گرافیک کامپیوتری

در برنامه نویسی C++

مؤلف

مهندس مهدی اشرفی

انتشارت پندار پارس

سرشناسه : اشرفی، مهدی، ۱۳۴۷ -
عنوان و نام پدیدآور : گرافیک کامپیوتری در برنامه‌نویسی C++/ مولف مهدی اشرفی.
مشخصات نشر : تهران: پندار پارس: مانلی، ۱۳۸۹.
مشخصات ظاهری : ۱۹۲ ص.: مصور، جدول، نمودار.
شابک : ۵۰۰۰۰ ریال: 978-964-2989-55-3
وضعیت فهرست نویسی : فیپا
موضوع : گرافیک کامپیوتری
موضوع : سی (زبان برنامه‌نویسی کامپیوتر)
رده بندی کنگره : ۱۳۸۹ ۴۳گ۵۸الف/۳۸۵T
رده بندی دیویی : ۰۰۶/۶۷
شماره کتابشناسی ملی : ۴۱۷۱۸۱۲

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ۱٤، واحد ۱٦ **www.pendarepars.com**

تلفن: ٦٦٥٧٢٣٣٥ – تلفکس: ٦٦٩٢٦٥٧٨ همراه: ۰۹۱۲۲٤٥۲۳٤۸ **info@pendarepars.com**

نام کتاب : گرافیک کامپیوتری، در برنامه‌نویسی C++

ناشر : انتشارات پندار پارس ناشر همکار: انتشارات مانلی

تالیف : مهندس مهدی اشرفی

چاپ اول : آذر ۸۹

شمارگان : ۱۰۰۰ نسخه

لیتوگرافی، چاپ، صحافی : ترام‌سنج، صالحان، نوین برتر

قیمت : ۵۰۰۰۰ تومان شابک : ۳-۵۵-۲۹۸۹-۹٦٤-۹۷۸

فهرست

مقدمه

برنامه‌نویسی یکی از جذاب‌ترین فعالیتهای کامپیوتری می‌باشد که در آن می‌توانید هرگونه تصمیمی که در مورد انجام فعالیتی در کامپیوتر داشته باشید را به اجرا در آورید و نگران محدودیتهایی که در اغلب نرم‌افزارها وجود دارد نباشید. زیرا با توجه به شناختی که از کامپیوتر دارید می‌توانید هر کاری را در کامپیوتر انجام دهید و با توجه به نیاز خود برنامه دلخواه‌تان را بنویسید. ولی هنگامیکه با استفاده از یک نرم‌افزار بخواهید کاری را انجام دهید باید توجه داشته باشید که آیا آن نرم‌افزار قادر به انجام آن کار می‌باشد یا نه؟ و آیا نرم‌افزار بهتری نیز وجود دارد یا خیر؟ البته با توجه به پیشرفت علم کامپیوتر در همه زمینه‌ها، امروزه نرم‌افزارهای بسیار زیادی وجود دارد که می‌توانید با استفاده از آنها تمام نیازمندیهای خود را برآورده سازید. شاید شما نیز از آن دسته از افرادی باشید که فکر کنید، امروزه دیگر نیازی به یادگیری برنامه‌نویسی نیست و بجای آن می‌توان با یادگرفتن چند نرم‌افزار به راحتی کارهای کامپیوتری خود را انجام داد. به این دسته از افراد نیز خواندن این کتاب را توصیه می‌کنیم، زیرا با دانستن چگونگی نوشته شدن برنامه و هدف طراحی هر ابزار بهتر می‌توانید از آن نرم‌افزار استفاده کنید و کارایی خود را افزایش دهید.

