بسم الله الرحمن الرحیم

# کارگاه آموزشی گپ

فاطمه جعفری

دانشگاه اصفهان- گروه ریاضی(گروه پژوهشی کاگ)

۲۸ دی ۱۴۰۰

# مقدمات گپ

## GAP

- مخفف

*Groups, Algorithm and Programming*

- زبان برنامه‌نویسی، برای انجام محاسبات در ریاضیات در شاخه‌های مختلف جبر

# مقدمات گپ

## GAP

- مخفف

*Groups, Algorithm and Programming*

- زبان برنامه‌نویسی، برای انجام محاسبات در ریاضیات در شاخه‌های مختلف جبر

آدرس یکی از سایت‌های اینترنت که می‌توان نسخه‌های به روز شده‌ی GAP را از آن دریافت کرد:

*www.gap – system.org*

**Output**

```
 ┌─────┐   GAP 4.9.2 of 04-Jul-2018
 │ GAP │   https://www.gap-system.org
 └─────┘   Architecture: x86_64-apple-darwin16.7.0-default64
Configuration:  gmp 6.1.2, readline
Loading the library and packages ...
Packages:   AClib 1.3, Alnuth 3.1.0, AtlasRep 1.5.1, AutPGrp 1.9,
            Browse 1.8.8, CRISP 1.4.4, Cryst 4.1.17, CrystCat 1.1.8,
            CTblLib 1.2.2, FactInt 1.6.2, FGA 1.4.0, GAPDoc 1.6.1, IO 4.5.1,
            IRREDSOL 1.4, LAGUNA 3.9.0, Polenta 1.3.8, Polycyclic 2.14,
            PrimGrp 3.3.1, RadiRoot 2.8, ResClasses 4.7.1, SmallGrp 1.3,
            Sophus 1.24, SpinSym 1.5, TomLib 1.2.6, TransGrp 2.0.2,
            utils 0.54
Try '??help' for help. See also '?copyright', '?cite' and '?authors'
gap>
```

**Output**

```
 ┌─────┐   GAP 4.9.2 of 04-Jul-2018
 │ GAP │   https://www.gap-system.org
 └─────┘   Architecture: x86_64-apple-darwin16.7.0-default64
Configuration:  gmp 6.1.2, readline
Loading the library and packages ...
Packages:   AClib 1.3, Alnuth 3.1.0, AtlasRep 1.5.1, AutPGrp 1.9,
            Browse 1.8.8, CRISP 1.4.4, Cryst 4.1.17, CrystCat 1.1.8,
            CTblLib 1.2.2, FactInt 1.6.2, FGA 1.4.0, GAPDoc 1.6.1, IO 4.5.1,
            IRREDSOL 1.4, LAGUNA 3.9.0, Polenta 1.3.8, Polycyclic 2.14,
            PrimGrp 3.3.1, RadiRoot 2.8, ResClasses 4.7.1, SmallGrp 1.3,
            Sophus 1.24, SpinSym 1.5, TomLib 1.2.6, TransGrp 2.0.2,
            utils 0.54
Try '??help' for help. See also '?copyright', '?cite' and '?authors'
gap>
```

- پس از فرخوانی برنامه‌ی گپ، چند لحظه طول می‌کشد تا گپ با باز کردن کتابخانه‌ی خود، آماده‌ی انجام دستورات شود. علامت آمادگی گپ: gap>

**Output**

```
 ┌─────┐   GAP 4.9.2 of 04-Jul-2018
 │ GAP │   https://www.gap-system.org
 └─────┘   Architecture: x86_64-apple-darwin16.7.0-default64
Configuration:  gmp 6.1.2, readline
Loading the library and packages ...
Packages:   AClib 1.3, Alnuth 3.1.0, AtlasRep 1.5.1, AutPGrp 1.9,
            Browse 1.8.8, CRISP 1.4.4, Cryst 4.1.17, CrystCat 1.1.8,
            CTblLib 1.2.2, FactInt 1.6.2, FGA 1.4.0, GAPDoc 1.6.1, IO 4.5.1,
            IRREDSOL 1.4, LAGUNA 3.9.0, Polenta 1.3.8, Polycyclic 2.14,
            PrimGrp 3.3.1, RadiRoot 2.8, ResClasses 4.7.1, SmallGrp 1.3,
            Sophus 1.24, SpinSym 1.5, TomLib 1.2.6, TransGrp 2.0.2,
            utils 0.54
Try '??help' for help. See also '?copyright', '?cite' and '?authors'
gap>
```

