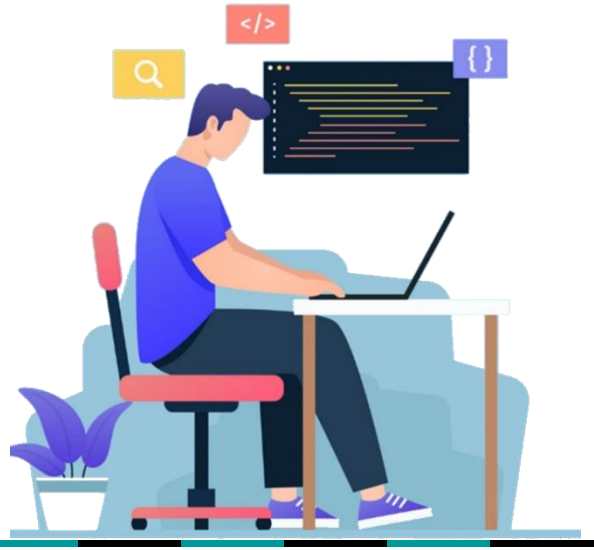
اصول و مبانی برنامه‌نویسی



مجید شبیری

کارشناسی ارشد IT، گرایش شبکه

از دانشگاه صنعتی امیرکبیر

تفکر الگوریتمی و تکنیک‌های حل مسئله

سرفصل مطالب

حل مسئله (Problem Solving)

- **کامپیوتر** به تنهایی **قادر به حل مسئله نیست.**
- کامپیوتر فقط می‌تواند **دستورات برنامه‌نویس** (راه حل ارائه شده و پیاده سازی شده مسئله) را اجرا کند.
- این **برنامه‌نویس** است که ابتدا مسئله را **حل** و سپس **پیاده‌سازی** می‌نماید.

گام‌های حل مسئله

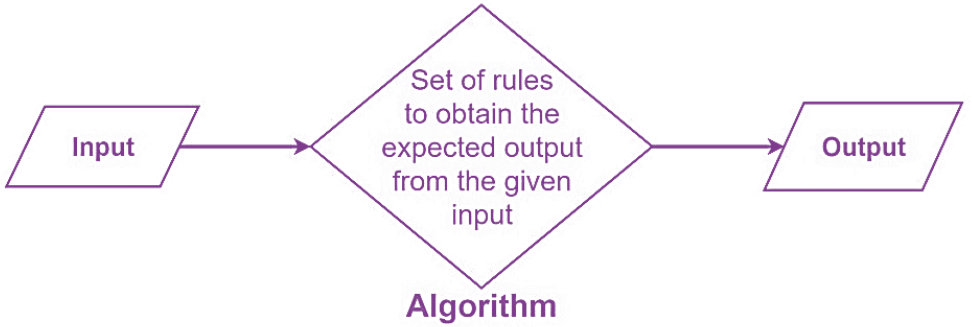
- **درک** صحیح مسئله
- طراحی **الگوریتم**
- پیاده‌سازی (تبدیل الگوریتم به **برنامه**)
- **ارزیابی** صحت عملکرد محصول

درک صحیح مسئله

- بررسی دقیق **صورت مسئله**
- تشخیص **قابلیت** (functionality) های مورد نیاز
- تشخیص درست **خروجی مورد نیاز**
- یافتن **راهی مناسب برای تولید خروجی** مورد نظر
- ترسیم **ارتباطی صحیح** میان **ورودی** و **خروجی**
- تشخیص **ورودی‌های مورد نیاز** (اجتناب از گرفتن ورودی غیرضروری)

الگوریتم چیست؟

- الگوریتم (Algorithm) یعنی دنباله‌ای از گام‌های متناهی برای **رسیدن به پاسخ** یک **مسئله**.
- الگوریتم تکنیک مناسبی برای بیان **راه‌حل** یک **مسئله** می‌باشد.
- الگوریتم معمولاً برای مسائل ریاضی و کامپیوتری نوشته می‌شود.
- برای نوشتن الگوریتم، باید **راه‌حل مسئله** به شکل **گام به گام** بیان شود.



ویژگی‌های یک الگوریتم

- گام‌های الگوریتم باید **متناهی**، **شفاف**، **بدون ابهام** و **انجام‌پذیر** باشد.
- گام‌های الگوریتم باید **موثر** (effective) باشند یعنی هر گام باید کار مشخصی انجام دهد و بی‌تاثیر نباشد.
- **ورودی** و **خروجی** الگوریتم باید **خوش تعریف** (well-defined) باشد.
- الگوریتم می‌تواند ورودی نداشته باشد ولی **حداقل یک خروجی** باید داشته باشد.
- الگوریتم باید **قطعیت** (deterministic) داشته باشد؛ یعنی به ازای ورودی ثابت، همواره خروجی ثابتی ارائه دهد.

مزایا و معایب الگوریتم

مزایای الگوریتم

- نوشتن الگوریتم برنامه‌نویسی را ساده‌تر می‌کند.
- راه‌حل تقریباً به **زبان طبیعی** نوشته شده و **درک فرآیند** برای همه ساده است.
- فرآیند به صورت **گام به گام** نوشته شده و **درک مسائل پیچیده** نیز راحت است.

معایب الگوریتم

- الگوریتم‌نویسی نسبتاً به **زمان زیادی** نیاز دارد.
- **درک مسائل پیچیده** از روی الگوریتم می‌تواند سخت باشد.
- در الگوریتم، مشخص کردن حلقه **تکرار** (loop) و **انشعاب** (branching)، سخت است.

نیازمندی‌های طراحی الگوریتم

- داشتن **تعریف دقیق و واضح** از مسئله‌ای که باید حل شود (**problem**).
- مشخص بودن **محدودیت‌های مسئله** (**constraints**).
- مشخص شدن **ورودی‌های مورد نیاز** برای حل مسئله (**input**).
- مشخص بودن **خروجی‌های مورد انتظار** از الگوریتم (**output**).
- مشخص شدن یک **راه‌حل** برای ایجاد خروجی مورد انتظار با در نظر داشتن محدودیت‌های مسئله (**solution**).

