اصول کامپیوتر ۱

مبانی کامپیوتر و برنامه‌سازی

«جلسه‌ی دوازدهم»

دانشکده‌ی علوم ریاضی - دانشگاه شهید بهشتی

نیم‌سال اول ۹۰-۱۳۸۹

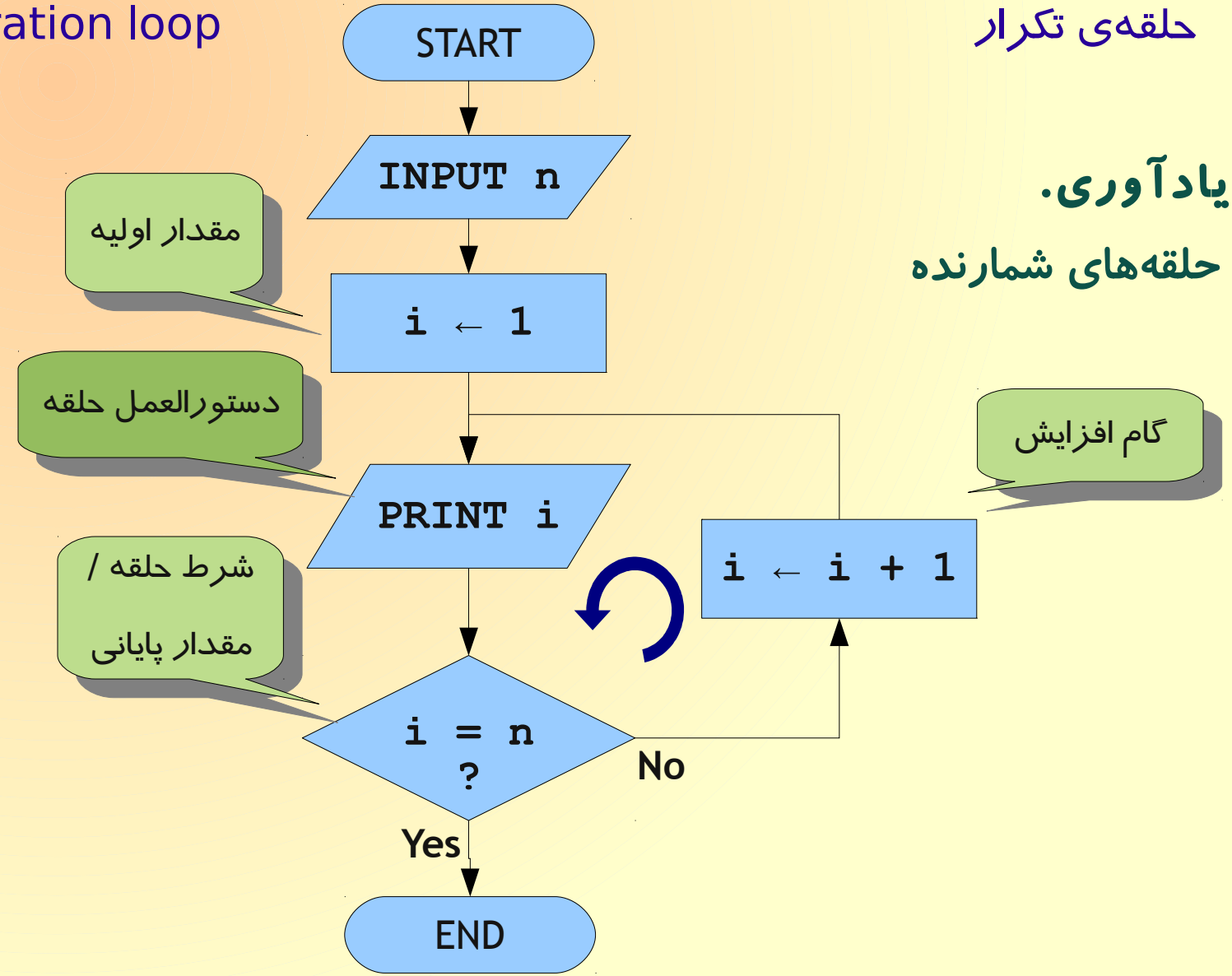
مدرس: سید علی کتان‌فروش

Iteration loop

حلقه‌ی تکرار

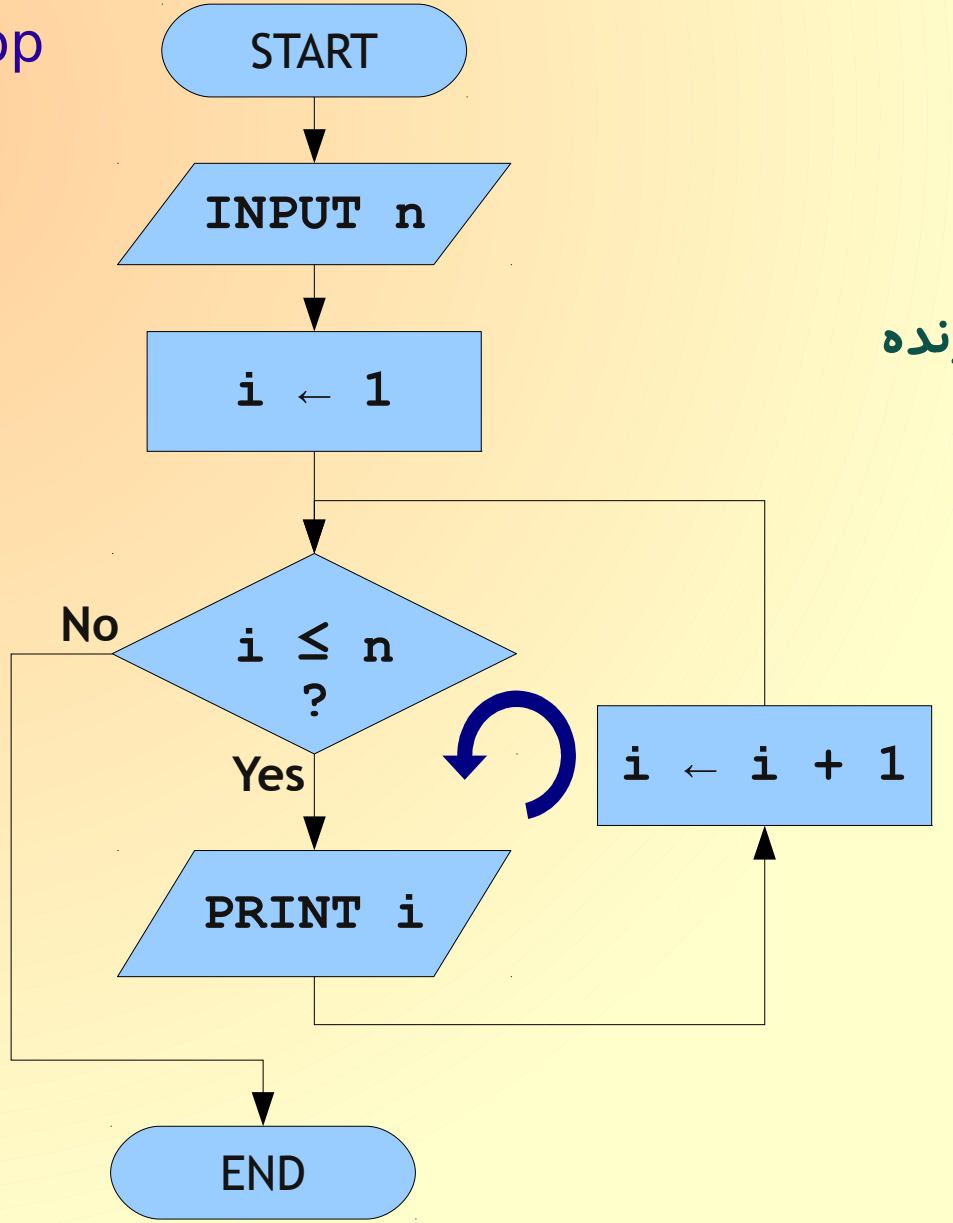
یادآوری.

