم نام ماکرو همانند نام یک متغیر در C است که با دستور #define حداقل باید یک فاصله داشته باشد و معمولاً رسم است به خاطر این که با متغیرهای معمولی اشتباه گرفته نشود، آن را با حروف بزرگ انتخاب می‌کنند مثلاً عبارت #define TRUE 1 سبب می‌شود در هر کجای برنامه که از ماکروی TRUE استفاده شده عدد یک قرار بگیرد. شکل کلی تعریف ماکرو (ثابت حقیقی) در حالت اول به شکل زیر است.

#define <نام ماکرو> <رشته>

مثال ۶:

```
#include <stdio.h>
main ( )
{ #define TRUE 1
printf ("%d", TRUE);
}
```

که در هنگام کامپایل کردن، برنامه پیش پردازنده ابتدا بجای عبارت TRUE مقدار آن یعنی یک را قرار داده و در خروجی عدد یک چاپ می‌شود.

نکته ۱۳: می‌توان در هنگام تعریف ثابت مجازی (ماکرو) بجای رشته، عدد، اسم یک تابع و یا هر علامت ویژه را قرار داد.

مثال ۷:

```
#include <stdio.h>
#define program main ( )
#define begin {
#define end. }
#define write_msg printf ("test");
program
begin
        write_msg
end.
```

با اجرا شدن برنامه به جای هر ثابت تعریف شده مقدار آن قرار می‌گیرد و در نتیجه عبارت test روی صفحه نمایش چاپ می‌شود به طور کلی ماکرو در زبان‌های برنامه‌نویسی شبیه توابع هستند یعنی یک بار تعریف شده و چند بار ممکن است از آنها استفاده شود (فراخوانی شود) فرق اصلی ماکرو با توابع در این است که در هنگام فراخوانی ماکرو، دستورات ماکرو (بدنه ماکرو) عیناً در محل فراخوانی ماکرو، تکرار می‌شوند (کپی می‌شوند) یعنی اگر در یک برنامه چند بار از ماکرو استفاده شود در هر بار فراخوانی ماکرو، کد ماکرو در محل فراخوانی ماکرو عیناً تکرار می‌شود و سبب می‌شود که حجم برنامه افزایش یابد ولی در هنگام فراخوانی تابع در هر کجای برنامه، کنترل برنامه به ابتدای تعریف تابع منتقل می‌شود و پس از اجرای تابع به خط بعد از فراخوانی تابع منتقل می‌شود.
کاربرد دیگر دستور #define در تعریف ماکرویی است که دارای پارامتر می‌باشند. در این مورد از دستور #define به صورت زیر استفاده می‌شود.

#define تعریف ماکرو (اسامی پارامترها) نام ماکرو

مثال ۸: برنامه‌ای که با استفاده از ماکرو دو عدد را در هم ضرب می‌کند.

پاسخ:

```
#include <stdio.h>
#define MULTIPLY (a,b)  a*b
void main ( )
{
   int i,j;
   scanf ("%d%d",&i,&j);
   printf ("%d",MULTIPLY (i,j));
}
```

در هنگام چاپ به جای عبارت MULTIPLY (i,j) عبارت معادل آن یعنی i*j قرار گرفته و مقدار حاصل چاپ می‌شود.

نکته ۱۴: در تعریف یک ماکرو جدید می‌توان از یک ماکروی تعریف شده دیگر استفاده کرد.

```
#define    BLUE          1
#define    BLUE-BLINK    BLUE+128
```

مثال ۹: حال در هر قسمت از برنامه که ماکرو BLUE-BLINK قرار بگیرد. عبارت 1+128 به جای آن جایگزین می‌شود.

نکته ۱۵: اگر رشته جلوی #define بیش از یک خط باشد در انتهای خط کاراکتر '\' قرار داده می‌شود تا معلوم شود متن خط بعد ادامه همین خط است.

```
#define "this is an error\
        Occur during compile"
```

نکته ۱۶: با دستور #define به یک ثابت یک نام مجازی اختصاص می‌دهیم در نتیجه هیچ حافظه‌ای در دستور #define به اسم مجازی تخصیص داده نمی‌شود.