- پس از فرخوانی برنامه‌ی گپ، چند لحظه طول می‌کشد تا گپ با باز کردن کتابخانه‌ی خود، آماده‌ی انجام دستورات شود. علامت آمادگی گپ: gap>
- در حالت عادی رنگ صفحه گپ مشکی و رنگ متن آن سفید است. ولی با راست کلیک کردن روی صفحه‌ی گپ و انتخاب گزینه‌ی Option پنجره‌ای باز می‌شود که در آن می‌توان رنگ صفحه و متن را به هر رنگ دلخواهی تغییر داد.

# مقدمات گپ

- ✓ در انتهای هر دستور گپ باید یک نقطه ویرگول ; قرار گیرد و سپس دکمه‌ی enter فشار داده شود.

# مقدمات گپ

- ✓ در انتهای هر دستور گپ باید یک نقطه ویرگول ; قرار گیرد و سپس دکمه‌ی enter فشار داده شود.
- ✓ اگر پس از دستور به جای، ; از ; ; استفاده کنیم، محاسبه توسط گپ انجام می‌شود اما نتیجه بر صفحه‌ی کامپیوتر ظاهر نمی‌شود.

# مقدمات گپ

- ✓ در انتهای هر دستور گپ باید یک نقطه ویرگول ; قرار گیرد و سپس دکمه‌ی enter فشار داده شود.
- ✓ اگر پس از دستور به جای، ; از ; ; استفاده کنیم، محاسبه توسط گپ انجام می‌شود اما نتیجه بر صفحه‌ی کامپیوتر ظاهر نمی‌شود.
- ✓ هر عبارتی پس از # در GAP خوانده نمی‌شود. به طور مثال نتیجه‌ی دستور

```
gap>#Group
```

به صورت

```
gap>
```

است (از این دستور می‌توان برای نوشتن یادداشت در برنامه استفاده کرد).

# مقدمات گپ

- ✓ در انتهای هر دستور گپ باید یک نقطه ویرگول ; قرار گیرد و سپس دکمه‌ی enter فشار داده شود.
- ✓ اگر پس از دستور به جای، ; از ; ; استفاده کنیم، محاسبه توسط گپ انجام می‌شود اما نتیجه بر صفحه‌ی کامپیوتر ظاهر نمی‌شود.
- ✓ هر عبارتی پس از # در GAP خوانده نمی‌شود. به طور مثال نتیجه‌ی دستور

```
gap>#Group
```

به صورت

```
gap>
```

است (از این دستور می‌توان برای نوشتن یادداشت در برنامه استفاده کرد).

- ✓ برای خارج شدن از برنامه، کافی است تایپ کنیم:

```
gap>quit;
```

# مقدمات گپ - ذخیره و باز خوانی یک برنامه

- ✓ برای ذخیره کردن یک برنامه در GAP کافی است در ابتدای آن برنامه از دستور InputLogTo("a");‌ استفاده کنیم.

# مقدمات گپ - ذخیره و بازخوانی یک برنامه

- ✓ برای ذخیره کردن یک برنامه در GAP کافی است در ابتدای آن برنامه از دستور InputLogTo("a");‌ استفاده کنیم.
- ✓ برای بازخوانی فایل ذخیره شده کافی است از دستور زیر استفاده کنیم: Read("a");

# فراخوانی راهنمای گپ

برای استفاده از راهنمای گپ کافی است کلمه‌ی مورد نظر را بعد از یک علامت سوال بنویسیم. به طور مثال دستور

```
gap >?matrix;
```

تمام دستورات متفاوت درونی گپ را که شامل واژه‌ی ماتریس باشند فهرست می‌کند(تصویر صفحه‌ی بعد را ببینید).