حلقه‌های شمارنده



Iteration loop

حلقه‌ی تکرار



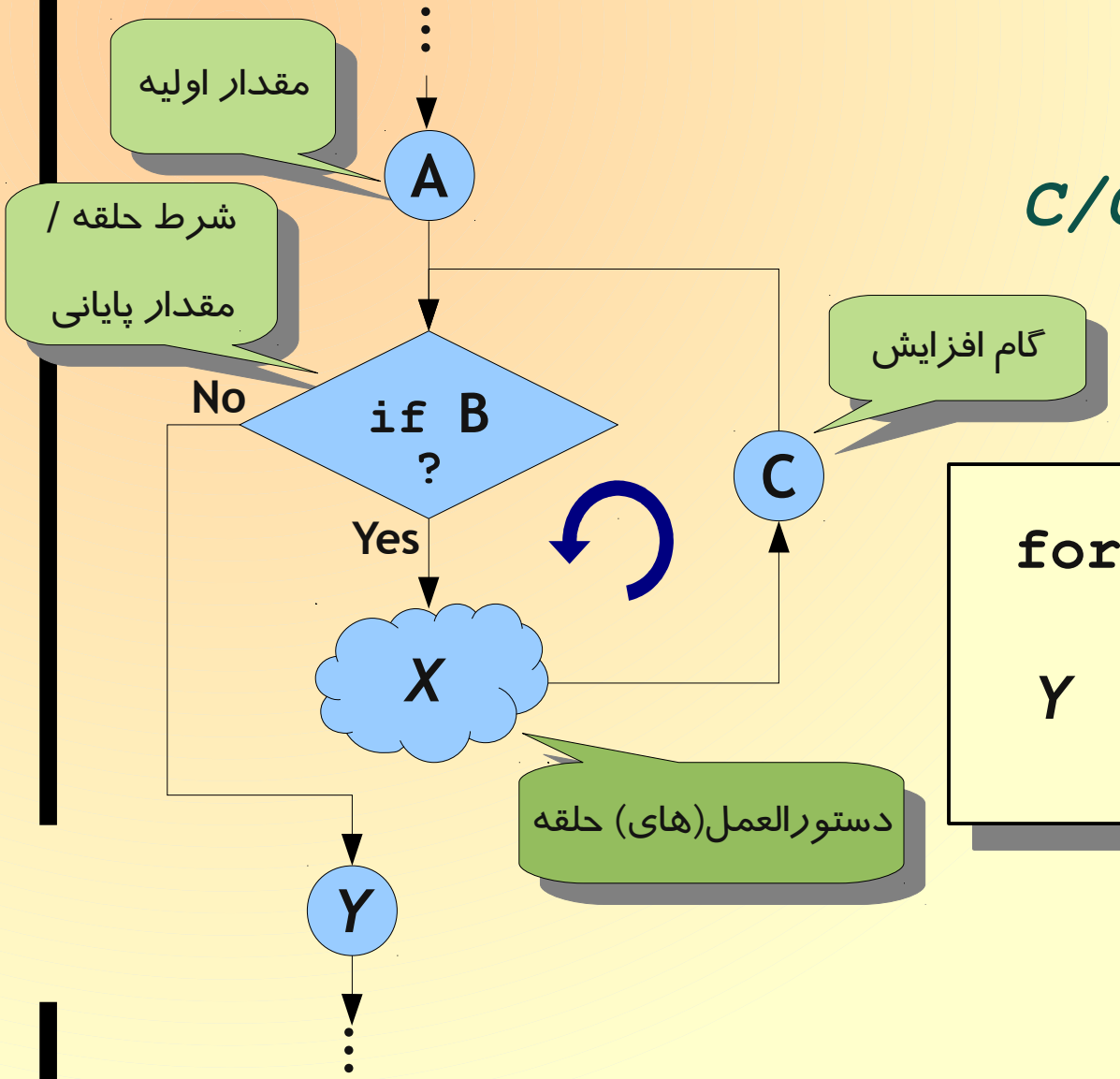
یادآوری.

حلقه‌های شمارنده

الگوی C/C++

Iteration loop

حلقه‌ی تکرار



یادآوری.

حلقه‌ی for در C/C++

```
for ( A; B; C )
    X;
Y
```

مثال. برنامه‌ای بنویسید که حاصلجمع اعداد صحیح ۱ تا n را محاسبه کند.

$$S = 1 + 2 + \cdots + n$$

با استفاده از نماد سیگما داریم:

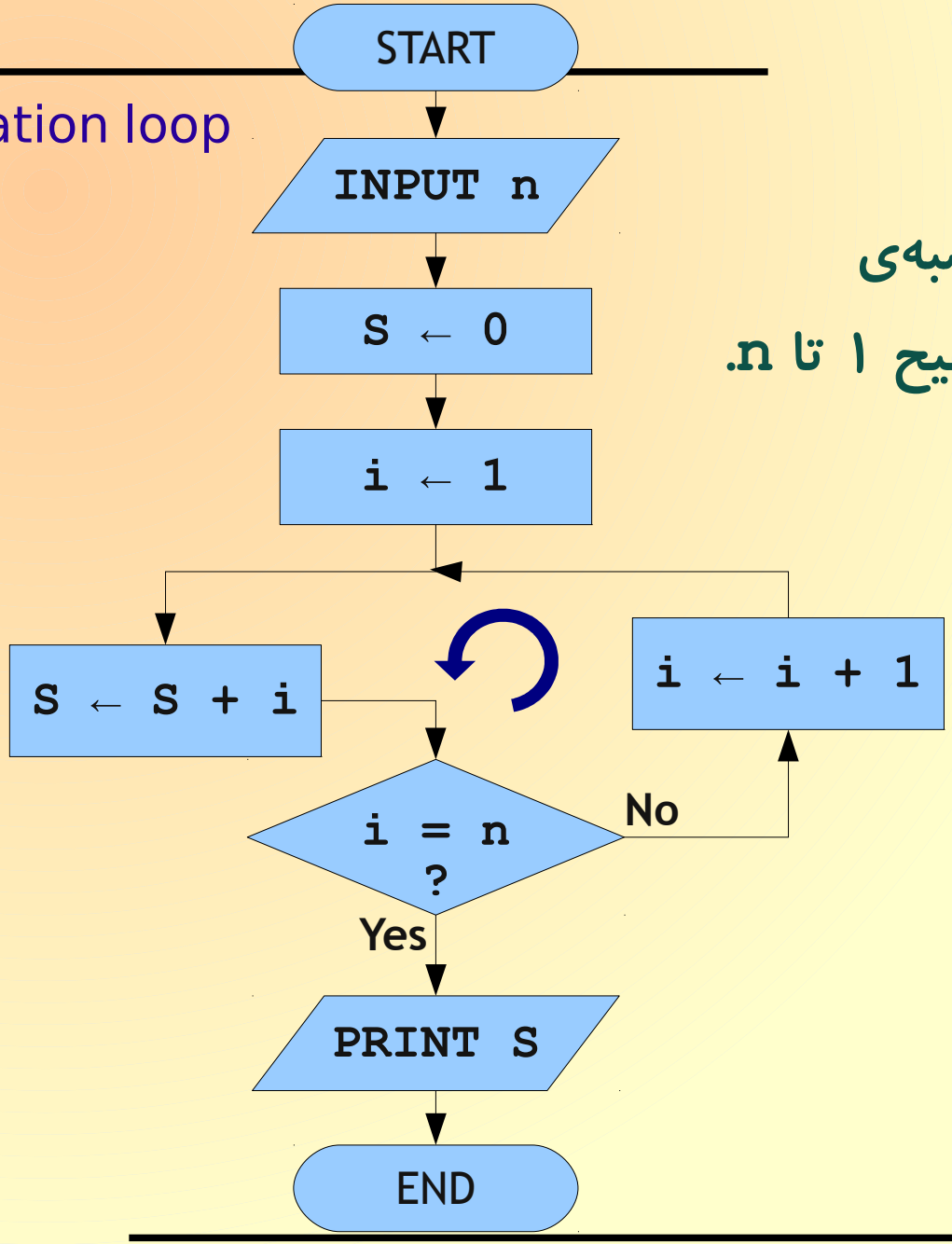
$$S = \sum_{i=1}^{n} i$$

Iteration loop

حلقه‌ی تکرار

مثال. الگوریتم محاسبه‌ی

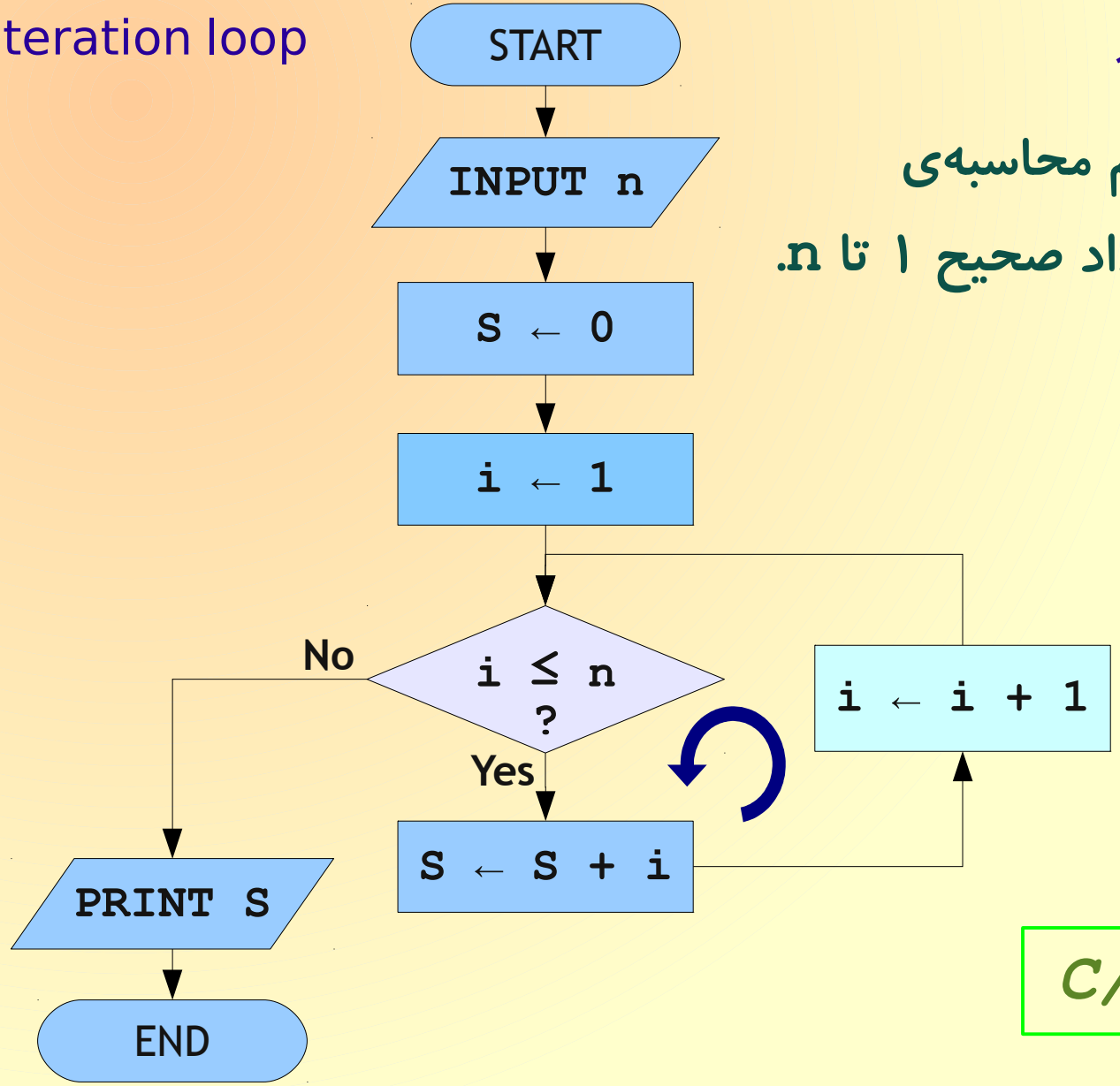
حاصلجمع اعداد صحیح ۱ تا n.



Iteration loop

حلقه‌ی تکرار

مثال. الگوریتم محاسبه‌ی حاصلجمع اعداد صحیح ۱ تا n.



الگوی C/C++

Iteration loop

حلقه‌ی تکرار

مثال. برنامه‌ای بنویسید که حاصلجمع اعداد صحیح ۱ تا n را محاسبه کند.