نکته ۱۷: می‌توانیم دستور #define را در هر کجای برنامه که می‌خواهیم بنویسیم ولی معمولاً رسم است که در ابتدای برنامه نوشته شود.

نکته ۱۸: در انتهای دستور #define به علت اینکه یکی از دستورات پیش‌پردازنده می‌باشد ; قرار نمی‌گیرد. البته در صورتی که قرار بگیرد جزئی از ثابت تعریف شده درنظر گرفته می‌شود و خطایی اعلام نمی‌شود.

نکته ۱۹: چون با فراخوانی ماکرو کد مربوط به آن در محل فراخوانی کپی می‌شود بهتر است اگر تعریف ماکرو بیش از دو و سه خط باشد به جای آن تابع معادل آن ماکرو را بنویسیم.

حذف ماکروی تعریف شده

اگر در برنامه‌ای، ماکرویی توسط دستور #define تعریف گردد دستور #undef از یک نقطه دلخواه (از نقطه‌ای که این دستور قرار می‌گیرد) به بعد، تعریف ماکرو را بی‌اثر می‌کند این دستور به شکل زیر بکار می‌رود.

```
#undef  <نام ماکرو>
```

تعریف ماکرویی که نام آن در دستور #undef آمده است از جایی که این دستور در برنامه ظاهر می‌گردد، به بعد منتفی می‌شود.

مثال ۱۰:

```
#include <stdio.h>
#define  MSG        "computer science"
main ( )
{
  printf (MSG);
#undef MSG
  printf (MSG);
}
```

☑ **پاسخ:** چون با عبارت #undef MSG تعریف MSG منتفی شده و ما بعد از آن سعی در چاپ کردن MSG داریم با خطا روبرو می‌شود در نتیجه برنامه اجرا نخواهد شد.

کلیدهای مهم در محیط ویرایشگر C

F1: برای نمایش اطلاعات راهنما (Help) مورد استفاده قرار می‌گیرد.

Ctrl+F1: فشردن این دو کلید زمانی که مکان‌نما زیر دستوری قرار داشته باشد راهنمای مربوط به آن دستور، روی صفحه ظاهر می‌شود.

F2: محتوی فایل جاری را ذخیره می‌کند. (Save)

F3: امکان انتخاب یک فایل از لیست فایل‌ها را برای باز کردن و نمایش آن به کاربر می‌دهد.

F4: از ابتدای برنامه تا خط جاری را اجرا می‌کند.

F5: برای تغییر اندازه پنجره جاری استفاده می‌شود.

F6: برای حرکت بین پنجره‌های باز شده.

F7: برنامه راخط به خط اجرا می‌کند در صورتی که خط حاوی فراخوانی تابع باشد کنترل به ابتدای خط شروع تابع منتقل می‌شود.

F8: برنامه را خط به خط اجرا می‌کند در صورتی که خط حاوی فراخوانی تابع باشد تابع یکباره اجرا می‌شود.

F9: فایل اجرایی (exe) برنامه را ایجاد می‌کند.

F10: منوی ویرایشگر C را فعال می‌کند.

Alt+F3: پنجره جاری را می‌بندد.

Alt+F5: می توان صفحه خروجی برنامه را مشاهده کرد.

Alt+F9: برنامه موجود در پنجره جاری ویرایشگر C را ترجمه و فایل با پسوند obj آن را تولید می‌کند.

Ctrl+F5:برای تغییر محل پنجره جاری با کلیدهای جهت‌دار و یا تغییر اندازه پنجره جاری با فشردن کلید Shift توأم با کلیدهای جهت‌دار.

Alt+1..9: هر پنجره باز شده روی محیط کار دارای شماره‌ای است که می‌توان با فشرد کلید Alt به همراه شماره آن پنجره، پنجره موردنظر را به عنوان پنجره جاری قرار داد.

Ctrl+F8: با فشردن این کلید می‌توان محتوای یک متغیر را هنگام اجرای برنامه به صورت دستی (توسط کلید F8 یا F7) مشاهده کرد.

Ctrl+F9: محتوای برنامه جاری را اجرا می‌کند. در صورتی که برنامه جاری از قبل ترجمه نشده باشد و یا پس از آخرین عمل ترجمه تغییر کرده باشد عمل ترجمه را نیز قبل از اجرا انجام می‌دهد.

