تعریف الگوریتم

هر دستورالعملی که مراحل انجام کاری را با زبانی دقیق و با جزئیات کافی بیان نماید بطوریکه ترتیب مراحل و شرط خاتمه عملیات در آن کاملا" مشخص شده باشد را الگوریتم گویند.

- منظور از زبان دقیق: آن است که الگوریتم دقیقا" به همان صورتیکه مورد نظر نویسنده است اجرا گردد.
- منظور از جزئیات کافی ، آن است که در طول اجرای الگوریتم عملیات ناشناخته پیش نیامده و باعث انحراف از مسیر و هدف اصلی نگردد.
- منظور از ترتیب مراحل، آن است که مراحل اجرای الگوریتم قدم به قدم و با رعایت تقدم و تأ خر مشخص شده باشد.
- منظور از شرط خاتمه، پایان پذیر بودن الگوریتم می باشد و بهر حال الگوریتم باید در زمانی دلخواه و تحت شرایط یا شرایط داده شده خاتمه پذیرد.

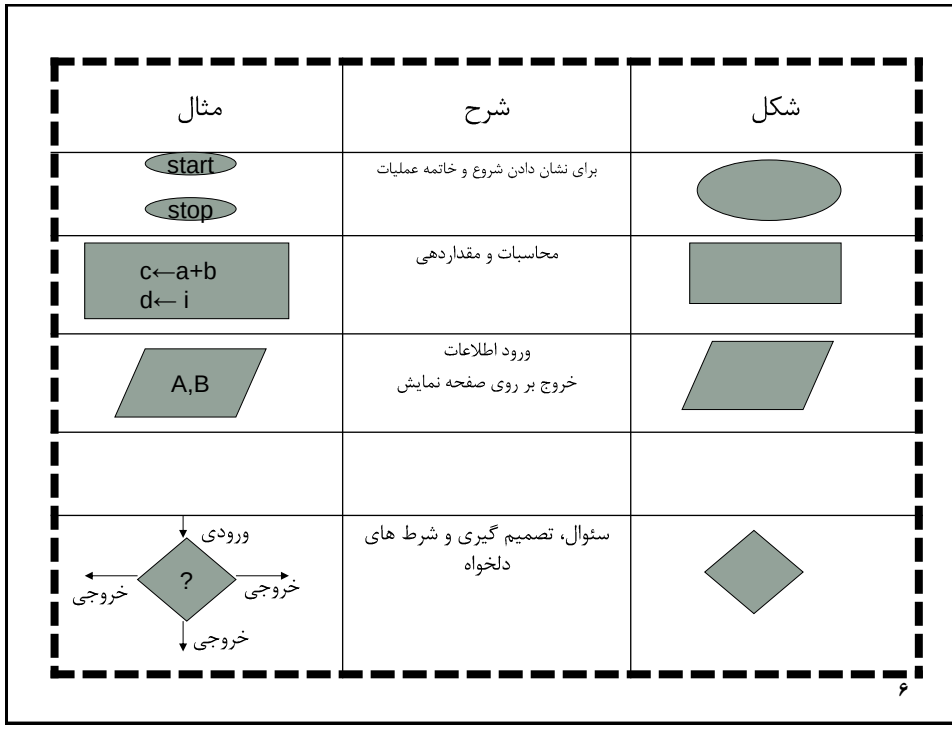
مراحل تهیه الگوریتم

برای تهیه یک الگوریتم خوب و کارآمد باید مراحل خاصی اجرا شوند:

1- تعریف دقیق مسئله: باید مسئله را تجزیه و تحلیل کرده تا کوچکترین ابهامی در فهم آن وجود نداشته باشد.
2- تعیین عوامل اصلی(متغیرهای) مورد نیاز
3- تعیین ورودی و خروجی مسئله : (داده ها و اطلاعات)
4- بررسی راه حل های مختلف مسئله
5- انتخاب یک راه حل مناسب
6- اشکال زدایی

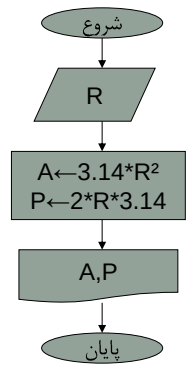
فلوچارت

- بیان تصویری الگوریتم
- مراحل انجام کار با اشکال هندسی نشان داده می شوند.
- مراحل انجام کار توسط خطوط به هم وصل می گردند.