یکی از بزرگترین دغدغه‌های فکری دانشجویانی که در زمینه برنامه‌نویسی فعالیت دارند، نحوه فکر کردن در مورد طریقه نوشتن برنامه، یا به عبارتی الگوریتم و روش کار برنامه‌نویسی می‌باشد. شاید امروزه با پیشرفت کامپیوتر در زمینه‌های سخت‌افزاری[1] و نرم‌افزاری[2] برنامه‌نویسی گرافیکی به شکلی که در این کتاب به آن پرداخته خواهد شد چندان در حال حاضر کاربردی نباشد. ولی اشتباه نکنید، برای نوشتن بزرگترین نرم‌افزارها هم از همین مقدمات استفاده شده است و برنامه‌نویسان بزرگ نیز اول به آموزش اصول برنامه‌نویسی پرداخته‌اند. برنامه‌نویسی را می‌توان به یک پلکان تشبیه کرد که باید آنرا قدم به قدم و پله پله طی کرد و تا زمانی که شما اصول و قواعد آن را بدرستی فرا نگرفته باشید نمی‌توانید در این زمینه کارهای بزرگی انجام دهید و آنطور که می‌خواهید در زمینه برنامه‌نویسی مسلط شوید.

در این کتاب سعی شده است با توجه به جذابیتهای محیط گرافیکی، اصول برنامه‌نویسی به درستی و با شکلی جدید ارایه گردد تا در کنار یادگیری برنامه‌نویسی در محیط گرافیکی با شیوه صحیح برنامه-نویسی نیز آشنا شوید.

---

[1] Hardware

[2] Software

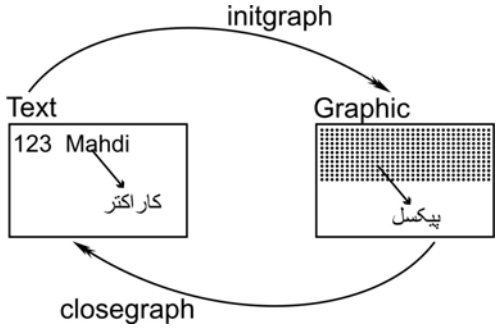
فصل اول

آشنایی با محیط گرافیکی

اغلب سیستمهای کامپیوتری با توجه به نوع سیستم عامل آنها دارای دو محیط کاری متن[1] و گرافیک می-باشند. در محیط متنی، که در اغلب ویراستارها دیده می‌شود، می‌توان یک کاراکتر را تایپ نمود و یا یک کاراکتر را حذف کرد، ولی نمی‌توان در این محیط، نیمی از یک کاراکتر را حذف نمود. به همین دلیل می-گوییم در محیط‌های متنی، واحد آدرس‌پذیر صفحه، کاراکتر می‌باشد.

محیط‌های متنی جدید در ویراستاری مانند Word بسیار تغییر کرده و می‌توان اندازه و نوع فونت متنی را که می‌نویسیم تغییر دهیم. در صورتیکه در محیط‌های متنی قدیمی‌تر مانند Dos اندازه و نوع فونت ثابت بود و قابل تغییر نبود.

در محیط‌های گرافیکی، واحد آدرس‌پذیر صفحه، پیکسل می‌باشد و می‌توان به هر پیکسل[2] از صفحه، رنگ دلخواهی را داد. اغلب زبانهای برنامه‌نویسی از محیط متنی به عنوان محیط پیش‌فرض برنامه‌نویسی استفاده می‌کنند، در این محیط شما می‌توانید اعداد و حروف را وارد نمایید و یا با استفاده از دستورات،



شکل ۱-۱

خروجی آنها را بر روی صفحه نمایش دهید. اگر تاکنون برنامه‌هایی مشابه جمع اعداد، مرتب‌سازی اعداد، مرتب‌سازی چند نام و محاسبه ریشه‌های معادله را نوشته باشید، حتماً با این محیط آشنا شده‌اید.