تست‌های طبقه‌بندی شده فصل اول

۱ـ در زبان C، برنامه الزاماً با تابع ..... شروع می‌شود. (آزاد ـ ۷۵)

۱) scanf ۲) printf ۳) main ۴) getche

۲ـ در زبان برنامه‌نویسی C، شروع برنامه با کدام تابع زیر آغاز می‌شود؟ (آزاد ـ ۷۵)

۱) int ۲) # printe ۳) main ۴) {

۳ـ در خصوص کلمات کلیدی و شناسه‌های مجاز، کدام گزینه صحیح می‌باشد؟ (آزاد ـ ۷۵)

۱) getche main for while_ num ۲) _num 39 HP do 2zero scanf else
۳) – B00d K3A5 ali if printf int ۴) c printf getch mt.s double seiteh

۴ـ کدام جمله در مورد کلاس‌های حافظه در زبان برنامه‌نویسی C، درست است؟ (آزاد ـ ۸۴)

۱) کلاس حافظه اتوماتیک (auto)، کامپایلر را مجبور می‌کند که یک متغیر را به شکل اتوماتیک ساخته و در کل مدت اجرای برنامه آنرا حفظ کند و به محل تعریف متغیر ارتباطی ندارد.
۲) کلاس حافظه ثبات (register)، به کامپایلر پیشنهاد می‌کند که متغیر از نوع اتوماتیک را درون رجیستر پردازنده قرار دهد.
۳) کلاس حافظه ثبات (register)، به کامپایلر می‌گوید تا متغیری بسازد که فقط یک مرتبه مقدار اولیه می‌گیرد و همواره آخرین مقدار خود را نگه می‌دارد.
۴) در زبان برنامه نویسی C، اصلاً کلاس حافظه وجود ندارد.

۵ـ کدام گزینه صحیح است؟ (آزاد ـ ۸۴)

۱) زبان برنامه نویسی c مانند زبان پاسکال حساس به حروف (case sensitive) است.
۲) قابل حمل بودن، یعنی پس از کمپایل‌کردن یک برنامه، می‌توان آن برنامه را بدون تغییر روی هر کامپیوتر و سیستم عامل دیگری اجرا نمود.
۳) تعداد کلمات کلیدی (keyword) ارتباطی به قدرت یک زبان برنامه نویسی ندارد.
۴) در زبان برنامه نویسی C، امکان دسترسی و بهره‌برداری از سخت‌افزار وجود ندارد.

۶ـ علامت دستورات پیش‌پردازنده در زبان C کدام یک از علایم زیر است؟ (مؤلف)

۱) $ ۲) # ۳) @ ۴) &

۷ـ در زبان C، برنامه‌ها از کجا شروع به اجرا می‌شود؟ (مؤلف)

۱) از هر تابعی می‌تواند ۲) main ( ) ۳) #include ۴) {

۸ـ کدام جمله درباره ۲ دستور #include <a.h> و #include "a.h" درست است؟ (مؤلف)

۱) این دو دستور هیچ فرقی با هم ندارند.
۲) دستور #include <a.h> ابتدا فهرست جاری را می‌گردد.
۳) دستور #include "a.h" ابتدا مسیر مشخص شده در مسیر options → directories و سپس فهرست جاری را می‌گردد.
۴) دستور #include <a.h> فقط مسیر options → directories را جستجو می‌کند.

۹ـ کدام زبان از نوع غیر ساخت‌یافته است؟ (مؤلف)

۱) کوبول ۲) C ۳) پاسکال ۴) alogl

۱۰ـ چه نوعی از فایل‌های C قابلیت حمل دارند؟ (ازنظر پسوند) (مؤلف)

۱) .exe ۲) .obj ۳) .c ۴) .pas

۱۱ـ کدام گزینه غلط است؟ (مؤلف)

۱) در هر سطر می‌توان چندین دستور نوشت.
۲) C یک زبان ساخت یافته است.
۳) حروف کوچک و بزرگ در C فرقی ندارد.
۴) C یک زبان میانی است.