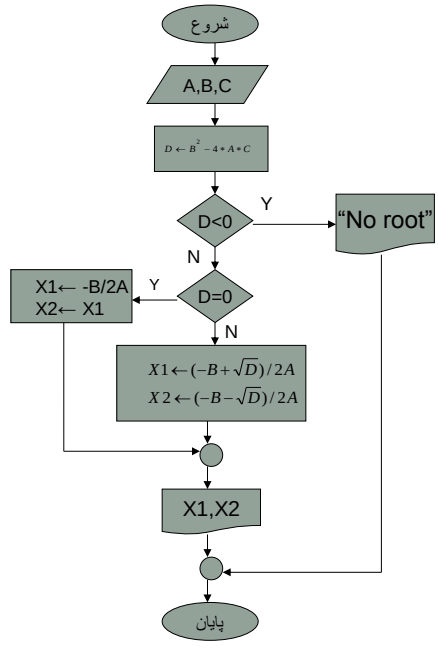
۶

مثال 1 : فلوچارتی رسم کنید که شعاع یک دایره را خوانده، مساحت و محیط آنرا نمایش دهد.



مثال2: ریشه های یک معادله درجه دوم

- $AX^2+BX+C=0$
- $D=B^2-4AC$
- اگر $D<0$، معادله ریشه ندارد
- اگر $D=0$، حاصل عبارت $-B/2A$ را در X1و X2 قرار بده
- حاصل عبارت $(-B-\sqrt{D})/2A$ را در X1 قرار بده
- حاصل عبارت $(-B+\sqrt{D})/(2A)$ را در X2 قرار بده
- مقادیر X1 و X2 را نمایش بده



برنامه نویسی

- یک برنامه در واقع مجموعه ای از دستورات است که در حافظه ذخیره می شود و سپس کامپیوتر آنها را اجرا می کند.

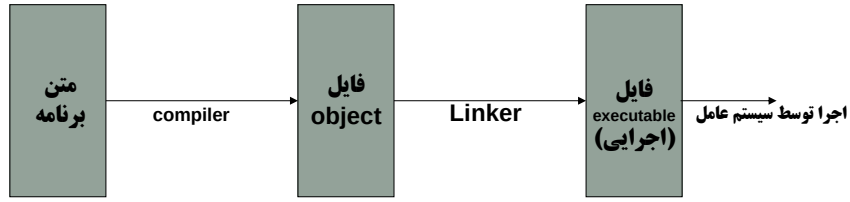
زبان ماشین و اسمبلی

- چون برنامه های اولیه به صورت کدهای دودوئی که مستقیما قابل فهم به زبان ماشین بود نوشته می شدند به این برنامه ها برنامه ها به زبان ماشین (machine language program) گفته می شد و به هر دستور، یک دستور زبان ماشین گفته می شد.
- مشکل نوشتن برنامه به زبان ماشین سختی نوشتن و ناخوانایی آن بود.
- برای رفع این مشکل زبانهای اسمبلی شکل گرفت. در این زبان ها برای هر دستور زبان ماشین یک عبارت تعریف شده است. مثلا برای جمع دو خانه حافظه A و B عبارت ADD A,B . در این حالت برنامه نویس به جای نوشتن یک سری ۰ و۱ ، با این عبارات برنامه خود را می نوشت.
- برنامه ای که برنامه اسمبلی نوشته شده توسط کاربر را به زبان ماشین تبدیل می کند اسمبلر خوانده می شود.

زبان های سطح بالا

- با وجود آنکه زبان های اسمبلی کار برنامه نویسی را آسان می کرد اما باز برنامه ها طولانی و معمولا ناخوانا بودند.
- برای رفع این مشکل زبان های سطح بالا بوجود آمدند. در این زبان ها هر چند دستور زبان ماشین به یک عبارت با معنا که به زبان معمولی نزدیک بود تبدیل می شود.
- دو برنامه کامپایلر(compiler) و linker روی هم کار تبدیل برنامه سطح بالا به زبان ماشین را انجام می دهند.