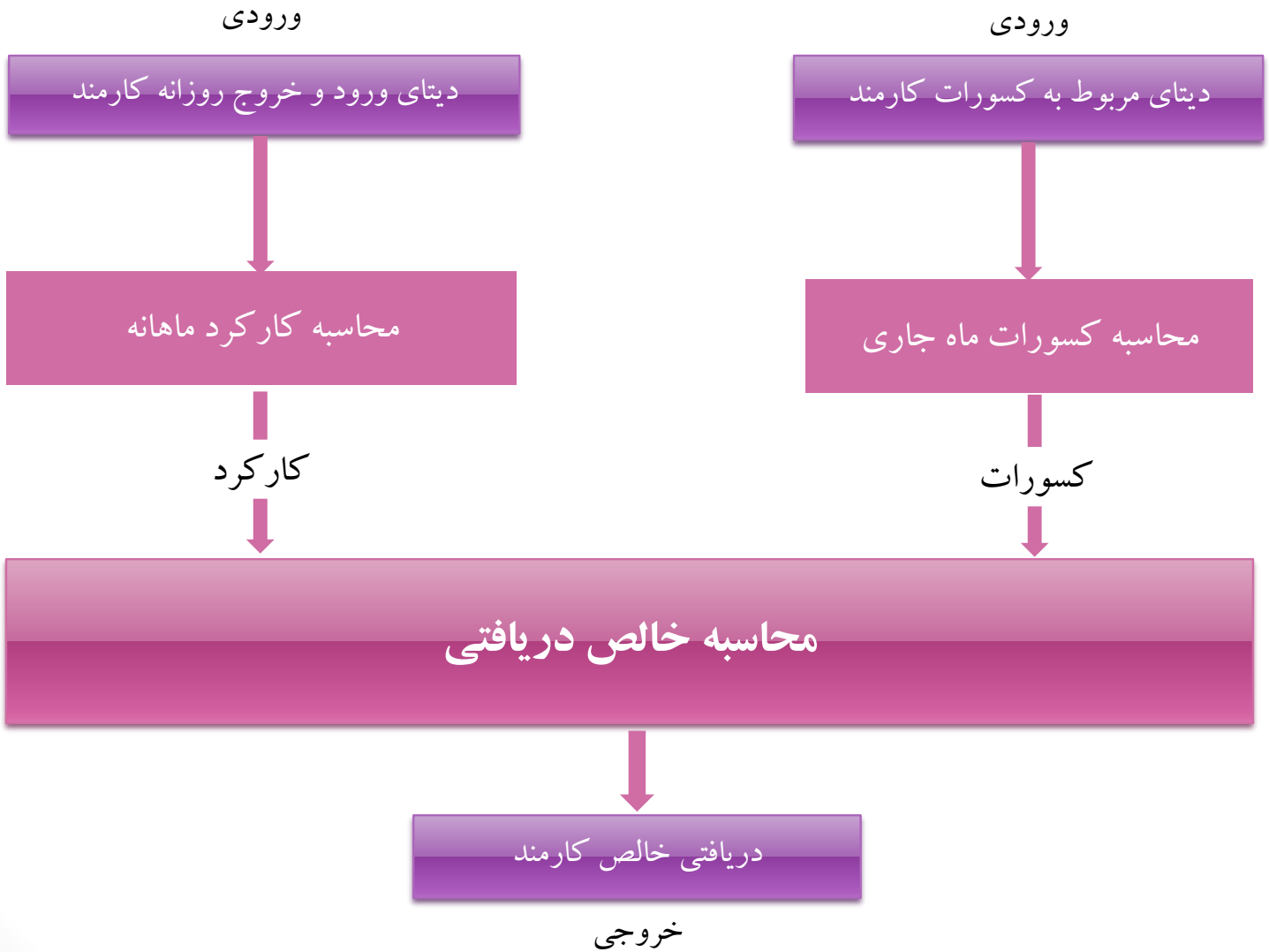
مشکلات رایج در مسائل بزرگ

- سختی **کدنویسی**
- سختی **نگهداری** سورس کد
- سختی تشخیص و تصحیح **خطا**

راهکار : آنالیز مسئله

- یک **مسئله بزرگ** قبل از اینکه حل شود ابتدا باید **آنالیز** شده و به **زیرمسئله** (ماژول) های جداگانه تقسیم شود.
- سپس برای هر **زیرمسئله**، راه حل ارائه می‌شود و مراحل طراحی الگوریتم، پیاده سازی و ... انجام می‌شود.
- به این سبک توسعه، **برنامه نویسی ماژولار** (modular programming) گفته می‌شود که مشکلات توسعه مسائل بزرگ را برطرف می‌کند.

برنامه نویسی ماژولار (modular programming)



جداسازی ماژول‌های یک برنامه حسابداری

۱. ماژول محاسبه کارکرد ماهانه
۲. ماژول محاسبه کسورات
۳. ماژول محاسبه خالص دریافتی

روش‌های طراحی الگوریتم

بازگشتی (Recursive)

- به طور متوالی، هر مسئله به زیرمسائلی از همان نوع تقسیم می‌شود تا به آخرین زیر مسئله برسد.
- سپس به طور سلسله مراتبی، برای حل هر مسئله از پاسخ زیرمسائل آن استفاده می‌شود.

تقسیم و غلبه (Divide and Conquer)

- مسئله بزرگ و پیچیده ابتدا به زیرمسائل کوچکتر مستقل از همان نوع تقسیم می‌شود.
- سپس با حل مسائل کوچک، مسئله اصلی حل می‌شود.

برنامه‌نویسی پویا (Dynamic Programming)

- نتایج گذشته برای استفاده در آینده جمع‌آوری می‌شوند.
- این راه حل مانند تقسیم و غلبه عمل می‌کند و برای حل مسائل بهینه‌سازی کاربرد دارد.

روش‌های طراحی الگوریتم

حریصانه (Greedy)

- این نوع الگوریتم در هر مرحله بهترین راه‌حل را انتخاب می‌کند (بهینگی محلی) و به بهینه شدن کل مسئله فکر نمی‌کند.
- این نوع الگوریتم نیز برای حل مسائل بهینه‌سازی مناسب است.