۱۲ـ بلوک‌های یک برنامه به زبان C با چه علامتی شروع و به پایان می‌رسند؟ (مؤلف)

۱) (,{ ۲) [,] ۳) -(,) ۴) },{

۱۳ـ کدام گزینه یک گروه از واژه‌های کلیدی C را نشان نمی‌دهد؟ (مؤلف)

۱) char, void , float , double
۲) struct , for , goto , do
۳) else , enum , break , auto
۴) sizeof , file , case , val

۱۴ـ حاصل اجرای تکه برنامه زیر چیست؟ (مؤلف)

```
#define str printf ("one");
main ( )
{
#define str printf ("two");
str
}
```

۱) خطا دارد زیرا انتهای str سمی‌کولن ندارد.
۲) خطا دارد زیرا دو شناسه با نام str نمی‌توان در برنامه تعریف کرد.
۳) عبارت one چاپ می‌شود.
۴) عبارت two چاپ می‌شود.

۱۵ـ خروجی برنامه زیر چیست؟ (مؤلف)

```
#include <stdio.h>
#define A(a,b)   (a+b)*5+a*b
main ( )
{
   printf ("%d", A(2,3));
}
```

۱) 81
۲) 31
۳) 55
۴) در مقابل define عبارت محاسباتی نمی‌توان به کار برد.

۱۶ـ کدام گزینه درباره توضیحات (comment) درست است؟ (مؤلف)

۱) با علامت \\ می‌توان توضیحاتی در چند خط ارائه کرد.
۲) با علامت /* */ می‌توان توضیحاتی در چند خط ارائه کرد.
۳) با علامت * *\ می‌توان توضیحاتی در چند خط ارئه نمود.
۴) با علامت // می‌توان توضیحاتی در چند خط ارائه نمود.

۱۷ـ حاصل اجرای قطعه برنامه زیر چیست؟ (مؤلف)

```
#include <stdio.h>
#define Boolean "#define True 1"
main ( )
{
printf (Boolean);
}
```

۱) خطا دارد زیرا دو دستور define# را نمی‌توان در یک خط تعریف کرد.
۲) True چاپ می‌شود.
۳) "#define True1" چاپ می‌شود.
۴) #define True 1 چاپ می‌شود.

۱۸ـ کدام یک از گزینه‌های زیر پیام Hello World را چاپ می‌کند؟ (مؤلف)

```
#define msg "Hello world"
```

۱) printf (msg);
۲) printf ("msg");
۳) printf("%s",msg);
۴) گزینه ۱ و ۳

۱۹ـ حاصل اجرای قطعه‌برنامه زیر چیست؟ (مؤلف)

```
char ch;
#define massage printf ("Enter character"); scanf ("%c", &ch);
main ( )
{
massage
}
```

۱) پیغام Enter character ظاهر شده و منتظر وارد کردن مقداری برای متغیر ch می‌ماند.
۲) تعریف مقابل define# خطا است. زیرا مقابل دستور define# نمی‌توان از دستورات C استفاده کرد.
۳) چون انتهای دستور massage سمی‌کولن ندارد خطا دارد.
۴) خطا می‌دهد زیرا مقابل دستور define# می‌توان یک دستور تعریف کرد.

۲۰ـ فایل هدر دارای چه محتوایی است؟ (مؤلف)

۱) فایل‌های obj که حاوی کتابخانه‌های داخلی C می‌باشند.
۲) فایل‌های اجرایی می‌باشند.
۳) فایل‌های متنی بوده و کد توابع در آنها نوشته شده است.
۴) فایل‌های متنی هستند که معرفی توابع، روال‌ها و ثابت‌های برنامه در آن قرار دارد.

۲۱ـ قطعه برنامه زیر دارای کدام نوع خطا است؟ (مؤلف)

```
#define i 0
int j,n;
   n=j/i;
```

۱) خطای منطقی
۲) خطای زمان کامپایل
۳) خطای زمان اجرا
۴) هیچکدام

۲۲ـ خروجی تکه برنامه زیر کدام است؟ (مؤلف)

```
#define  f(x,y)x*y+x
int i=5 , j=3;
printf ("%d" , 2*f(i,j));
```

۱) 35
۲) 40
۳) 30
۴) 50

۲۳ـ کدام تعریف زیر خطا دارد؟ (مؤلف)

۱) int If;
۲) int For;
۳) int do;
۴) int printf;

۲۴ـ کدام تعریف زیر خطا دارد؟ (مؤلف)

۱) int I;
۲) Int i;
۳) int Int;
۴) int INT;

۲۵ـ کاربرد عملگر # چیست؟ (مؤلف)

۱) برای استفاده از کتابخانه
۲) برای تعریف متغیر
۳) برای تعریف ماکرو
۴) گزینه‌های ۱ و ۳

پاسخنامه تشریحی فصل اول

۱ـ گزینه «۳» در زبان C برنامه با تابع main شروع می‌شود.