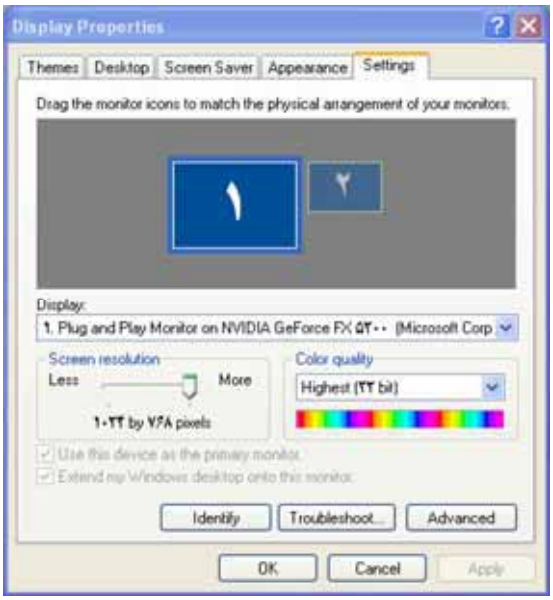
---

[1] Text

[2] Pixel یک نقطه نورانی موجود بر روی صفحه نمایش

در برنامه‌های گرافیکی باید محیط متنی را به محیط گرافیکی تبدیل کنیم. برای این منظور از دستور initgraph استفاده می‌شود. دستور initgraph مخفف Initializes the graphics system به معنی راه‌اندازی و آماده کردن محیط گرافیکی می‌باشد.

برخلاف محیط متنی که دارای ابعاد ثابتی بود (اغلب در DOS این صفحه دارای ۸۰ ستون و ۲۵ سطر بود) محیط گرافیکی بسیار متنوع و قابل تنظیم می‌باشد که نمونه‌ای از آن را می‌توان در تنظیمات صفحه مانیتور در Windows مشاهده نمود.(شکل ۲-۱)



شکل ۲-۱

باتوجه به این تنوع، دسته‌بندیهایی براساس سخت‌افزار وجود دارد که در دستور initgraph باید آنها را مشخص کنیم. دستور initgraph دارای سه پارامتر می‌باشد:

```
initgraph(graph driver, graph mode, path to driver);
```

- Graph driver راه‌انداز محیط گرافیکی می‌باشد که با توجه به سخت‌افزار تعیین می‌شود.

در حال حاضر راه‌اندازهای گرافیکی متنوعی وجود دارند و هر روز در حال تکامل هستند. قدیمی‌ترین راه‌انداز گرافیکی CGA (مخفف Color Graphics Adaptor) می‌باشد که در آن ابعاد صفحه ۲۰۰×۳۲۰ پیکسل با ٤ رنگ است. در جدول ۱-۱ انواع راه‌اندازهای گرافیکی و معادل عددی هرکدام دیده می‌شود.

- هر راه‌انداز گرافیکی را می‌توان به شکلهای مختلفی استفاده کرد که به آنها Graph mode می‌گویند. با تعیین این مقدار، دقت صفحه نمایش[1]، تعداد و نوع رنگها مشخص می‌شود. در جدول ۲-۱ تمامی حالتها و تنظیمات مربوط به هر کدام مشاهده می‌شود.
- در قسمت سوم باید مسیر فایلهای راه‌انداز محیط گرافیکی مورد نیاز برنامه را معرفی کنیم.

برای تعیین راه‌انداز و حالت دلخواه در زبان C دو متغیر صحیح با نامهای دلخواه تعریف می‌کنیم، ولی می‌توان از نامهای gd و gm که مشخص کننده graph driver و graph mode می‌باشد استفاده کرد و حالت موردنظر برای هر کدام را مشخص نمود. بطور مثال می‌توانیم بنویسیم:

```
int   gd,gm;
gd  = VGA;
gm = VGAHI;
```

توجه: زبان C نسبت به حروف کوچک و بزرگ حساس می‌باشد، و در صورت استفاده از نام حالتهای موردنظر باید آنها را با حروف بزرگ نوشت.