فراگیر (Brute Force)

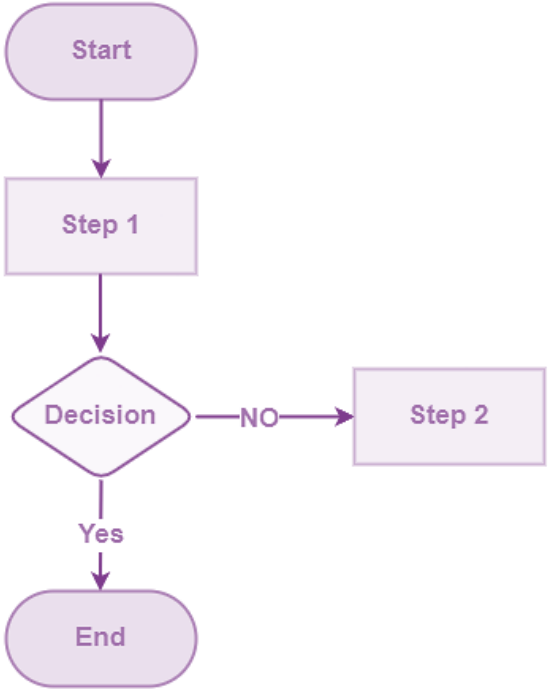
- این نوع الگوریتم، تمام راهکارهای ممکن را یکبار بررسی می‌کند و راهکاری که بهتر باشد را انتخاب می‌کند.
- این استراتژی بسیار ساده بوده و به "just do it" معروف است.

عقب‌گرد (Backtracking)

- این نوع الگوریتم یک مکانیزم جستجو در عمق را دنبال کرده و در هر مرحله از جستجو اگر شرط برقرار نشود، به عقب برگشته و مسیر دیگری را پیش می‌گیرد.

فلوچارت چیست؟

- **فلوچارت** یک ابزار گرافیکی برای نمایش فرآیند **الگوریتم** و گام‌های انجام آن به صورت **تصویری** و به کمک **اشکال هندسی** می‌باشد.
- فلوچارت از یک سری باکس‌های هندسی و خطوط متصل کننده تشکیل می‌شود.



ترمینال : start , end

ورودی/خروجی : input , output

دستورات یا گام‌های پردازشی: instruction

تصمیم گیری: if-else

مزایا و معایب فلوچارت

مزایای فلوچارت

- فلوچارت معمولاً **سریع‌تر** آماده می‌شود.
- **ساده‌سازی نمایش منطق مسئله** (با نمایش گرافیکی گام‌های حل مسئله).
- **بهبود و سهولت ارتباطات بین تیمی** (ساده‌تر شدن ارائه راهکارها بین اعضای تیم).
- **مستندسازی** (فلوچارت کمک می‌کند تا مستندات کامل‌تر و با کیفیت‌تری برای محصول تولید کنیم).

معایب فلوچارت

- **دیباگ و اشکال‌یابی** از روی فلوچارت نسبتاً سخت‌تر است.
- ترسیم فلوچارت برای **مسائل پیچیده** نسبتاً سخت‌تر است.

شبه‌کد چیست؟

- شبه‌کد (Pseudo code) یکی از روش‌های **نمایش** گام‌های **الگوریتم** به **شیوه‌ای نزدیک‌تر به کدنویسی** می‌باشد.
- در نوشتن شبه کد، هیچ زبان خاصی استفاده نمی‌شود و صرفاً یک شیوه نگارش مشابه با کدنویسی است.

```
procedure largest_array(A)

   Declare largest as integer
   Set largest to 0
   FOR EACH value in A DO
      IF A[n] is greater than largest THEN
         largest ← A[n]
      ENDIF
   END FOR
   Display largest

end procedure
```

مزایای شبه‌کد

مزایای شبه‌کد

- تبدیل **شبه‌کد** به **سورس‌کد** توسط برنامه‌نویس با صرف **زمان کمتری** انجام خواهد بود.
- برای **مسائل پیچیده**، نوشتن شبه‌کد نسبتاً **ساده‌تر** از روش‌های دیگر توصیف الگوریتم است.
- **تحلیل** و **آنالیز پیچیدگی الگوریتم** مسئله از روی شبه‌کد، **ساده‌تر**، **بهینه‌تر** و **موثرتر** انجام می‌شود.
- **اشکال‌زدایی** (debug) شبه‌کد و یافتن خطاهای احتمالی، نسبتاً **ساده‌تر** است.

مثال ۱: الگوریتم جمع سه عدد

نیازمندی‌ها

- **تعریف دقیق و واضح :** جمع ۳ عدد و پرینت حاصل‌جمع در خروجی.
- **محدودیت های مسئله** : اعداد باید فقط حاوی ارقام باشند و نمی توانند حاوی کاراکترهای دیگر باشند.
- **ورودی های مورد نیاز** : سه عدد با فرمت صحیح (مطابق با محدودیت های مسئله).
- **خروجی‌های مورد انتظار** : یک عدد صحیح که از حاصل جمع ۳ عدد ورودی مسئله بدست آمده است.
- **راه حل مسئله** : جمع کردن ۳ عدد ورودی از طریق عملگر + بصورت a+b+c و ذخیره نتیجه در متغیر sum.

مثال ۱: الگوریتم جمع سه عدد

الگوریتم

1- **START**

2- **Declare** 3 integer variables **num1**, **num2** and **num3**

3- **Take** the 3 numbers, to be added, as **inputs** in variables **num1**, **num2**, and **num3** respectively.

4- **Declare** an integer variable **sum** to store the resultant sum of the 3 numbers.

5- **Add** the 3 numbers and store the result in the variable **sum**.