مراحل ایجاد یک برنامه



مراحل ایجاد برنامه سطح بالا: نوشتن متن برنامه ، کامپایل، link و اجرا

به مجموع دو عمل compile و link اصطلاحا Build می گوییم.

Build=compile+link Builder=compiler+linker

نکته

- در یک زبان عادی برای بیان یک مفهوم می توان از عبارات مختلفی استفاده کرد :
  - علی به خانه رفت.
  - علی خانه رفت.
  - علی رفت خانه.
  - رفت علی خانه.

- در یک زبان برنامه نویسی نیاز دستورات نیاز به بیان دقیق دارند و باید طبق ساختار مشخصی که در زبان مشخص شده استفاده شوند تا کامپایلر قادر به درک آنها باشد.

ساده ترین برنامه به زبان C

```
void main( )
{
}
```

نکات:

1. خط اول در هر برنامه C باید وجود داشته باشد.
2. { شروع برنامه
3. } خاتمه برنامه
4. دستورات برنامه در داخل {} نوشته می شوند.

نمایش مراحل سه گانه نوشتن، ***COMPILE***، ***LINK*** و اجرا

1. **نوشتن**
2. **Compile**
3. **Link**
4. **اجراء**

IDE

مشکل: زمانبر بودن

رفع مشکل: ارائه نرم افزارهایی که امکان ویرایش، کامپایل، link **و اجرا را در یک محیط فراهم می کنند.**

IDE: Integrated Development Environment

مانند: Borland C، Visual C **و....**

نکته –۱

- فاصله گذاری
- حساس بودن به حالت حروف (case sensitivity)

نکته–۲

- error: به خطاهای برنامه نویسی error می گویند.
- انواع خطاها در برنامه نویسی:
  - خطاهای زمان compile(compile errors):
    - مانع کامپایل صحیح برنامه می شوند.
  - خطاهای زمان link (Link errors):
    - برای کامپایل مزاحمتی ایجاد نمی کنند اما مانع Link برنامه می شوند.
  - خطاهای زمان اجرا: (Run time errors):
    - کامپایل و Link با موفقیت انجام می شود ولی اجرای برنامه دچار اشکال می شود .

ERROR

- حسن سیب را خورد.
- هسن سیب را خورد.
  - متناظر با خطای کامپایل
- را حسن خورد سیب.
  - متناظر با خطای Link
- سیب حسن را خورد.
  - متناظر با خطای زمان اجرا

مثال ۲) نمایش متن بر روی مانیتور

برنامه ای بنویسید که پیغام Hello را در مانیتور نشان دهد.

```
#include <stdio.h>
void main()
{
  printf("Hello");
}
```

پیغام Hello **چاپ می شود و مکان نما بعد از حرف** o **قرار می گیرد.**

نکات

- **Stdio.h نمونه ای از یک header file است. فایل های header جزئیات غیر مرتبط با کاربر را از دید او مخفی می کند و موجب می شود برنامه ای خلاصه تر و خواناتر داشته باشیم.**

- **دستور .... include# یک راهنمای پیش پردازش (preprocessor directive) یا راهنمای کامپایلر (compiler directive) خوانده می شود. Compiler قبل از شروع کامپایل محتویات این فایل را به برنامه اضافه می کند و سپس کامپایل آغاز می گردد.**

- **در انتهای هر دستور زبان C داخل main علامت ; قرار داده می شود.**

\N

- Printf("Hello welcome");

خروجی ؟؟

- **فاصله ها هم نمایش داده می شوند.**
- **می خواهیم خروجی ما به صورت زیر باشد:**

Hello
Welcome

```
#include <stdio.h>
Void main( )
{
   Printf("Hello
Welcome");
}
```

```
#include <stdio.h>
Void main( )
{
   Printf("Hello\nWelcome");
}
```

خطا

\T

```
#include <stdio.h>
Void main( )
{

  Printf("Hello\tWelco
  me");
}
```

حرکت به اندازه یک tab

n\و t\ اصطلاحا کاراکترهای کنترلی نامیده می شوند.
کاراکترهای کنترلی در printf عینا چاپ نمی شوند،
بلکه اثراتی دیگر از خود بروز می دهند.