می‌توان به جای نوشتن نام حالتها از معادل عددی هر کدام که در جداول ۱-۱ و ۲-۱ دیده می‌شود استفاده کرد. پس می‌توان دستورات قبلی را به شکل زیر نیز نوشت:

```
int   gd,gm;
gd  = 9;
gm = 2;
```

در C راه حل دیگری نیز برای تعریف حالت گرافیکی وجود دارد. در این روش به graph driver مقدار DETECT را می‌دهیم. در این حالت بهترین حالت گرافیکی قابل استفاده در سیستم با توجه به سخت‌افزار تعیین می‌شود و دیگر نیازی به حفظ کردن نام حالتهای گرافیکی یا معادل عددی آنها نمی‌باشد. در این حالت فقط دستورات زیر را می‌نویسیم و دیگر نیازی به مقداردهی به gm نیست.

DETECT	CGA	MCGA	EGA	EGA64	EGAMONO	IBM8514	HERCMONO	ATT400	VGA	PC3270
0	1	2	3	4	5	6	7	8	9	10

جدول ۱-۱: مقادیر عددی معادل هر یک از راه‌اندازهای گرافیکی

[1] Resolution

Graphics Driver	Graphics Modes	Value	Column	Row	Palette	Pages
CGA	CGAC0	0	320	200	C0	1
	CGAC1	1	320	200	C1	1
	CGAC2	2	320	200	C2	1
	CGAC3	3	320	200	C3	1
	CGAHI	4	640	200	2 color	1
MCGA	MCGAC0	0	320	200	C0	1
	MCGAC1	1	320	200	C1	1
	MCGAC2	2	320	200	C2	1
	MCGAC3	3	320	200	C3	1
	MCGAMED	4	640	200	2 color	1
	MCGAHI	5	640	480	2 color	1
EGA	EGALO	0	640	200	16 color	4
	EGAHI	1	640	350	16 color	2
EGA64	EGA64LO	0	640	200	16 color	1
	EGA64LHI	1	640	350	4 color	1
EGAMONO	EGAMONOHI	3	640	350	2 color	1
	EGAMONOHI	3	640	350	2 color	2
HERC	HERCMONOHI	0	720	348	2 color	2
ATT400	ATT400C0	0	320	200	C0	1
	ATT400C1	1	320	200	C1	1
	ATT400C2	2	320	200	C2	1
	ATT400C3	3	320	200	C3	1
	ATT400MED	4	640	200	2 color	1
	ATT400HI	5	640	400	2 color	1
VGA	VGALO	0	640	200	16 color	2
	VGAMED	1	640	350	16 color	2
	VGAHI	2	640	480	16 color	1
PC3270	PC3270HI	0	720	350	2 color	1
IBM851	IBM8514LO	0	640	480	256 color	
	IBM8514HI	1	1024	760	256 color	

جدول ۲-۱: راه‌اندازهای گرافیکی و حالتهای هر کدام

```
int   gd,gm;
gd  = DETECT;
```

این سه تعریف با هم برابرند و نتیجه آنها صفحه‌ای با ابعاد ۶۴۰×۴۸۰ و ۱۶ رنگ خواهد بود.

برای استفاده از راه‌انداز و حالت گرافیکی موردنظر در دستور initgraph در قسمت اول gd و در قسمت دوم gm را می‌نویسیم. اگر به هر دلیلی دستور initgraph نتواند محیط گرافیکی را ایجاد نماید این دو متغیر مقدارشان تغییر می‌کند. برای برگرداندن مقادیر جدید به برنامه، در زبان C باید آدرس این دو متغیر به تابع ارسال شود و این کار توسط علامت & قبل از نام هر متغیر انجام می‌شود.

در قسمت سوم برای تعیین مسیر باید آدرس شاخه bgi که شامل فایلهای راه‌انداز و فایلهای فونت می‌باشد را تعیین کنیم. برای این منظور آدرس درایو و نام شاخه‌ای[1] که C در آن نصب شده را می‌نویسیم. بطور مثال اگر کامپایلر C بر روی درایو c: و در شاخه TC نصب شده باشد مسیر را به شکل "c:\\tc\\bgi" می‌نویسیم.