6- **Print** the value of the variable **sum**

7- **END**

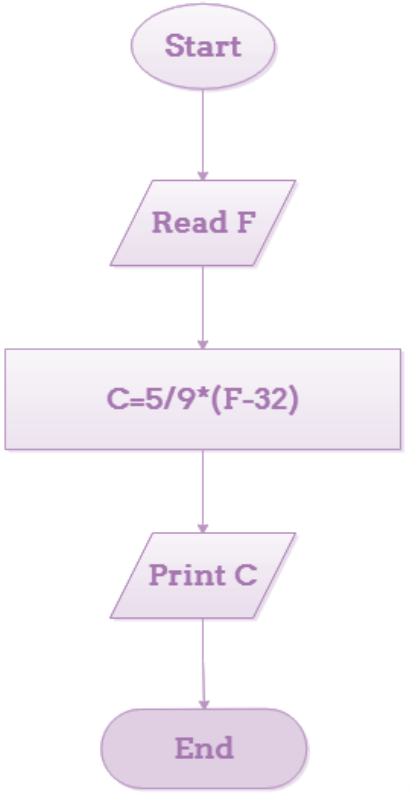
مثال ۲: الگوریتم تبدیل دمای فارنهایت به سلسیوس

Step 1: Read temperature in **Fahrenheit** (**F**),

Step 2: Calculate temperature with formula :

C=5/9*(F-32)

Step 3: Print **C**.



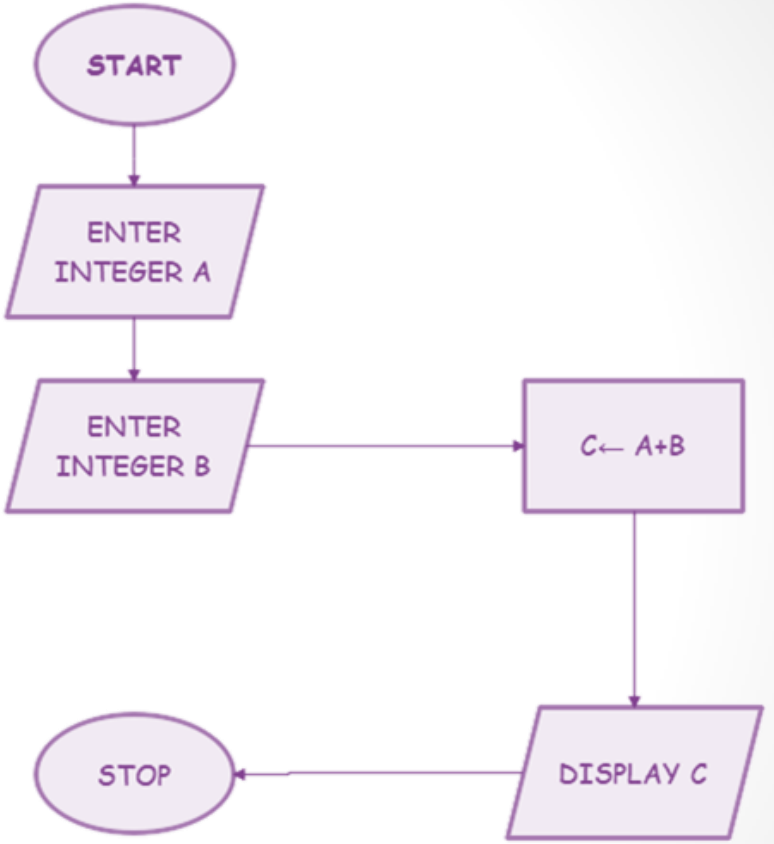
مثال ۳: الگوریتم جمع دو عدد

Step 1: Read the Integer **A**.

Step 2: Read Integer **B**.

Step 3: Perform the addition by using the formula: **C= A + B.**

Step 4: Print the Integer **C**.



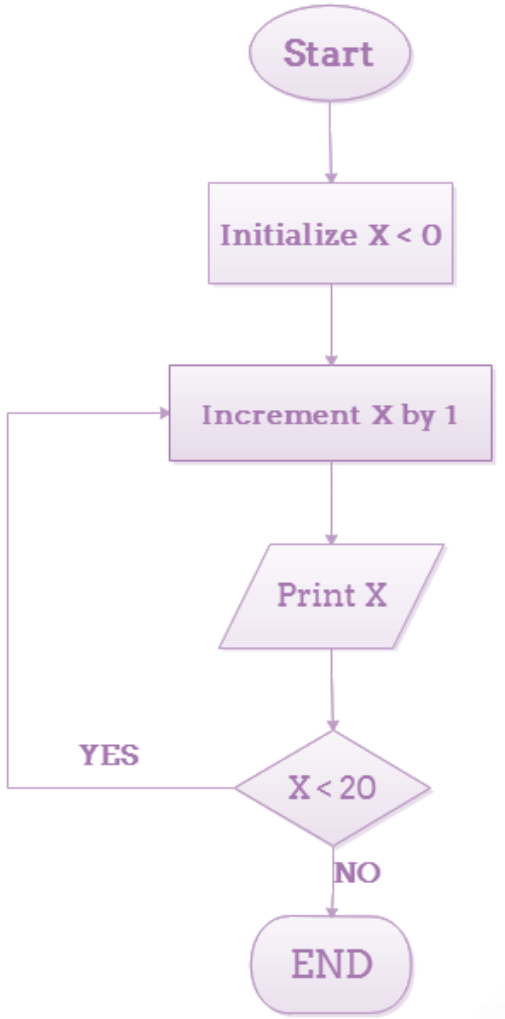
مثال ٤: الگوریتم پرینت اعداد ۱ تا ۲۰

Step 1: Initialize **X** as 0,

Step 2: Increment **X** by 1,

Step 3: Print **X**,

Step 4: If **X** is less than **20** then go back to step 2.