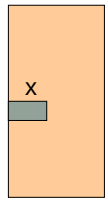
- آشنایی با مفهوم متغیرها، اپراتورها (عملگرها)
- چاپ مقدار متغیرها
- دستورات ورودی
- فرمت بندی خروجی
- تبدیل انواع
- تقدم عملگرها

مثال

- برنامه ای بنویسید که حاصل ۲+۲ را محاسبه کند.

```
void main( )
{
   int x;
   x=2+2;
}
```

نکته–۱

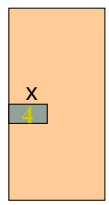


- int x;
  - هنگام اجرای این دستور:
    - یک مکان ۴ بایتی حافظه به برنامه اختصاص داده می شود.
    - این مکان را x می نامد.
    - به برنامه اجازه می دهد که یک عدد صحیح (integer) در x بریزد.
  - اصطلاحا به x، یک متغیر (Variable) از نوع صحیح (int) گفته می شود.
  - به دستور بالا دستور تعریف متغیر می گوییم.
  - x نام متغیر می باشد.
  - نام متغیر
    - ترکیبی از حروف a تا z و A تا Z، ارقام 0 تا 9 و _ (Undeline) می باشد.
    - نباید با یک رقم شروع شود.

مثال : اسامی مجاز: a، asd، ali، A09 ، _A

اسامی غیر مجاز: 9a، A-s

نکات–۲



- X=2+2;
  - ابتدا 2 با 2 جمع می شود.
    - به علامت + اصطلاحا یک operator (عملگر) میگوییم.
    - هر یک از اعداد 2 یک operand (عملوند) خوانده میشوند.
  - سپس حاصل جمع در x قرار می گیرد.

    یا اصطلاحا 4 به x تخصیص می یابد.
  - به = اصطلاحا یک اپراتور تخصیص مقدار و به دستور بالا یک دستور تخصیص مقدار (Assignment statement) میگوییم.
  - درسمت چپ یک دستور تخصیص مقدار فقط و فقط نام یک متغیر باید باشد. سمت راست می تواند ترکیبی از نام متغیر و مقادیر باشد.

بعضی انواع متغیرها

	مصرف حافظه (بایت)	نوع	محدوده
int	2	صحیح	32767 تا 32767-
float	4	اعشاری	ماکزیمم 6 رقم اعشار دقت- $(10^{-37} - 10^{38})$
double	8	اعشاری با دقت بالا	ماکزیمم 15 رقم اعشار دقت $(10^{-307} - 10^{308})$
long double	8	اعشاری با دقت بالا	ماکزیمم 19 رقم اعشار دقت $(10^{-4932} - 10^{4932})$
char	1	کاراکتر	127 تا 127-

تعریف نادرست داده های مربوط به هر متغیر

```
int x;
x=2.3;
```

```
short x;
x=3243243255556;
```

```
float x;
x=3;
```

بعضی انواع اپراتورها

- **محاسباتی:**
  + − * / % (باقیمانده) ++ −−
- **تخصیص مقدار:**
  =
- **مقایسه ای:**
  > < == !=

صورت های دیگر نوشتن برنامه

```
Void main ()
{
   int x;
   int y;
   y=2;
   x=y+2;
}
```

```
Void main ()
{
   int x,y;
   y=2;
   x=y+2;
}
```

```
Void main ()
{
   int y=2;
   int x;
   x=y+2;
}
```

```
Void main ()
{
   int y=2,x;
   x=y+2;
}
```

```
void main( )
{
   int x;
   x=2+2;
}
```

```
Void main ()
{
   int z=2,y=2,x;
   x=z+y;
}
```

```
Void main ()
{
   int x=4;
   int y=5;
   x=2;
   y=x+2;
}
```

آخرین مقادیر

x,y

x=2 y=4

نمایش مقادیر در مانیتور

- مسئله: می خواهیم مقدار یک متغیر را روی مانیتور چاپ کنیم.

```
#include <stdio.h>
Void main ()
{
   int x=4;
   int y=5;
   x=2;
   y=x+2;
   printf("Result=%d",y);
}
```

%d یعنی یک عدد صحیح را روی مانیتور نشان بده
این عدد صحیح y خواهد بود.