توجه: چون در زبان C کاراکتر \ (Back slash) در رشته‌ها دارای معنی می‌باشد و کاراکتر بعد از آن تفسیر می‌شود، مانند \n که برای رفتن به ابتدای خط می‌باشد، بنابراین در اینجا چون خود کاراکتر \ موردنیاز می‌باشد آن را بصورت دوتایی می‌نویسیم.

نوشتن آدرس بصورت ثابت، این مشکل را دارد که اگر بخواهیم این برنامه را در کامپیوتر دیگری نیز اجرا کنیم باید کامپایلر C در همان درایو و شاخه‌ای که نوشته‌اید، وجود داشته باشد. برای رهایی از این مشکل می‌توانیم آدرس را ننویسیم و از یک رشته تهی استفاده کنیم. در این حالت فهرست جاری برای یافتن فایلها جستجو می‌شود و می‌توانیم فایلهای موردنظر را از شاخه bgi در فهرست جاری (شاخه bin) کپی کنیم و اگر بخواهیم فقط فایل اجرایی را در کامپیوتر دیگری اجرا کنیم، همراه فایل اجرایی این فایلها را نیز در کامپیوتر مقصد در یک شاخه کپی می‌کنیم. اگر از راه‌انداز VGA استفاده کنیم فقط فایل راه‌انداز EGAVGA.BGI موردنیاز می‌باشد. بنابراین باید این فایل را از شاخه bgi در شاخه bin کپی کنیم. همچنین اگر از فونتی استفاده شود آن فایلها نیز باید کپی گردد.

با جمع‌بندی مطالبی که گفته شده می‌توانیم دستور initgraph را به شکل زیر بنویسیم:

```
initgraph( &gd , &gm , "" );
```

بعد از اجرای این دستور یک صفحه مشکی گرافیکی ایجاد می‌شود که آماده اجرای دستورات می‌باشد. در انتهای برنامه از دستور closegraph(); برای تبدیل محیط گرافیکی به محیط متن استفاده می-

---

[1] Sub directory (در Dos) یا Folder (در Windows)

شود، ولی چون بعد از اجرای این دستور کل صفحه پاک می‌شود و یک صفحه مشکی متنی نمایش داده می‌شود، قبل از اجرای این دستور توسط دستوراتی مانند getch(); یا delay توقفی در برنامه ایجاد می‌کنیم تا بتوانیم خروجی برنامه را ببینیم.

برای اجرای دستورات گرافیکی باید فایل سرآیند[1] Graphics.h را توسط #include به برنامه معرفی کنیم. همچنین در اغلب برنامه‌های گرافیکی که می‌نویسیم دستورات موجود در فایل سرآیند Conio.h نیز موردنیاز می‌باشد. با توجه به کلیه مطالب گفته شده شروع و پایان برنامه‌ها بطور کلی به شکل زیر می‌تواند می‌باشد:

```
#include<graphics.h>
#include<conio.h>
void main(void){
  int  gd , gm;
  gd = DETECT;
  initgraph( &gd , &gm , "" );
```

دستورات گرافیکی موردنیاز برنامه

```
  getch();
  closegraph();
}
```

تابع graphresult: اگر به هر دلیلی موفق به آماده کردن محیط گرافیکی جهت نوشتن برنامه‌های گرافیکی نشوید، می‌توانید توسط این تابع مشکل را فهمیده و آن را برطرف کنید.

مقدار بازگشتی این تابع Error یا همان مشکل بوجود آمده را مشخص می‌کند. این مقادیر و مشکلات مربوط به هر کدام در جدول ۳-۱ آمده است.