مثال ۵: الگوریتم جمع اعداد ۱ تا ۵۰

Step 1: Declare number **N= 0** and **sum= 0**

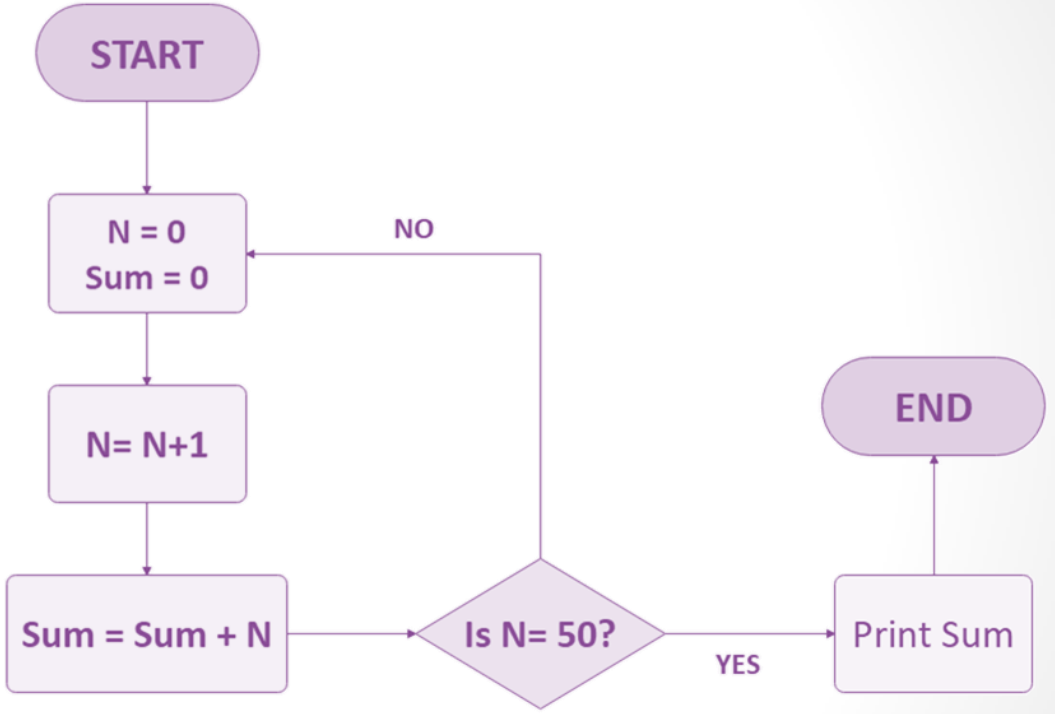
Step 2: Determine N by **N= N+1**

Step 3: Calculate the sum by the formula:

Sum= N + Sum.

Step 4: Add a loop between steps 2 and 3 until **N= 50**.

Step 5: Print **Sum**.



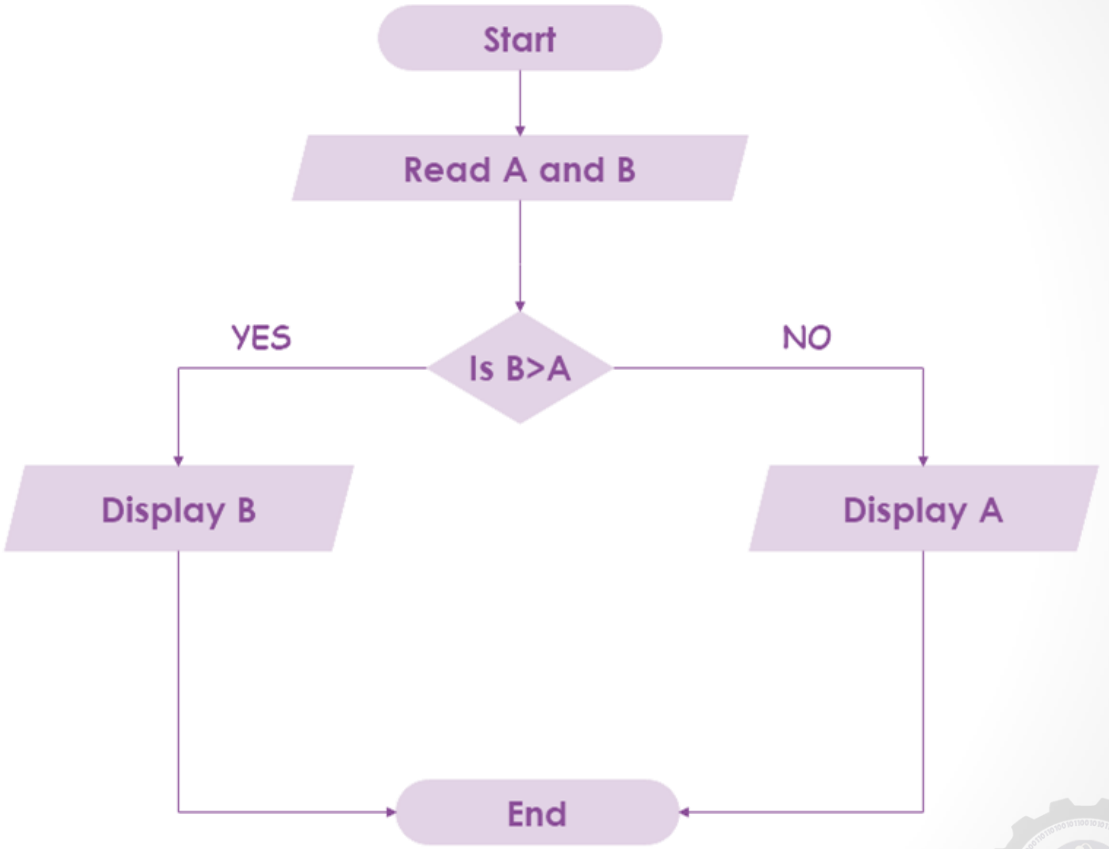
مثال۶: الگوریتم پیداکردن max دو عدد

Step 1: Read the Integer **A**.

Step 2: Read Integer **B**.

Step 3: If **B** is greater than **A**, then print **B**,

else print **A**.