%d نیز یک کاراکتر کنترلی می باشد.

نکات

- چاپ بیش از یک مقدار

```
Void main ()
{
   int x=2,y=2,z;
   z=x+y;
   printf(“%d + %d =%d”,x,y,z);

}
```

کاراکترهای کنترلی مشابه %D

%d	اعداد صحیح دهدهی	
%f	اعداد اعشاری	float x=23.2; printf(“%f”,x);
%c	کاراکتر	char ch=‘B’; char ch=66; printf(“%c”,ch);
%x	اعداد مبنای 16 صحیح	int a=0x234; printf(“%x”,a);

چند مثال

- printf(“%d”,23);
  - 23 چاپ می شود.
- printf(“%c”,’a’)
  - کاراکتر a روی مانیتور چاپ می شود.
- printf(“%d”,’a’);
  - کد اسکی کاراکتر a یعنی ۹۷ چاپ می شود.
- printf(“%c”,97)
  - کاراکتری که کد اسکی آن ۹۷ است یعنی a چاپ می شود.

دستورات ورودی

- **با استفاده از دستورات ورودی می توانیم مقدار متغیرها را از طریق** keyboard **وارد کنیم.**

دستور SCANF

برنامه ای بنویسید که عددی صحیح را از keyboard **دریافت کند و مربع آن را چاپ کند.**

```
#include <stdio.h>
void main()
{
   int x;
   scanf(“%d”,&x);
   printf(“%d”,(x*x));
}
```

علامت آدرس

یک عدد صحیح را از کاربر دریافت و آن را در متغیر x **می ریزد.**

نکات

- به طرزی مشابه printf
  - برای خواندن اعداد اعشاری: %f
  - خواندن کارکترها:%c
  - خواندن اعداد مبنای 16: %x
- برای استفاده از scanf هم باید از stdio.h استفاده کرد.
- بعد از وارد کردن مقدار باید ENTER زده شود.

مثال

- برنامه ای بنویسید که دو عدد را از صفحه کلید دریافت کرده و حاصل ضرب آنها را چاپ کند.

```
#include <stdio.h>
void main()
{
   int x,y;
   printf("\nEnter two numbers");
   scanf("%d%d",&x,&y);
   printf("The sum=%d",(x*y));
}
```

هنگام وارد کردن اعداد باید بین آنها لااقل یک فاصله قرار داده شود یا ENTER زده شود.
بعد از وارد کردن عدد آخر باید ENTER زده شود.
دستور scanf بالا را می توان به صورت زیر هم نوشت:

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

می توان بیش از دو مقدار هم با دستور scanf دریافت کرد.

یک مشکل در گرفتن کاراکتر با SCANF

- برنامه ای بنویسید که دو کاراکتر بگیرد و سپس آنها را در کنار هم چاپ کند.
ظاهرا این برنامه باید به صورت زیر باشد:

```
#include <stdio.h>
void main()
{
  char c1,c2;
  scanf("%c%c",&c1,&c2);
  printf("%c%c",c1,c2);
}
```

اما هنگام اجرای برنامه ملاحظه می کنیم که تنها یکی از کاراکترهای وارد شده چاپ می شود.

بافر ورودی

- علت رخ دادن این مشکل:
  - داده هایی که در هنگام اجرای scanf وارد می کنیم در داخل محلی از حافظه که بافر دستور scanf خوانده می شود ذخیره می شود. دستور printf بسته به آنچه برایش مشخص شده است از این بافر عنصر بر می دارد و روی مانیتور چاپ می کند.
  - در این مثال هنگام اجرای scanf ما ابتدا یک کاراکتر (مثلا a) ، سپس فاصله یا ENTER و بعد کاراکتربعد (مثلا b) را وارد کرده ایم. با توجه به اینکه فاصله یا ENTER در واقع خود یک کاراکتر هستند در واقع ما سه کاراکتر وارد کرده ایم که هر ۳ در بافر scanf قرار می گیرند.
  - هنگام اجرای printf دو تا کاراکتر اول یعنی a و فاصله (یا ENTER) چاپ می شود و کاراکتر مورد نظر (b) چاپ نمی شود.
  - برای رفع مشکل می توان نوشت :

```
scanf("%c%c%c",&c1,&c2,&c3);
```