# فراخوانی راهنمای گپ

```
  Choose an entry to view, 'q' for none (or use ?<nr> later):
[1]   cvec (not loaded): Matrix
[2]   Reference: Matrix Constructions
[3]   Reference: Matrix Groups
[4]   Reference: Matrix Groups in Characteristic 0
[5]   Reference: MatrixByBlockMatrix
[6]   Reference: matrix spaces
[7]   Reference: MatrixAlgebra
[8]   Reference: MatrixOfAction
[9]   Reference: MatrixLieAlgebra
[10]  Reference: MatrixAutomorphisms
[11]  Reference: matrix automorphisms
[12]  AtlasRep: matrix MeatAxe format
[13]  Browse: Matrix Display
[14]  Congruence (not loaded): MatrixByEvenInterval
[15]  Congruence (not loaded): MatrixByOddInterval
[16]  Congruence (not loaded): MatrixByFreePairOfIntervals
[17]  cvec (not loaded): MatrixNC
[18]  FR (not loaded): MatrixQuotient
[19]  GaussForHomalg (not loaded): Matrix entry manipulation
[20]  GBNP (not loaded): MatrixQA
[21]  GradedRingForHomalg (not loaded): MatrixOfWeightsOfIndeterminates
[22]  GradedRingForHomalg (not loaded): MatrixOverGradedRing (constructor for matrices over graded rings)
[23]  GUAVA (not loaded): MatrixRepresentationOnRiemannRochSpaceP1
[24]  GUAVA (not loaded): MatrixRepresentationOfElement
[25]  GUAVA (not loaded): MatrixTransformationOnMultivariatePolynomial
[26]  HAPprime Datatypes (not loaded): Matrix echelon reduction and head elements
[27]  hecke (not loaded): MatrixDecompositionMatrix
[28]  MatricesForHomalg (not loaded): MatrixOfSymbols
[29]  polycyclic: Matrix Representations
```

# فراخوانی راهنمای گپ

```
  Choose an entry to view, 'q' for none (or use ?<nr> later):
[1]   cvec (not loaded): Matrix
[2]   Reference: Matrix Constructions
[3]   Reference: Matrix Groups
[4]   Reference: Matrix Groups in Characteristic 0
[5]   Reference: MatrixByBlockMatrix
[6]   Reference: matrix spaces
[7]   Reference: MatrixAlgebra
[8]   Reference: MatrixOfAction
[9]   Reference: MatrixLieAlgebra
[10]  Reference: MatrixAutomorphisms
[11]  Reference: matrix automorphisms
[12]  AtlasRep: matrix MeatAxe format
[13]  Browse: Matrix Display
[14]  Congruence (not loaded): MatrixByEvenInterval
[15]  Congruence (not loaded): MatrixByOddInterval
[16]  Congruence (not loaded): MatrixByFreePairOfIntervals
[17]  cvec (not loaded): MatrixNC
[18]  FR (not loaded): MatrixQuotient
[19]  GaussForHomalg (not loaded): Matrix entry manipulation
[20]  GBNP (not loaded): MatrixQA
[21]  GradedRingForHomalg (not loaded): MatrixOfWeightsOfIndeterminates
[22]  GradedRingForHomalg (not loaded): MatrixOverGradedRing (constructor for matrices over graded rings)
[23]  GUAVA (not loaded): MatrixRepresentationOnRiemannRochSpaceP1
[24]  GUAVA (not loaded): MatrixRepresentationOfElement
[25]  GUAVA (not loaded): MatrixTransformationOnMultivariatePolynomial
[26]  HAPprime Datatypes (not loaded): Matrix echelon reduction and head elements
[27]  hecke (not loaded): MatrixDecompositionMatrix
[28]  MatricesForHomalg (not loaded): MatrixOfSymbols
[29]  polycyclic: Matrix Representations
```

خروج از راهنمای گپ با کلیک کردن حرف $q$ انجام می‌شود.