مثال ۷: الگوریتم تشخیص زوج و فرد

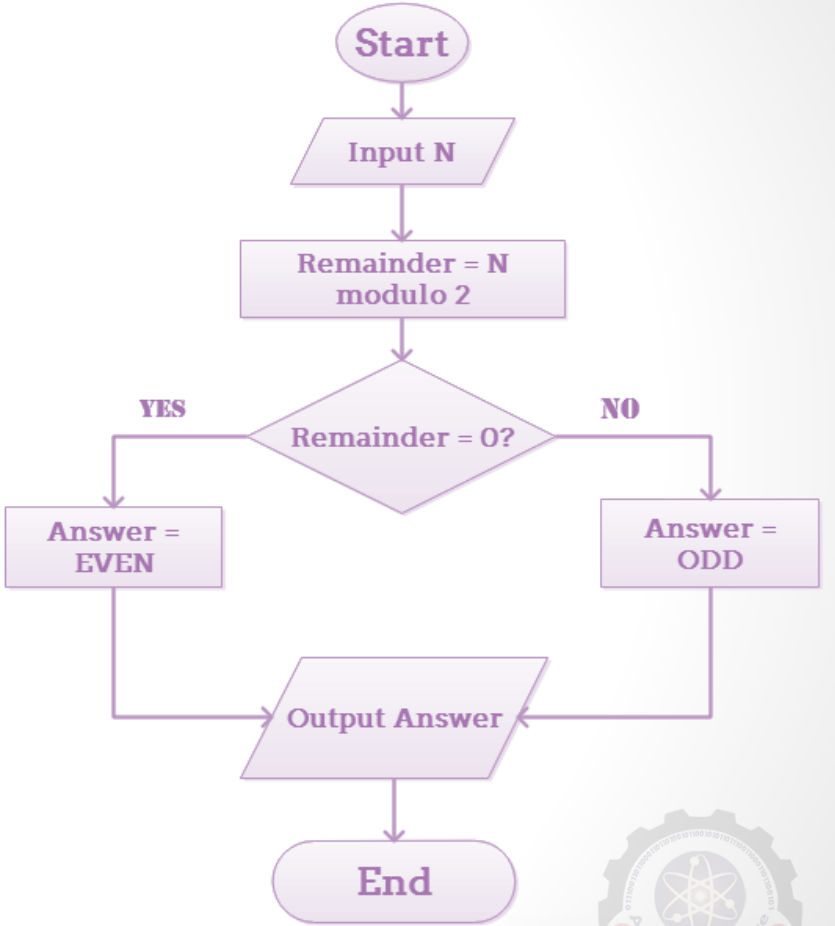
Step 1: Read number **N**.

Step 2: Set **remainder** as **N modulo 2**.

Step 3: **If** the **remainder** is equal to **0** then number **N** is **even**,

else number **N** is **odd**.

Step 4: Print **output**.



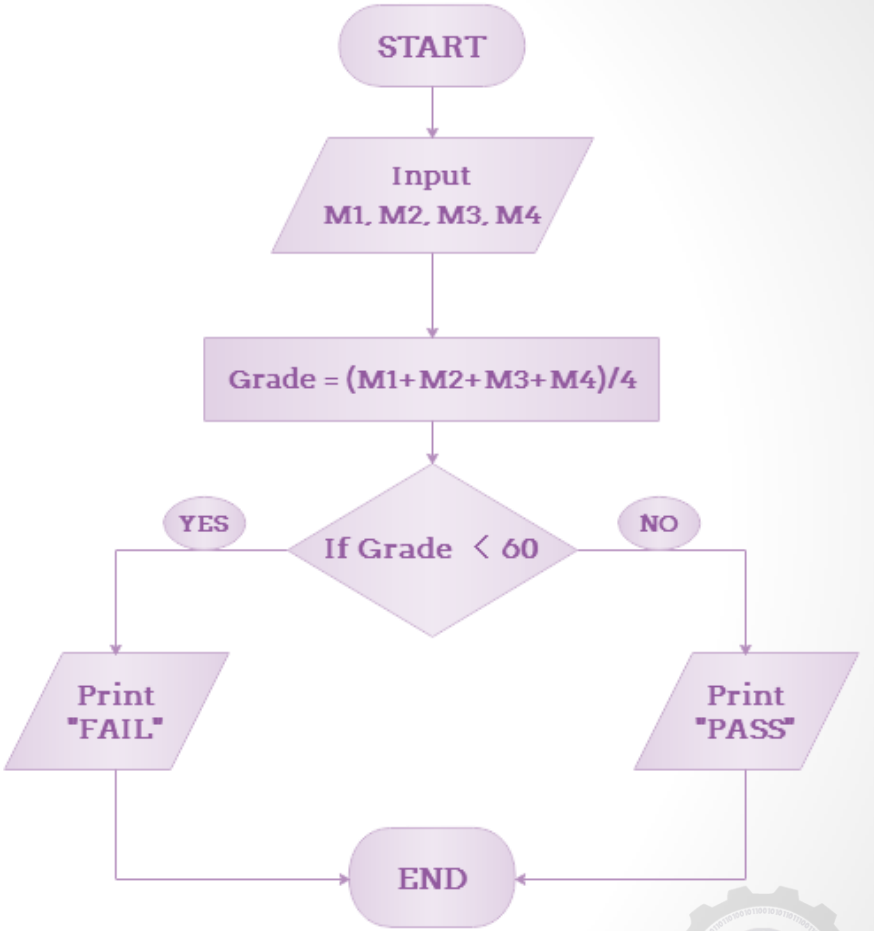
مثال۸: الگوریتم تشخیص قبولی یا مردودی

Step 1: Input **grades** of 4 courses **M1**, **M2**, **M3** and **M4**,

Step 2: Calculate the **average** grade with formula:

$$\textbf{Grade}=(M1+M2+M3+M4)/4$$

Step 3: If the **average** grade is less than **60**, print "**FAIL**", else print "**PASS**".



مثال۹: محاسبه اعداد دنباله فیبوناچی

هر عدد در سری فیبوناچی، با جمع دو عنصر قبلی ایجاد می شود. دو عنصر اول سری فیبوناچی را می توان 0,1 یا 1,1 در نظر گرفت.

$$F_n = F_{n-1} + F_{n-2}$$

$F_8 = 0\ 1\ 1\ 2\ 3\ 5\ 8\ 13$

Step 1: Declare the variables **i**, **a**, **b**, **show**.

Step 2: Enter the values for the variables, **a=0**, **b=1**, **show=0**

Step 3: Enter the **terms of the Fibonacci series** to be printed, i.e=, **1000**.

Step 4: Print **the first two terms** of the series.