راه حل ۱

```
int main( int argc, char *argv[]){
  int n;
  cout << " n ? ";
  cin >> n;

  int i,S;
  S = 0;
  for ( i=1; i<=n; i=i+1 )
    S = S + i;

  cout << "Sum = " << S << endl;
  ....
```

چرا حلقه‌ی تکرار؟

رابطه‌ی ساده‌ای برای محاسبه‌ی

مجموع اعداد ۱ تا n وجود دارد.

Iteration loop

حلقه‌ی تکرار

مثال. برنامه‌ای بنویسید که حاصلجمع اعداد صحیح ۱ تا n را محاسبه کند.

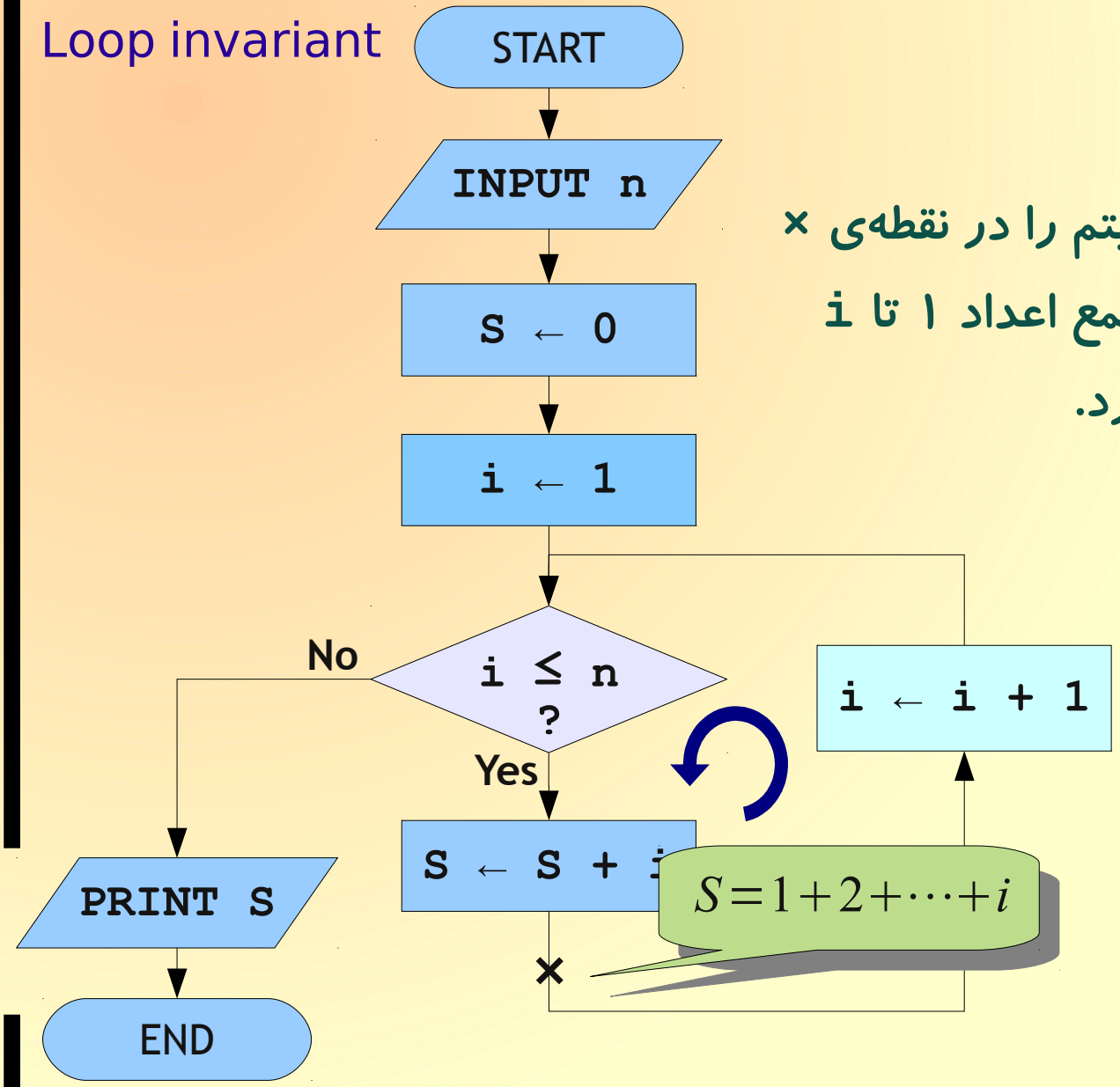
راه حل ۲

```
int main( int argc, char *argv[]){
  int n;
  cout << " n ? ";
  cin >> n;

  int S;
  S = n * (n+1) / 2;

  cout << "Sum = " << S << endl;

  ....
```

Loop invariant

ناوردای حلقه‌ی

هرگاه اجرای الگوریتم را در نقطه‌ی × متوقف کنیم حاصلجمع اعداد ۱ تا i در متغیر S قرار دارد.