اگر بخواهیم یک متغیر را در گپ تعریف کنیم باید از علامت " := " استفاده کنیم. به طور مثال اگر بخواهیم a را ماتریس $\begin{pmatrix} ۰ & ۰ & ۱ \\ ۰ & ۱ & ۰ \\ ۱ & ۰ & ۰ \end{pmatrix}$ در نظر بگیریم، کافی است از دستور زیر استفاده کنیم:

```
gap> a := [[۰, ۰, ۱], [۰, ۱, ۰], [۱, ۰, ۰]];
```

که نتیجه‌ی دستور فوق به صورت زیر است:

```
[[۱, ۰, ۰], [۰, ۱, ۰], [۰, ۰, ۱]]
```

اگر بخواهیم یک متغیر را در گپ تعریف کنیم باید از علامت " := " استفاده کنیم. به طور مثال اگر بخواهیم a را ماتریس $\begin{pmatrix} ۰ & ۰ & ۱ \\ ۰ & ۱ & ۰ \\ ۱ & ۰ & ۰ \end{pmatrix}$ در نظر بگیریم، کافی است از دستور زیر استفاده کنیم:

```
gap> a := [[۰, ۰, ۱], [۰, ۱, ۰], [۱, ۰, ۰]];
```

که نتیجه‌ی دستور فوق به صورت زیر است:

```
[[۱, ۰, ۰], [۰, ۱, ۰], [۰, ۰, ۱]]
```

با استفاده از دستور

```
gap> Display(a);
```

ماتریس a به صورت زیر نوشته می‌شود.

```
[[۰, ۰, ۱],
 [۰, ۱, ۰],
 [۱, ۰, ۰]]
```

دستور زیر دترمینان ماتریس صفحه‌ی قبل را محاسبه می‌کند:

$gap > Determinant(a);$

**نکته:** کاربرد به جای حروف بزرگ انگلیسی در توابع درونی گپ حائز اهمیت است. به طور مثال اگر دستور فوق را به صورت زیر بنویسیم

$gap > determinant(a);$

آن‌گاه خروجی آن به صورت زیر خواهد بود

*Error, Variable :*
*‘determinant‘ must have a value not in any function at line ۷ of *stdin**

# اعمال حسابی

اعمال حسابی در گپ به صورت زیر است:

۱. جمع:

```
gap>1+2;
```

۲. تفریق:

```
gap>2-1;
```

۳. ضرب:

```
gap>2*1;
```

۴. تقسیم:

```
gap>2/2;
```

۵. نما:

```
gap>2 ^ 2;
```

۶. مقایسه:

```
gap>1< 2;  2=2; 1<>2; 2>1;  2>=1; 1<=2; ⟶ true
```

# جایگشت‌ها در گپ

تمرین: همه‌ی عناصری از گروه متقارن $S_5$ را مشخص کنید که فاصله‌ی همینگ آنها از جایگشت $a := (12345)$ برابر ۳ باشد.
در ادامه همه‌ی دستورهای لازم برای حل تمرین فوق را بررسی می‌کنیم.

# جایگشت‌ها در گپ

تمرین: همه‌ی عناصری از گروه متقارن $S_5$ را مشخص کنید که فاصله‌ی همینگ آنها از جایگشت $a := (1\,2\,3\,4\,5)$ برابر ۳ باشد.
در ادامه همه‌ی دستورهای لازم برای حل تمرین فوق را بررسی می‌کنیم.

- با دستور زیر $G$ را گروه متقارن روی مجموعه‌ی ۵ عضوی قرار می‌دهیم:

```
gap >G := SymmetricGroup(5);
```

# جایگشت‌ها در گپ

تمرین: همه‌ی عناصری از گروه متقارن $S_5$ را مشخص کنید که فاصله‌ی همینگ آنها از جایگشت $a := (1\,2\,3\,4\,5)$ برابر ۳ باشد.
در ادامه همه‌ی دستورهای لازم برای حل تمرین فوق را بررسی می‌کنیم.