- به طور کلی استفاده از scanf برای گرفتن کاراکتر مناسب نیست. دو دستور که مخصوص گرفتن کاراکتر هستند و مشکل بالا را ندارند getche و getch می باشند.

دستور GETCHE

- **تنها برای دریافت کاراکتر از** keyboard **به کار می رود.**
- **تفاوت آن با** scanf **این است که در اینجا نیازی به زدن** Enter **بعد از وارد کردن مقدار نیست.**
- **نیاز به** header file **با نام** conio.h **دارد.**

```
#include <conio.h>
#include <stdio.h>
void main()
{
char ch;
ch=getche();
printf("\n%c",ch);
}
```

دستور getch

- **کاملا مشابه** getche **می باشد. تنها تفاوت آن این است که ورودی را نمی بینیم.**

فرمت بندی خروجی

```
#include <stdio.h>
Void main()
{
   float num=3.4;
   printf(“%f”,num);
}
```

- **روی مانیتور،** 3.400000 **چاپ خواهد شد.**
- %f **به طور پیش فرض عدد را با** 6 **رقم اعشار نشان می دهد.**
- **با فرمت بندی خروجی می توانیم مشخص کنیم اعداد چگونه نمایش داده شوند.**

فرمت بندی خروجی (ادامه)

- **در فرمت بندی خروجی دو چیز مشخص می شود:**
  - **تعداد ارقام اعشار**
  - **تعداد مکان های در نظر گرفته شده برای نمایش عدد (طول میدان)**

مثالها:

```
float x=2.3;
printf("%.3f",x);
```

مقدار چاپ شده : 2.300

```
float x=2.3;
printf("%.7f",x);
```

مقدار چاپ شده : 2.3000000

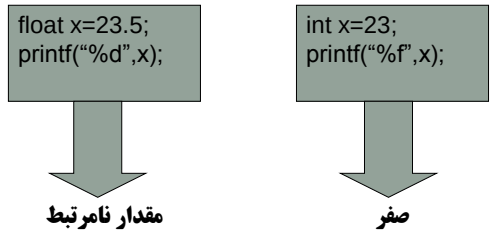
رقم بعد از نقطه تعداد ارقام اعشاری را مشخص می کند.

برخی از انواع خطاها
تخصیص مقدار خارج از محدوده عدد

```
short num;
num=234567;
```

عددی در محدوده short
بین 32768– **تا** 32767

استفاده از کاراکتر کنترلی نامناسب



در استفاده از کاراکترهای کنترلی % در printf باید دقت کرد که متناظر با نوع متغیر باشد. مثال های بالا نمونه هایی از خطاهای زمان اجرا می باشد.

اپراتورهای ++ و --

X= X+1

≡ X++ یا ++X

X=X-1

≡ X-- یا --X

تفاوت I++ و ++I يا I-- و --I

```
#include <stdio.h>
void main()
{
  int i=0;

    printf("%d", i++);
}   printf("%d",++ i);
```

خروجي: 0

خروجي: 1

=+ و =-

```
X=X+23;
≡ X+=23;

X=X+Y;
≡ X+=Y;
```

```
X=X-23;
≡ X-=23;

X=X-Y;
≡ X-=Y;
```

```
X*=z;
X/=Y;
X%=z;
```

تقدم (اولویت) عملگرها

X=2+4*5%2;

؟

مثال

2+4%2=2+0=2

(2+4)%2=(6)%2=0

()
++ - -
/ * %
+ -
> >= < <=
== !=
= += -= *= /= %=

در یک عبارت ،تقدم اپراتورهایی که در یک سطح اولویت قرار دارند از چپ به راست می باشد.