Loop invariant

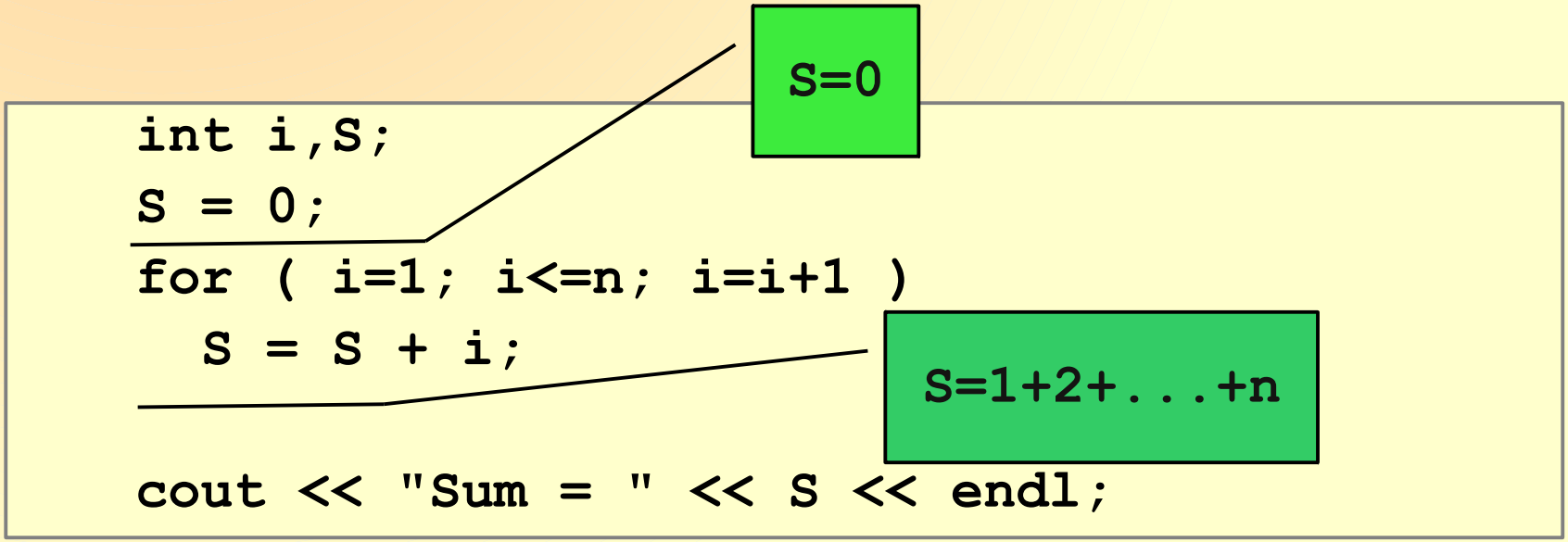
ناوردای حلقه‌ی

- گزاره‌ای را که در انتهای دستورالعمل(های) حلقه، همواره ارزش درست داشته باشد را اصطلاحاً ناوردای حلقه (Loop invariant) می‌نامند.
- در برنامه‌ی محاسبه‌ی مجموع اعداد ۱ تا n، گزاره‌ی «S مجموع اعداد ۱ تا i است» یک ناوردای حلقه است.

```
int i,S;
S = 0;
for ( i=1; i<=n; i=i+1 )
  S = S + i;

cout << "Sum = " <
```

$S = 1 + 2 + \cdots + i$

وضعیت متغیرها پیش از اجرای یک دستورالعمل (یا مجموعه‌ای از دستورات) را اصطلاحاً precondition و پس از آنرا اصطلاحاً postcondition می‌گوئیم.



- در مثال قبل، ناوردای حلقه پیش از اجرای حلقه نیز گزاره‌ای درست است.

$$S = 1 + 2 + \cdots + i$$

$$i = 0 \Rightarrow S = 0$$

- بدیهی است این گزاره پس از اجرای حلقه نیز همچنان برقرار است.

Loop invariant

ناوردای حلقه‌ی

در مسائلی که برای حل آن‌ها از حلقه‌های تکرار استفاده می‌کنید دو نکته را مورد توجه قرار دهید:

1. **رابطه‌ی مناسبی برای تعریف ناوردای حلقه بدست آورید.**
2. **مقادیر مناسبی برای نسبت‌دهی به متغیرهای مرتبط با ناوردای حلقه پیش از اجرای حلقه بیابید.**

مثال. برنامه‌ای بنویسید که عدد صحیح و نامنفی `n` را از کاربر دریافت کند و `n!` را محاسبه و چاپ کند.

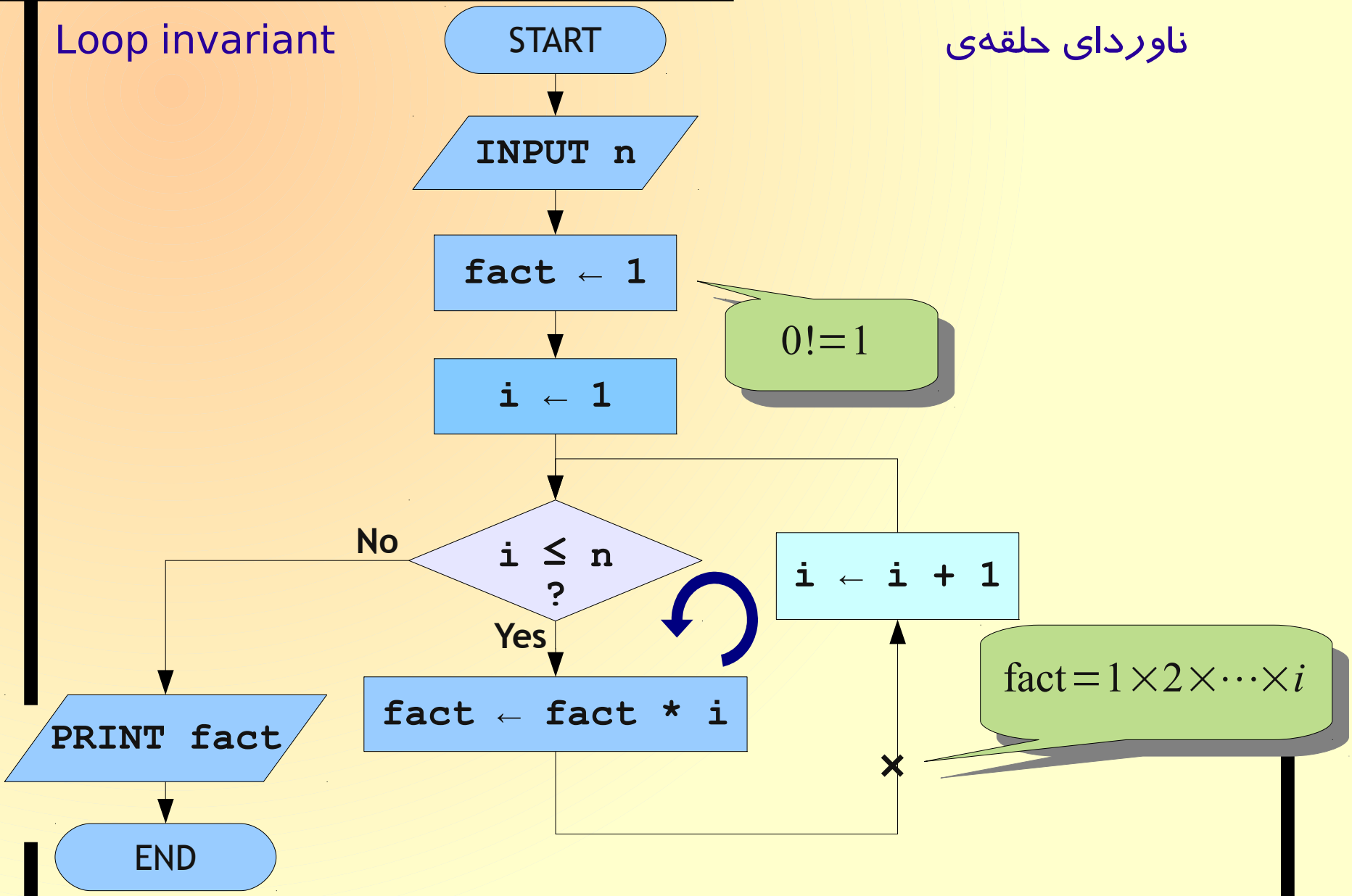
$$n! = 1 \times 2 \times \cdots \times n$$

با استفاده از نماد حاصلضرب داریم:

$$n! = \prod_{i=1}^{n} i$$

Loop invariant

ناوردای حلقه‌ی



Iteration loop

حلقه‌ی تکرار

مثال. برنامه‌ی محاسبه‌ی n!

```
int main( int argc, char *argv[]){
  int n;
  cout << " n ? ";
  cin >> n;

  int i,fact;
  fact = 1;
  for ( i=1; i<=n; i++ )
    fact = fact * i;

  cout << n << "! = " << fact << endl;
  ....
```