Step 5: Loop the following steps:

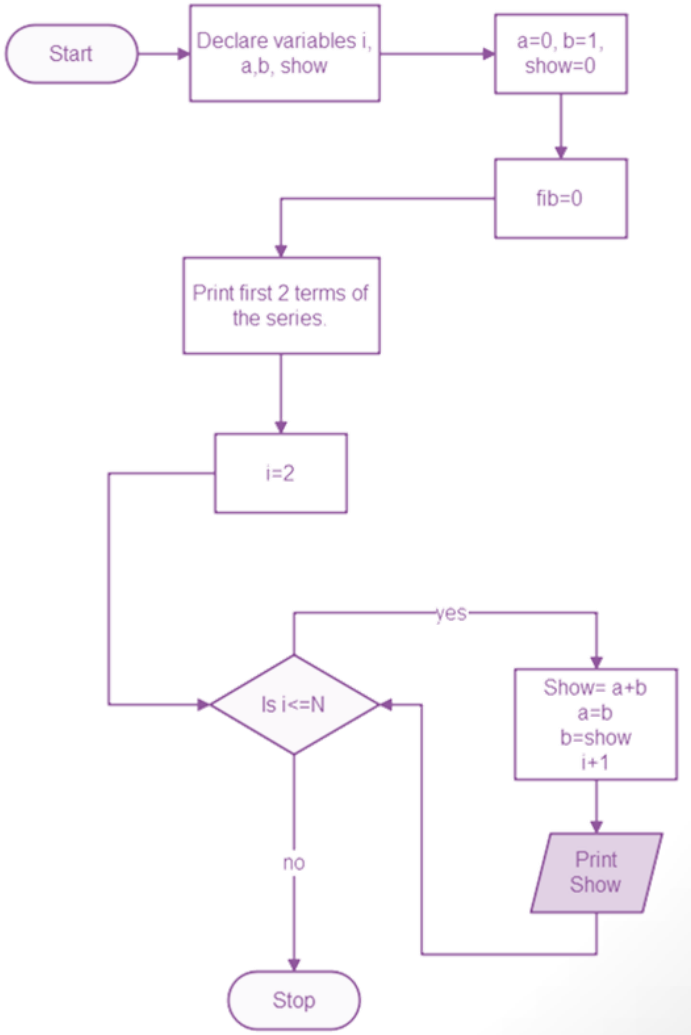
Show = **a** + **b**

a= **b**

b = **show**

Add **1** to the value of **i** each time.

Print **Show**



مثال ۱۰: پیدا کردن max از بین ۱۰۰ قیمت و کاهش ۱۰٪ از آن

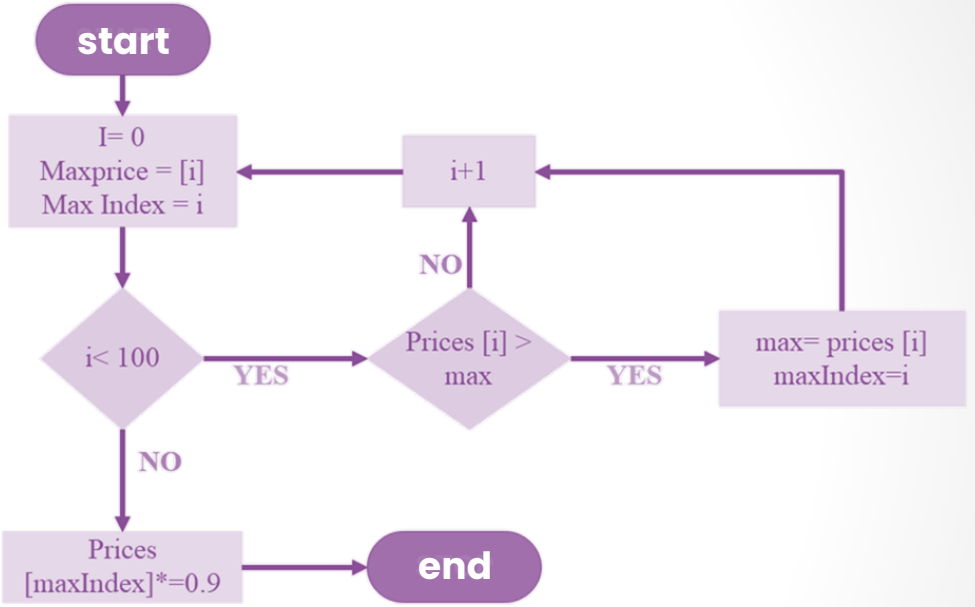
Step 1: Read the 100 **prices**.

Step 2: Compare the first **price** with the next and let the greater of the two be '**max**' in the **maxIndex**.

Step 3: Loop it until the largest price has been found.

Step 4: Reduce the '**max**' value by **10%** using the formula:

prices [max index] = prices [max index] x 0.9



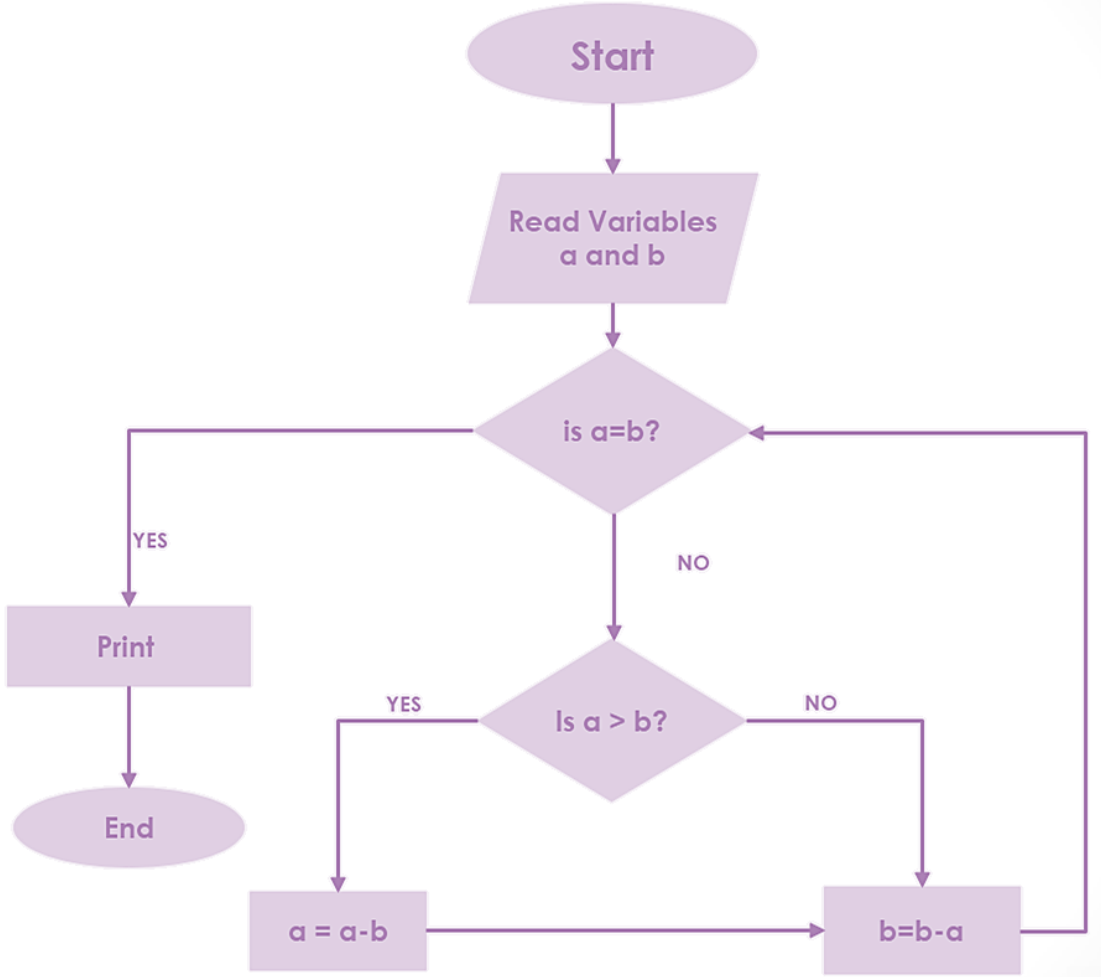
مثال ۱۱: بزرگترین خارج قسمت مشترک (HCF)

Step 1: Read the variables **a** and **b**.

Step 2: If **a = b**, go to step 4.

Step 3: If **a > b**, then: **a = a - b**. Return to step 2.

Step 4: Print **a or b**



12 → 1, 2, 3, 4, 6, 12

16 → 1, 2, 4, 8, 16

مثال ۱۲: کوچکترین مضرب مشترک (LCM)

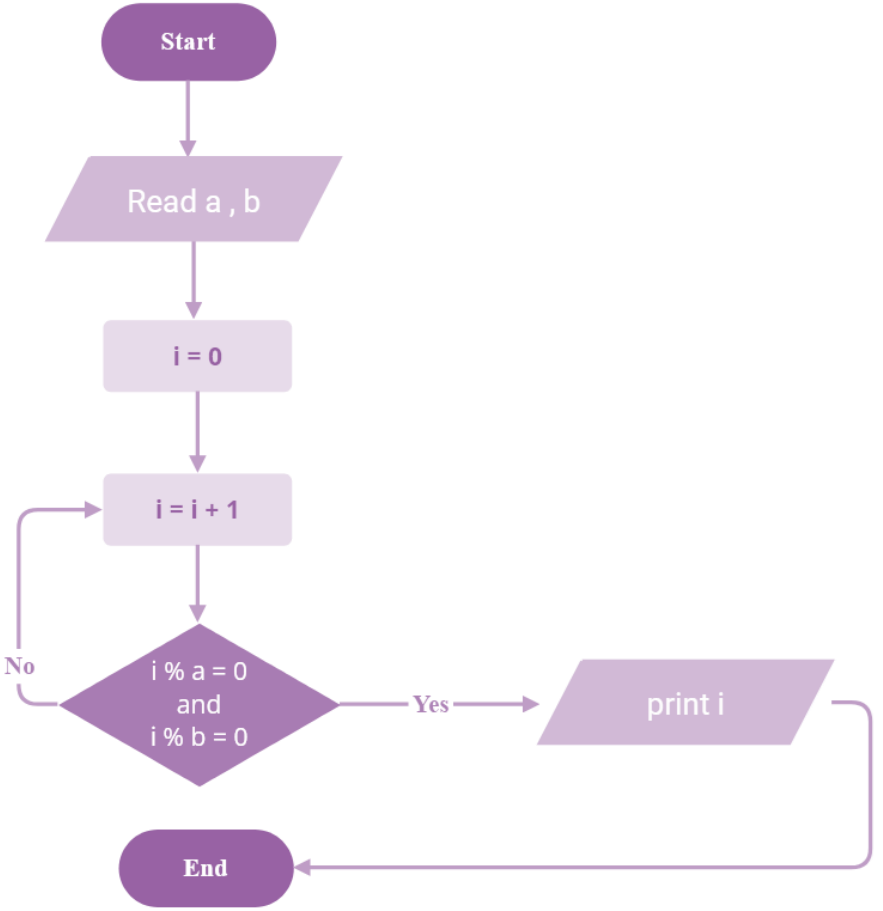
Step 1: Read the variables **a**, **b** and set **i = 0**

Step 2: **i= i +1**

Step 3: If **i % a** is **0** and **i % b** is **0** go to step 4

else Return to step 2.

Step 4: Print **i**



3 → 3, 6, 9, **12**, 15 ...

4 → 4, 8, **12**, 16, 20 ...

https://www.tutorialspoint.com/learn_c_by_examples/index.htm

https://www.tutorialspoint.com/data_structures_algorithms/index.htm

https://www.tutorialspoint.com/design_and_analysis_of_algorithms/index.htm

https://www.geeksforgeeks.org/introduction-to-algorithms/

https://www.w3schools.com/c/tryc.php?filename=demo_compiler

اصول و مبانی برنامه‌نویسی

مجید شبیری

کارشناسی ارشد IT، گرایش شبکه

از دانشگاه صنعتی امیرکبیر