۲ـ گزینه «۳» در زبان C برنامه با تابع main شروع می‌شود.

۳ـ گزینه «۱» در کلمات کلیدی و شناسه‌های مجاز اعداد و علائم نمی‌توان استفاده شود.

۴ـ گزینه «۲» کلاس حافظه ثبات به کامپایلر پیشنهاد می‌کند که متغیر از نوع اتوماتیک را درون ثبات قرار دهد.

۵ـ گزینه «۳»

۶ـ گزینه «۲» دستورات پیش پردازنده در زبان C با علامت # شروع می‌شوند.

۷ـ گزینه «۲» تمام برنامه‌های زبان C از تابع با نام ( ) main شروع می‌شوند.

۸ـ گزینه «۴» دستور #include <a.h> فقط مسیر مشخص شده در قسمت options → directories را جستجو می‌کند.

۹ـ گزینه «۱» زبان‌های کوبول، فرترن و بیسیک غیرساخت‌یافته هستند.

۱۰ـ گزینه «۳» فایل‌های سورس c که با پسوند c. هستند قابل حمل هستند بقیه فایل‌ها از قبیل exe و obj از سخت‌افزاری به سخت‌افزار دیگر متفاوت است.

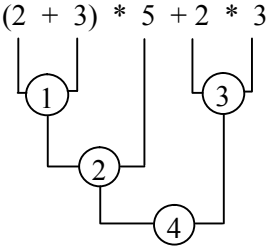
۱۱ـ گزینه «۳» زبان C یک زبان حساس به حروف است (Case Sensitive) یعنی در زبان C مثلاً بین دو شناسه max و MAX تفاوت وجود دارد.

۱۲ـ گزینه «۴» در زبان C تمام بلوک‌ها بین دو علامت { و } قرار می‌گیرد.

۱۳ـ گزینه «۴» بقیه گزینه‌ها جزو کلمات کلیدی زبان C هستند.

۱۴ـ گزینه «۴» پیش‌پردازنده #define می‌تواند داخل بلاک تابع ( ) main و بلاک سایر توابع قرار گیرد. در این صورت شناسه تعریف شده توسط آنها، به صورت محلی بوده و در همان بلاک تعریفشان شناخته هستند. در این قسمت نیز با اولین #define شناسه str به صورت سراسری تعریف می‌شود ولی #define دوم که داخل بلاک ( ) main قرار دارد شناسه str را به صورت محلی تعریف می‌کند و درون تابع ( ) main ارجاع به str تعریف شده در ( ) main صورت می‌گیرد، در نتیجه رشته two در خروجی نمایش داده می‌شود.

۱۵ـ گزینه «۲» در هنگام اجرای برنامه ماکروی A با مقدار 3 و 2 فراخوانی می‌شود و سپس در پارامترهای متناظرش قرار می‌گیرد (a=2,b=3) سپس چون اولویت پرانتز از همه بیشتر است ابتدا حاصل عبارت داخل پرانتز حساب می‌شود (2+3) سپس چون ضرب اولویت بالاتری از جمع دارد حاصل ضرب سمت چپ محاسبه می‌شود و سپس حاصل ضرب سمت راست سپس نتیجه به دست آمده با هم جمع می‌شود یعنی:



۱۶ـ گزینه «۲» با علامت /* */ می‌توان توضیحاتی در چند خط ارائه کرد و با علامت // می‌توان توضیح در یک خط ارائه نمود.

۱۷ـ گزینه «۴» عبارت مقابل ثابت مجازی Boolean چون داخل علامت " " قرار داده شده به عنوان یک رشته درنظر گرفته می‌شود و در نهایت رشته #define True 1 چاپ می‌شود.