Iteration loop

حلقه‌ی تکرار

مثال. برنامه‌ای بنویسید که `n` عدد از کاربر دریافت کند و میانگین آن‌ها را محاسبه و چاپ کند. کاربر `n` را تعیین می‌کند.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

مثال. برنامه‌ی محاسبه‌ی میانگین.

```
int main( int argc, char *argv[]){
  int n;
  cout << " n ? ";
  cin >> n;

  double x,S;
  int i;
  S = 0;
  for ( i=0; i<n; i++ ) {
    cin >> x ;
    S = S + x;
  }
  cout << "Mean = " << S/n << endl;
  ....
```

مثال. برنامه‌ای بنویسید که n عدد از کاربر دریافت کند و ماکزیمم آن‌ها را محاسبه و چاپ کند. کاربر n را تعیین می‌کند.

$$Maximum = \max\{x_1, x_2, \ldots x_n\}$$

- **ناوردای حلقه**

$$Maximum = max\{x_1, x_2, \ldots, x_i\}$$

- **رابطه‌ی بهنگام‌سازی**

$$max\{x_1, x_2, \ldots, x_i\} = max\left\{x_i, max\{x_1, x_2, \ldots, x_{i-1}\}\right\}$$

- **مقدار اولیه در `precondition`**

$$max\{x_1\} = x_1$$

مثال. برنامه‌ی محاسبه‌ی ماکزیمم.

راه حل ۱

```
int main( ){
  int n;
  cout << " n ? ";
  cin >> n;

  double x,max;
  cin >> x;
  max = x;

  int i;
  for ( i=1; i<=n-1; i++ ) {
    cin >> x ;
    if ( max < x ) max = x;
  }
  cout << "MAX = " << max << endl;
  ....
```

مثال. برنامه‌ی محاسبه‌ی ماکزیمم.

راه حل ۱

```
int main( ){
  int n;
  cout << " n ? ";
  cin >> n;

  double x,max;
  cin >> x;
  max = x;

  int i;
  for ( i=1; i<n; i++ ) {
    cin >> x ;
    if ( max < x ) max = x;
  }
  cout << "MAX = " << max << endl;
  ....
```

- ناوردای حلقه

$$Maximum = max\{x_1, x_2, \ldots, x_i\}$$

- رابطه‌ی بهنگام‌سازی

$$max\{x_1, x_2, \ldots, x_i\} = max\{x_i, max\{x_1, x_2, \ldots, x_{i-1}\}\}$$

- مقدار اولیه در `precondition`

$$max\{\} = -\infty$$

مثال. برنامه‌ی محاسبه‌ی ماکزیمم.

راه حل ۲

```
int main( int argc, char *argv[]){
  int n;
  cout << " n ? ";
  cin >> n;

  double x,max;
  int i;
  max = - DBL_MAX;
  for ( i=0; i<n; i++ ) {
    cin >> x ;
    if ( max < x ) max = x;
  }
  cout << "MAX = " << max << endl;
  ....
```

مثال. برنامه‌ای محاسبه‌ی ماکزیمم.

راه حل ۲

```
int main( int argc, char *argv[]){
  int n;
  cout << " n ? ";
  cin >> n;

  double x,max;
  int i;
  max = - DBL_MAX;
  for ( i=0; i<n; i++ ) {
    cin >> x ;
    if ( max < x ) max = x;
  }
  cout << "MAX = " << max << endl;
  ....
```

DBL_MAXبزرگترین عدد اعشاری قابل نمایش در کامپیوتر با دقت مضاعف است

#include <cfloat>

مثال. برنامه‌ای محاسبه‌ی ماکزیمم. با فرض اینکه اعداد داده شده از نوع اعداد صحیح هستند.

```
int main( ){
  int n;
  cout << " n ? ";
  cin >> n;

  int x,max;
  int i;
  max = INT_MIN;
  for ( i=0; i<n; i++ ) {
    cin >> x ;
    if ( max < x ) max = x;
  }
  cout << "MAX = " << max << endl;
  ....
```

INT_MINکوچکترین عدد صحیح قابل نمایش در کامپیوتر است

#include <climits>

Iteration loop

حلقه‌ی تکرار

مسئله. برنامه‌ای بنویسید که مقسوم‌علیه‌های عدد داده شده‌ی n را چاپ کند.

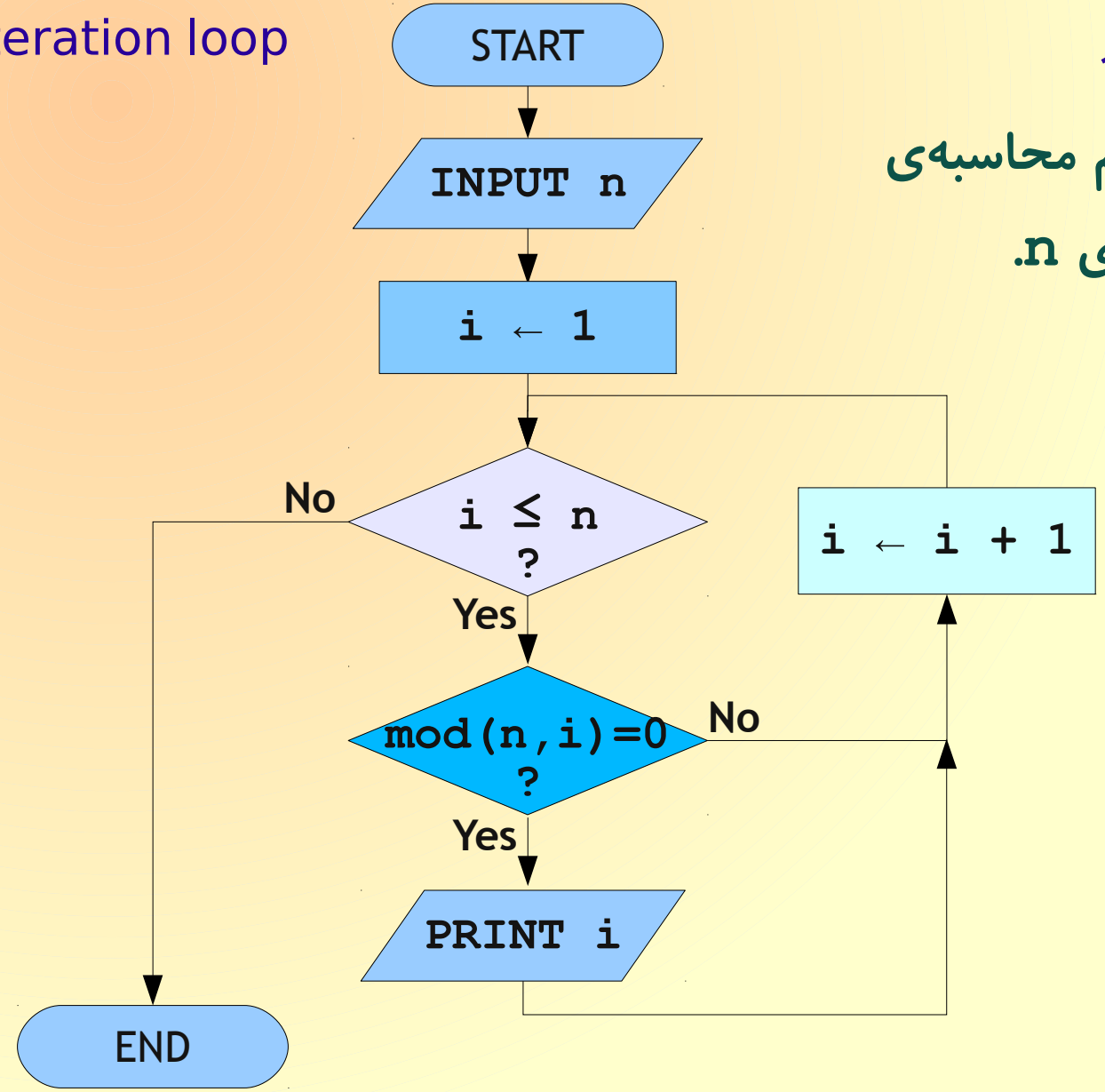
مثال. به ازای n=24 داریم 1,2,3,4,6,8,12,24