- با دستور زیر $G$ را گروه متقارن روی مجموعه‌ی ۵ عضوی قرار می‌دهیم:

```
gap >G := SymmetricGroup(5);
```

- با دستور زیر جایگشت $a$ را تعریف می‌کنیم:

```
gap >a := (1, 2, 3, 4, 5);
```

فاصله‌ی همینگ بین دو عنصر متمایز $a$ و$b$ از گروه $S_n$ عبارت است از تعداد نقاط انتقال یافته‌ی $ab^{-1}$ است به عبارت دیگر فاصله‌ی همینگ بین دو عنصر متمایز $a$ و$b$ عبارت است از

$$|\{i \in \{1, 2, \cdots, n\} | ab^{-1}(i) \neq i\}|.$$

فاصله‌ی همینگ بین دو عنصر متمایز $a$ و$b$ از گروه $S_n$ عبارت است از تعداد نقاط انتقال یافته‌ی $ab^{-1}$ است به عبارت دیگر فاصله‌ی همینگ بین دو عنصر متمایز $a$ و$b$ عبارت است از

$$|\{i \in \{1, 2, \cdots, n\} | ab^{-1}(i) \neq i\}|.$$

بنابراین با دستور زیر در گپ به راحتی می‌توان فاصله‌ی همینگ دو عنصر متمایز $a$ و$b$ از $S_n$ را مشخص کرد:

```
gap> NrMovedPoints(a*b^-1);
```

دستور زیر نقاط انتقال یافته‌ی $ab^{-1}$ را مشخص می‌کند:

```
gap> MovedPoints(a*b^-1);
```

برای حل تمرین مورد نظر لازم است که عناصر گروه $G$ را به صورت یک لیست مرتب داشته باشیم و برای این منظور از دستور زیر استفاده می‌کنیم:

```
AsList(G);
```

برای حل تمرین مورد نظر لازم است که عناصر گروه $G$ را به صورت یک لیست مرتب داشته باشیم و برای این منظور از دستور زیر استفاده می‌کنیم:

AsList(G);

```
gap> AsList(G);
[ (), (4,5), (3,4), (3,4,5), (3,5,4), (3,5), (2,3), (2,3)(4,5), (2,3,4), (2,3,4,5), (2,3,5,4), (2,3,5), (2,4,3),
  (2,4,5,3), (2,4), (2,4,5), (2,4)(3,5), (2,4,3,5), (2,5,4,3), (2,5,3), (2,5,4), (2,5), (2,5,3,4), (2,5)(3,4), (1,2),
  (1,2)(4,5), (1,2)(3,4), (1,2)(3,4,5), (1,2)(3,5,4), (1,2)(3,5), (1,2,3), (1,2,3)(4,5), (1,2,3,4), (1,2,3,4,5),
  (1,2,3,5,4), (1,2,3,5), (1,2,4,3), (1,2,4,5,3), (1,2,4), (1,2,4,5), (1,2,4)(3,5), (1,2,4,3,5), (1,2,5,4,3),
  (1,2,5,3), (1,2,5,4), (1,2,5), (1,2,5,3,4), (1,2,5)(3,4), (1,3,2), (1,3,2)(4,5), (1,3,4,2), (1,3,4,5,2),
  (1,3,5,4,2), (1,3,5,2), (1,3), (1,3)(4,5), (1,3,4), (1,3,4,5), (1,3,5,4), (1,3,5), (1,3)(2,4), (1,3)(2,4,5),
  (1,3,2,4), (1,3,2,4,5), (1,3,5,2,4), (1,3,5)(2,4), (1,3)(2,5,4), (1,3)(2,5), (1,3,2,5,4), (1,3,2,5), (1,3,4)(2,5),
  (1,3,4,2,5), (1,4,3,2), (1,4,5,3,2), (1,4,2), (1,4,5,2), (1,4,2)(3,5), (1,4,3,5,2), (1,4,3), (1,4,5,3), (1,4),
  (1,4,5), (1,4)(3,5), (1,4,3,5), (1,4,2,3), (1,4,5,2,3), (1,4)(2,3), (1,4,5)(2,3), (1,4)(2,3,5), (1,4,2,3,5),
  (1,4,2,5,3), (1,4,3)(2,5), (1,4)(2,5,3), (1,4,3,2,5), (1,4)(2,5), (1,4,2,5), (1,5,4,3,2), (1,5,3,2), (1,5,4,2),
  (1,5,2), (1,5,3,4,2), (1,5,2)(3,4), (1,5,4,3), (1,5,3), (1,5,4), (1,5), (1,5,3,4), (1,5)(3,4), (1,5,4,2,3),
  (1,5,2,3), (1,5,4)(2,3), (1,5)(2,3), (1,5,2,3,4), (1,5)(2,3,4), (1,5,3)(2,4), (1,5,2,4,3), (1,5,3,2,4),
  (1,5)(2,4,3), (1,5,2,4), (1,5)(2,4) ]
```