تابع grapherrormsg: این تابع باتوجه به کد خطایی که رخ داده است پیام مناسبی تولید می‌کند. در این تابع معمولاً از تابع graphresult به عنوان پارامتر ورودی، جهت تعیین نوع خطای رخ داده استفاده می‌شود.

هرگاه بخواهیم خطایی که باعث راه‌اندازی نشدن محیط گرافیکی شده است را ببینیم می‌توانیم برنامه را به شکل کاملتر زیر بنویسیم:

```
#include<graphics.h>
```

---

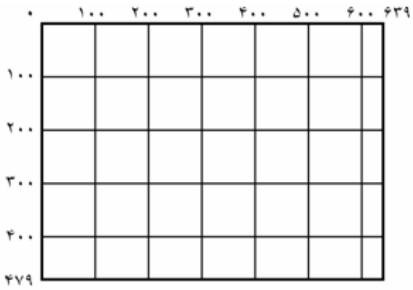
[1] Header file

```
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
void main(void){
  int  gd , gm;
  gd = DETECT;
  initgraph( &gd , &gm , "" );
  if(graphresult()!=grOk){
    printf("Graphics error: %s\n" ,
grapherrormsg(graphresult()));
    printf("Press any key to halt…");
    getch();
    exit(1);
  }
```

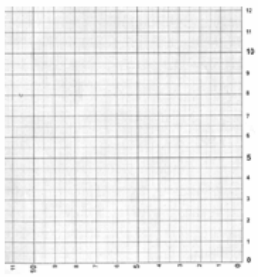
دستورات گرافیکی موردنیاز برنامه

```
  getch();
  closegraph();
}
```

اگر اولین برنامه گرافیکی اجرا شود دیگر هربار نیاز به بررسی خطا وجود ندارد. در این کتاب برای پرهیز از تکرار این دستورات در هر برنامه و حجیم شدن کتاب، برنامه‌ها به شکل اول نوشته می‌شود. برنامه‌های این فصل با هدف شناختن بهتر صفحه گرافیکی طراحی گردیده است. برای این منظور می‌توانید از صفحات میلی‌متری (شکل ۳-۱) استفاده کنید و یا صفحه‌ای را به شکل ٤-۱ خط‌کشی کنید تا بهتر بتوانید مختصات نقاط دلخواه‌تان را پیدا کنید. سپس شکلی را که می‌خواهید بر روی صفحه نمایش داده شود بر روی این صفحه رسم نمایید و مختصات هر نقطه آن را تعیین کنید و با دستورات لازم آنها را رسم نمایید.



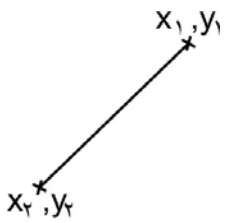
شکل ۴-۱


شکل ۳-۱

کد خطا	نام متناظر با خطا	توضیح خطا
۰	grOk	اشکالی وجود ندارد
-۱	grNoInitGraph	محیط گرافیک نصب نشده
-۲	grNotDetected	سخت‌افزار گرافیکی پشتیبانی نمی‌شود
-۳	grFileNotFound	فایل راه‌انداز گرافیکی پیدا نشد
-۴	grInvalidDriver	فایل راه‌انداز گرافیکی درست نیست
-۵	grNoLoadMem	حافظه کافی برای راه‌اندازی محیط گرافیکی وجود ندارد
-۶	grNoScanMem	کمبود حافظه در مرحله خواندن شاخه رنگ‌آمیزی
-۷	grNoFloodMem	کمبود حافظه برای رنگ‌آمیزی یک شاخه
-۸	grFontNotFound	فایل فونت وجود ندارد
-۹	grNoFontMem	حافظه کافی برای فراخوانی فایل فونت وجود ندارد
-۱۰	grInvalidMode	حالت گرافیکی برای این راه‌انداز صحیح نمی‌باشد
-۱۱	grError	اشکال گرافیکی وجود دارد
-۱۲	grIOerror	ورودی/خروجی گرافیکی اشکال دارد
-۱۳	grInvalidFont	فایل فونت صحیح نیست
-۱۴	grInvalidFontNum	شماره فونت صحیح نیست
-۱۵	grInvalidDeviceNum	شماره دستگاه صحیح نیست
-۱۸	grInvalidVersion	ورژن صحیح نیست