و به ازای n=25 داریم 1,5,25

Iteration loop

حلقه‌ی تکرار

مثال. الگوریتم محاسبه‌ی مقسوم‌علیه‌های n.



Iteration loop

حلقه‌ی تکرار

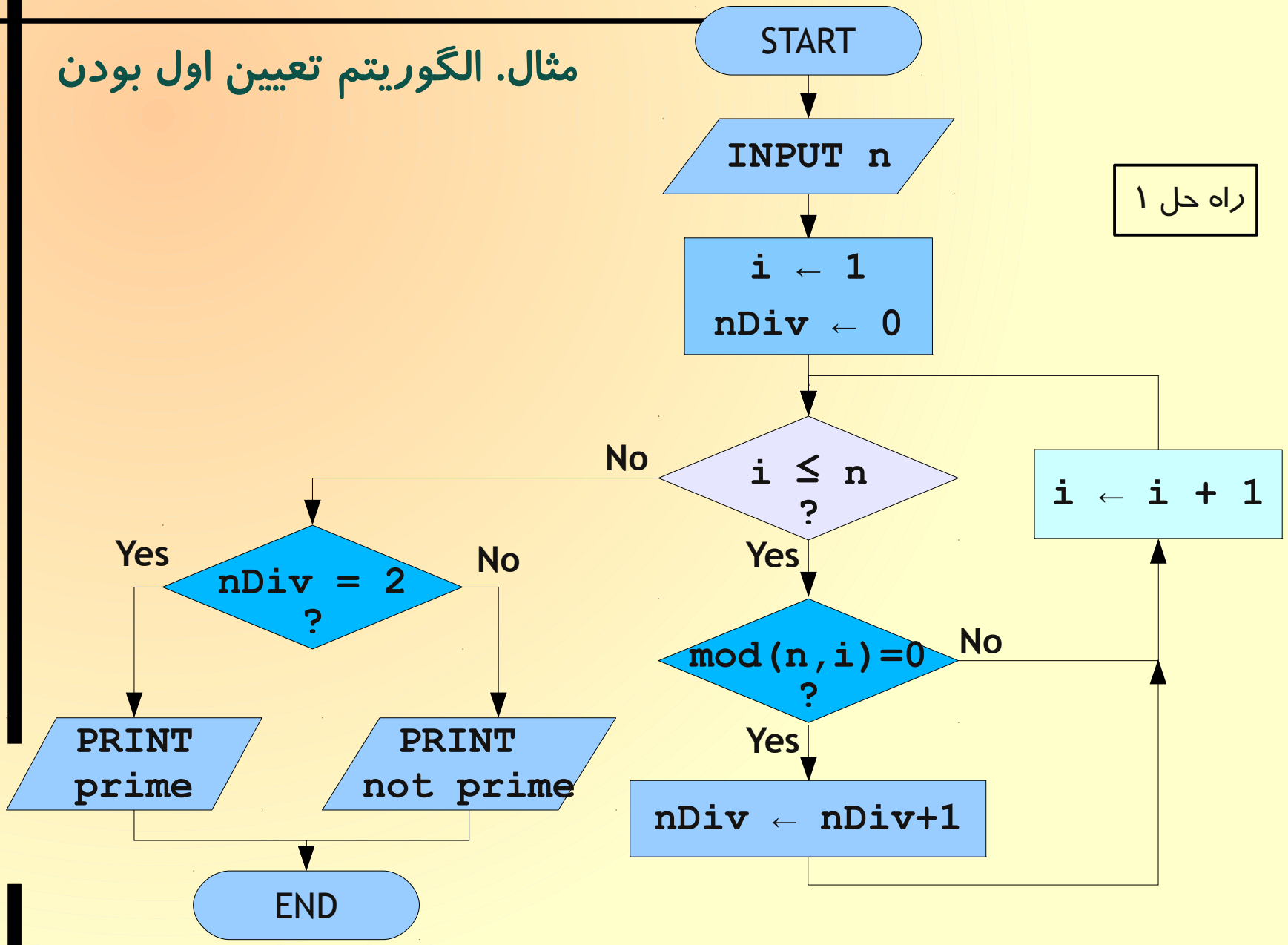
مثال. برنامه‌ی مقسوم‌علیه‌های n .

```
int main( int argc, char *argv[]){
  int n;
  cout << " n ? ";
  cin >> n;

  int i;
  for ( i=1; i<=n; i++ ) {
    if ( n%i == 0 )
     cout << i << endl;
  }
  system("pause");
  return EXIT_SUCCESS;
}
```

مسئله. برنامه‌ای بنویسید که تعیین کند عدد داده شده یک عدد اول است یا نه.

عدد $n>1$ اول است اگر تنها بر یک و n بخشپذیر باشد.