برای حل تمرین مورد نظر لازم است که عناصر گروه $G$ را به صورت یک لیست مرتب داشته باشیم و برای این منظور از دستور زیر استفاده می‌کنیم:

AsList(G);

```
gap> AsList(G);
[ (), (4,5), (3,4), (3,4,5), (3,5,4), (3,5), (2,3), (2,3)(4,5), (2,3,4), (2,3,4,5), (2,3,5,4), (2,3,5), (2,4,3),
  (2,4,5,3), (2,4), (2,4,5), (2,4)(3,5), (2,4,3,5), (2,5,4,3), (2,5,3), (2,5,4), (2,5), (2,5,3,4), (2,5)(3,4), (1,2),
  (1,2)(4,5), (1,2)(3,4), (1,2)(3,4,5), (1,2)(3,5,4), (1,2)(3,5), (1,2,3), (1,2,3)(4,5), (1,2,3,4), (1,2,3,4,5),
  (1,2,3,5,4), (1,2,3,5), (1,2,4,3), (1,2,4,5,3), (1,2,4), (1,2,4,5), (1,2,4)(3,5), (1,2,4,3,5), (1,2,5,4,3),
  (1,2,5,3), (1,2,5,4), (1,2,5), (1,2,5,3,4), (1,2,5)(3,4), (1,3,2), (1,3,2)(4,5), (1,3,4,2), (1,3,4,5,2),
  (1,3,5,4,2), (1,3,5,2), (1,3), (1,3)(4,5), (1,3,4), (1,3,4,5), (1,3,5,4), (1,3,5), (1,3)(2,4), (1,3)(2,4,5),
  (1,3,2,4), (1,3,2,4,5), (1,3,5,2,4), (1,3,5)(2,4), (1,3)(2,5,4), (1,3)(2,5), (1,3,2,5,4), (1,3,2,5), (1,3,4)(2,5),
  (1,3,4,2,5), (1,4,3,2), (1,4,5,3,2), (1,4,2), (1,4,5,2), (1,4,2)(3,5), (1,4,3,5,2), (1,4,3), (1,4,5,3), (1,4),
  (1,4,5), (1,4)(3,5), (1,4,3,5), (1,4,2,3), (1,4,5,2,3), (1,4)(2,3), (1,4,5)(2,3), (1,4)(2,3,5), (1,4,2,3,5),
  (1,4,2,5,3), (1,4,3)(2,5), (1,4)(2,5,3), (1,4,3,2,5), (1,4)(2,5), (1,4,2,5), (1,5,4,3,2), (1,5,3,2), (1,5,4,2),
  (1,5,2), (1,5,3,4,2), (1,5,2)(3,4), (1,5,4,3), (1,5,3), (1,5,4), (1,5), (1,5,3,4), (1,5)(3,4), (1,5,4,2,3),
  (1,5,2,3), (1,5,4)(2,3), (1,5)(2,3), (1,5,2,3,4), (1,5)(2,3,4), (1,5,3)(2,4), (1,5,2,4,3), (1,5,3,2,4),
  (1,5)(2,4,3), (1,5,2,4), (1,5)(2,4) ]
```

اگر بخواهیم به لیست عناصر گروه $G$ یک نام نسبت بدهیم یا بعد از دستور فوق می‌نویسیم:

$T := last;$

ویا از ابتدا می‌نویسم:

T:=AsList( G);

در این صورت $T[۱]$ عنصر اول، $T[۲]$ عنصر دوم و ... را نشان می‌دهند.