۱۸ـ گزینه «۴» برای چاپ کردن خروجی رشته‌ای به وسیله تابع printf می‌توان هم به وسیله %s فرمت خروجی (رشته‌ای) را تعیین کرد یا فقط نام ثابت مجازی را ذکر کرد.

۱۹ـ گزینه «۱» در مقابل دستور #define می‌توان چندین دستور نوشت هنگام کامپایل شدن برنامه و قبل از اجرا شدن برنامه، برنامه پیش‌پردازنده مقدار مربوط به متغیر مجازی را در محل فراخوانی آن کپی کرده و سپس برنامه اجرا می‌شود.

۲۰ـ گزینه «۴»

۲۱ـ گزینه «۳» در زمان اجرای برنامه به جای ثابت مجازی i مقدار آن یعنی صفر قرار می‌گیرد و سپس تقسیم بر صفر صورت می‌گیرد که منجر به خطای زمان اجرای تقسیم برصفر می‌شود.

۲۲ـ گزینه «۱» ماکرو ابتدا کپی می‌شود، سپس محاسبه می‌شود. $2*x*y+x=2*5*3+5=35$

۲۳ـ گزینه «۳» کلمه do , for , if رزرو شده هستند اما دو مورد اول با حروف بزرگ در سؤال آورده شده است.

۲۴ـ گزینه «۲» برای تعریف نوع صحیح باید از int با حروف کوچک استفاده شود.

۲۵ـ گزینه «۴» عملگر # برای استفاده از کتابخانه همراه کلمه include و برای تعریف ماکرو همراه کلمه define استفاده می‌شود.

آزمون فصل اول

۱ـ با کدام یک از پیش‌پردازنده‌های زبان C می‌توان یک دستور جدید تعریف کرد؟

۱) #undef ۲) #else ۳) #define ۴) #include

۲ـ با کدام یک از دستورات پیش‌پردازنده می‌توان یک ماکروی تعریف شده را حذف کرد؟

۱) #hdef ۲) #define ۳) #indef ۴) #undef

۳ـ هر دستور در C حداکثر می‌تواند چند کاراکتر داشته باشد؟

۱) 128 ۲) 255 ۳) 256 ۴) محدودیتی ندارد.

۴ـ کاربرد ";" در یک عبارت C چیست؟

۱) علامت تابع بودن است.
۲) انتهای یک عبارت را نشان می‌دهد.
۳) کاربردی ندارد.
۴) به کاربردن آن اختیاری است.

۵ ـ کدام یک از جملات زیر نادرست است؟

۱) وجود تابع ( ) main در هر برنامه الزامی است.
۲) در هر سطر می‌توان چندین دستور نوشت.
۳) فقط بعضی از گونه‌های C قابلیت برنامه‌نویسی پیمانه‌ای دارند.
۴) تمام نسخه‌های C دستور goto را دارا می‌باشند.

۶ ـ انواع خطاها در برنامه‌نویسی عبارتند از:

۱) خطاهای گرامری و غیرمنطقی
۲) خطای زمان اجرا، کامپایل و منطقی
۳) خطای گرامری و غیرگرامری
۴) خطای منطقی و زمان اجرا

۷ـ برای پاک کردن صفحه نمایش کدام گزینه درست است؟

۱) با استفاده از تابع ;clrscr که در فایل هدر conio.h قرار دارد.
۲) با استفاده از تابع ;( ) clrscr که در فایل هدر conio.h قرار دارد.
۳) با استفاده از تابع ( ) clrscr که در فایل هدر stdio.h قرار دارد.

۸ ـ کدام گزینه نحوه استفاده درست برای معرفی یک فایل هدر است؟

۱) #INCLUDE <conio.h>
۲) #include <conio.h>;
۳) #include 'conio.h';
۴) #include <conio.h>

۹ـ برای تغییر محل پنجره جاری از کدام کلیدهای ترکیبی استفاده می‌شود؟

۱) Ctrl – F5 ۲) Ctrl – F1 ۳) Alt – F5 ۴) Alt – F3

۱۰ـ کدام گزینه نادرست است؟

۱) زبان C نسبت به حروف کوچک و بزرگ حساس است.
۲) Modular programming از ویژگی‌های زبان C می‌باشد.
۳) امکان کار با سخت‌افزار از ویژگی‌های زبان C است.
۴) زبان C مفسری است.