مثال. الگوریتم تعیین اول بودن



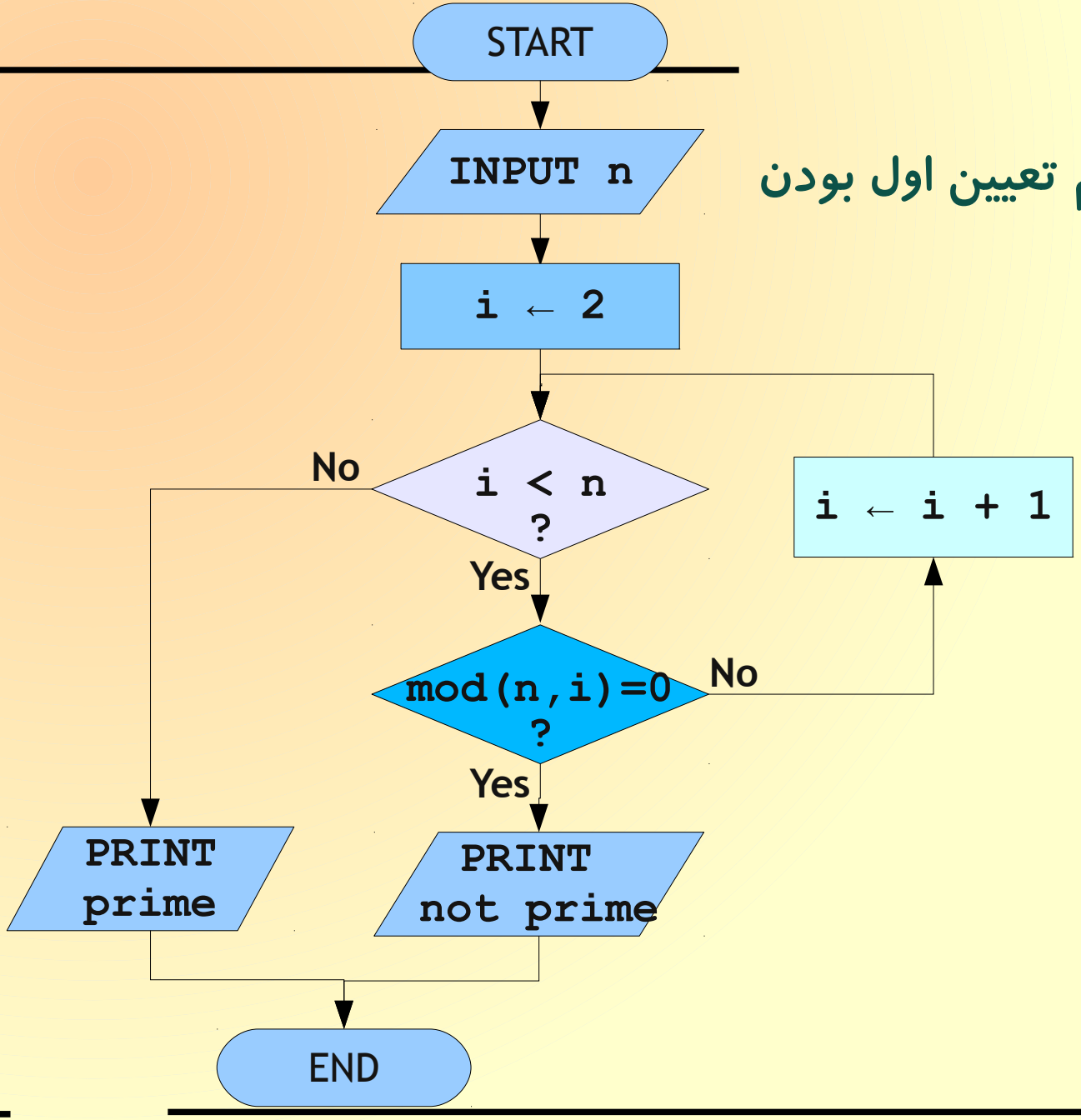
راه حل ۱

مثال. برنامه‌ی تشخیص اول بودن.

راه حل ۱

```
int main( ){
  int n;
  cout << " n ? ";
  cin >> n;

  int i;
  int nDivisor = 0;
  for ( i=1; i<=n; i++ ) {
    if ( n%i == 0 )
      nDivisor++;
  }
  if ( nDivisor == 2 )
    cout << n << " is prime." << endl;
  else
    cout << n << " is composite." << endl;
  ....
```

مثال. الگوریتم تعیین اول بودن



راه حل ۲

مثال. برنامه‌ی تشخیص اول بودن.

راه حل ۲

```
int main( ){
   int n;
   cout << " n ? ";
   cin >> n;

   int i;
   for ( i=2; i<n; i++ ) {
      if ( n%i == 0 )
         cout << n << " is composite." << endl;
   }
   cout << n << " is prime." << endl;
   ....
```

اگر n اول نباشد هر دو پیام چاپ می‌شوند!

علاوه‌بر آن، پیام «n مرکب است» ممکن است بارها چاپ شود!

مثال. برنامه‌ی تشخیص اول بودن.

راه حل ۲

```
int main( ){
  int n;
  cout << " n ? ";
  cin >> n;

  int i;
  for ( i=2; i<n; i++ ) {
    if ( n%i == 0 ) {
      cout << n << " is composite." << endl;
      break;
    }
  }

  if ( i==n )
    cout << n << " is prime." << endl;
  ....
```

دستور **break** باعث جهش روال اجرای برنامه به دستور پس از حلقه می‌شود. به عبارت دیگر، این دستور حلقه را می‌شکند.

مثال. برنامه‌ی تشخیص اول بودن.

راه حل ۳

```
int main( ){
  int n;
  cout << " n ? ";
  cin >> n;

  int i;
  for ( i=2; n%i==0 && i<n; i++ )
    ;

  if ( i < n )
    cout << n << " is composite." << endl;
  else
    cout << n << " is prime." << endl;

  ....
}
```

اگر n بر عددی مثل p بخشپذیر باشد حتماً بر q=n/p نیز بخشپذیر است.

بنابراین در دو مسئله‌ی قبل، بجای بررسی بخشپذیری n بر i=1,2,...,n کافی است تا جائی پیش رویم که

$$\frac{n}{i} \geqslant i$$

یعنی $i \leqslant \sqrt{n}$.

مثال. برنامه‌ی مقسوم‌علیه‌های n .

راه حل ۲

```
int main( int argc, char *argv[]){
  int n;
  cout << " n ? ";
  cin >> n;

  int i;
  for ( i=1; i<=sqrt(n); i++ ) {
    if ( n%i == 0 )
     cout << i << endl;
  }
  system("pause");
  return EXIT_SUCCESS;
}
```

مثال. برنامه‌ی مقسوم‌علیه‌های n .

راه حل ۲

```
int main( int argc, char *argv[]){
  int n;
  cout << " n ? ";
  cin >> n;

  int i;
  for ( i=1; i*i<=n; i++ ) {
    if ( n%i == 0 )
     cout << i << endl;
  }
  system("pause");
  return EXIT_SUCCESS;
}
```

بدون نیاز به فراخوانی تابع sqrt()

Iteration loop

حلقه‌ی تکرار

در مسئله‌ی تشخیص اول بودن، علاوه بر کاهش تعداد شرط‌های بخشپذیری به وسیله‌ی رویکرد $\sqrt{n}$ می‌توانیم 2 را استثناء کنیم و بررسی اول بودن عدد را تنها به اعداد فرد کاهش دهیم.

مثال. برنامه‌ی تشخیص اول بودن.

راه حل ۴

```
....
cin >> n;

int i=2;
if ( n%2 != 0 ) {
  for ( i=3; n%i==0 && i*i<=n; i+=2 )
    ;
}

if ( i*i <= n )
  cout << n << " is composite." << endl;
else
  cout << n << " is prime." << endl;

system("pause");
....
```