# چند نکته در مورد لیست‌ها در گپ

- لیست می‌تواند عنصر تکراری داشته باشد. مثلا دستور زیر:

```
gap> S := [۳, ۴, ۸, ۵, ۸, ۱۰];
```

خروجی زیر را دارد:

```
[۳, ۴, ۸, ۵, ۸, ۱۰]
```

# چند نکته در مورد لیست‌ها در گپ

- لیست می‌تواند عنصر تکراری داشته باشد. مثلا دستور زیر:

```
gap> S := [۳, ۴, ۸, ۵, ۸, ۱۰];
```

خروجی زیر را دارد:

```
[۳, ۴, ۸, ۵, ۸, ۱۰]
```

- ولی اگر دستور زیر را اجرا کنیم:

```
gap> Set(S);
```

آن‌گاه به خروجی زیر می‌رسیم:

```
[۳, ۴, ۵, ۸, ۱۰]
```

که با قبلی متفاوت است. در حقیقت با دستور فوق ما $S$ را به یک مجموعه در گپ تبدیل می‌کنیم. یک مجموعه در گپ فقط لیستی است که عناصر آن تکرار ندارند و در ترتیب صعودی نوشته شده‌اند. در اینجا چند نمونه آورده شده است.

gap> IsSet([5,6,8]); ⟶ true
gap> IsSet([5,6,8,7]); ⟶ false
gap> IsSet([5,6,6,8]); ⟶ false

```
gap> IsSet([5,6,8]); ⟶ true
gap> IsSet([5,6,8,7]); ⟶ false
gap> IsSet([5,6,6,8]); ⟶ false
```

- اگر بخواهیم طول یک لیست را بدانیم می‌توانیم از دستور زیر استفاده کنیم:

```
gap> Length(T);
```

```
gap> IsSet([5,6,8]); ⟶ true
gap> IsSet([5,6,8,7]); ⟶ false
gap> IsSet([5,6,6,8]); ⟶ false
```

- اگر بخواهیم طول یک لیست را بدانیم می‌توانیم از دستور زیر استفاده کنیم:

```
gap> Length(T);
```

- اگر بخواهیم اندازه‌ی یک گروه را بدانیم میتوانیم از دستور زیر استفاده کنیم:

```
gap> Size(G);
```

در ادامه یک تابع کاربردی در مورد لیست‌ها را معرفی می‌کنیم:

Filtered( list, function )

که نتیجه‌ی اجرای آن لیستی است که شامل آن عناصری از لیست list است که نتیجه‌ی تابع تک متغیره function روی آن true باشد. به طور مثال

در ادامه یک تابع کاربردی در مورد لیست‌ها را معرفی می‌کنیم:

Filtered( list, function )

که نتیجه‌ی اجرای آن لیستی است که شامل آن عناصری از لیست list است که نتیجه‌ی تابع تک متغیره function روی آن true باشد. به طور مثال

gap> Filtered( [1..11], IsPrime ); ⟶ [ 2, 3, 5, 7, 11 ]
gap> Filtered( [1..11], IsOdd ); ⟶ [ 1, 3, 5, 9,7, 11 ]

در ادامه یک تابع کاربردی در مورد لیست‌ها را معرفی می‌کنیم:

Filtered( list, function )

که نتیجه‌ی اجرای آن لیستی است که شامل آن عناصری از لیست list است که نتیجه‌ی تابع تک متغیره function روی آن true باشد. به طور مثال

gap> Filtered( [1..11], IsPrime ); ⟶ [ 2, 3, 5, 7, 11 ]
gap> Filtered( [1..11], IsOdd ); ⟶ [ 1, 3, 5, 9,7, 11 ]