جدول ۳-۱: جدول خطاها در محیط گرافیکی

دستور line برای رسم یک پاره خط می‌باشد که در آن باید مختصات نقطه شروع و پایان را مشخص کنیم. توجه داریم که در این دستور فرقی بین نقطه شروع و پایان وجود ندارد و می‌توان مختصات هر یک از دو سر پاره خط را نوشت و سپس مختصات نقطه دیگر را نوشت. بنابراین در شکل ۵-۱ دو دستور زیر به یک شکل کار می‌کنند.

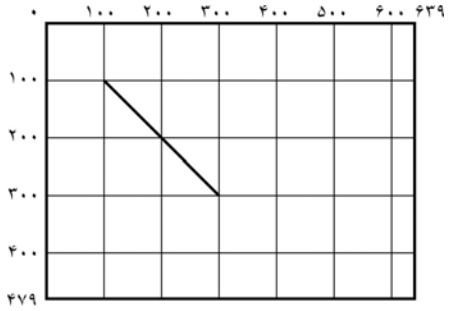


شکل ۵-۱

```
line(x1, y1, x2, y2); line(x, y2, x1 , y1);
```

برنامه ۱-۱: اگر بخواهیم بر روی صفحه نمایش خطی به مانند شکل زیر رسم نماییم کافیست مختصات دو سر خط را در دستور line بنویسیم و همانطور که از شکل پیداست یکی (۱۰۰،۱۰۰) و دیگری (۳۰۰،۳۰۰) می‌باشد.

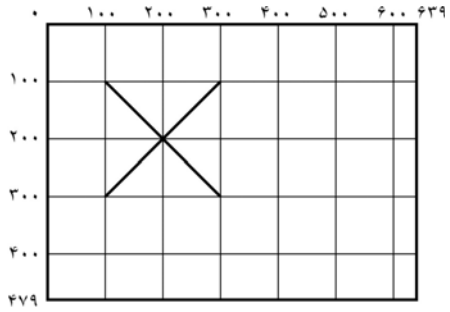
```
#include<graphics.h>
#include<conio.h>
void main(void){
  int gd , gm;
  gd = DETECT;
  initgraph(&gd , &gm ,  "");
  line(100 , 100 , 300 , 300);
  getch();
  closegraph();
}
```



شکل ۶-۱

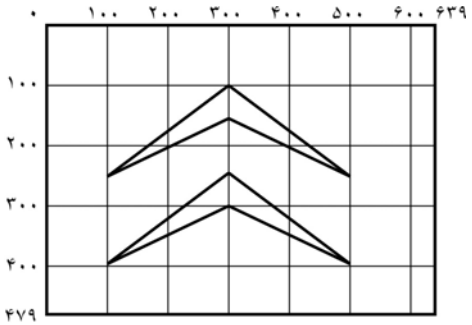
برنامه ۲-۱: برای رسم یک ضربدر کافیست در برنامه قبل خطی با مختصات (۳۰۰،۱۰۰) و (۱۰۰،۳۰۰) را اضافه کنیم.

```
#include<graphics.h>
#include<conio.h>
void main(void){
  int gd , gm;
  gd = DETECT;
  initgraph(&gd , &gm , "");
  line(100 , 100 , 300 , 300);
  line(300 , 100 , 100 , 300);
  getch();
  closegraph();
}
```



شکل ۷-۱

برنامه ۳-۱: برنامه‌ای بنویسید که آرم زانتیا را رسم نماید.



شکل ۸-۱

```
#include<graphics.h>
#include<conio.h>
```