گاهی اوقات، گپ تابع مورد نظر ما را ندارد ولی توابعی با خواص مناسب دارد و ما باید خود تابع مورد نیازمان را بسازیم. به عنوان مثال، تابع، *Order* که مرتبه‌ی یک عنصر از گروه را محاسبه می‌کند. اینجاست که نشانه‌گذاری پیکان گپ وارد می‌شود. تابع $i->e$ یک تابع جدید ایجاد می‌کند که $i$ را می‌گیرد و مقدار عبارت $e$ را برمی‌گرداند. مثلا

در ادامه یک تابع کاربردی در مورد لیست‌ها را معرفی می‌کنیم:

Filtered( list, function )

که نتیجه‌ی اجرای آن لیستی است که شامل آن عناصری از لیست list است که نتیجه‌ی تابع تک متغیره function روی آن true باشد. به طور مثال

gap> Filtered( [1..11], IsPrime ); ⟶ [ 2, 3, 5, 7, 11 ]
gap> Filtered( [1..11], IsOdd ); ⟶ [ 1, 3, 5, 9,7, 11 ]

گاهی اوقات، گپ تابع مورد نظر ما را ندارد ولی توابعی با خواص مناسب دارد و ما باید خود تابع مورد نیازمان را بسازیم. به عنوان مثال، تابع *Order* که مرتبه‌ی یک عنصر از گروه را محاسبه می‌کند. اینجاست که نشانه‌گذاری پیکان گپ وارد می‌شود. تابع $i->e$ یک تابع جدید ایجاد می‌کند که $i$ را می‌گیرد و مقدار عبارت $e$ را برمی‌گرداند. مثلا

gap> Filtered( T, i->Order(i)=5 );⟶ [ (1,2,3,4,5), (1,2,3,5,4), (1,2,4,5,3), (1,2,4,3,5), (1,2,5,4,3), (1,2,5,3,4), (1,3,4,5,2), (1,3,5,4,2), (1,3,2,4,5), (1,3,5,2,4), (1,3,2,5,4), (1,3,4,2,5), (1,4,5,3,2), (1,4,3,5,2), (1,4,5,2,3), (1,4,2,3,5), (1,4,2,5,3), (1,4,3,2,5), (1,5,4,3,2), (1,5,3,4,2), (1,5,4,2,3), (1,5,2,3,4), (1,5,2,4,3), (1,5,3,2,4) ]

- **تمرین:** همه‌ی عناصری از گروه متقارن $S_5$ را مشخص کنید که فاصله‌ی همینگ آنها از جایگشت (۱۲۳۴۵) برابر ۳ باشد.

- **تمرین:** همه‌ی عناصری از گروه متقارن $S_5$ را مشخص کنید که فاصله‌ی همینگ آنها از جایگشت (۱۲۳۴۵) برابر ۳ باشد.

```
gap> G:=SymmetricGroup(5);;
gap> a:=(1,2,3,4,5);;
gap> T:=AsList(G);;
gap> Filtered(T,i->NrMovedPoints(i*a^ -1)=3);
```

⟶ [ (3,4,5), (2,3)(4,5), (2,3,4), (1,2)(4,5), (1,2)(3,4), (1,2,3), (1,2,3,5,4), (1,2,4,5,3), (1,2,4,3,5), (1,2,5), (1,2,5,3,4), (1,3,4,5,2), (1,3,2,4,5), (1,3,4,2,5), (1,4,5), (1,4,5,2,3), (1,4,2,3,5), (1,5)(3,4), (1,5)(2,3), (1,5,2,3,4) ]

# چند تابع کاربردی برای لیست‌ها

- List(list,function)

  که نتیجه‌ی اجرای آن لیست جدیدی است که از اعمال تابع function روی عناصر لیست list حاصل می‌شود. به طور مثال

  gap> List([2,5,10,16],IsEven); ⟶[ true, false, true, true ]

# چند تابع کاربردی برای لیست‌ها

- List(list,function)

  که نتیجه‌ی اجرای آن لیست جدیدی است که از اعمال تابع function روی عناصر لیست list حاصل می‌شود. به طور مثال

  gap> List([2,5,10,16],IsEven); ⟶[ true, false, true, true ]

- Add(list,s)

  که نتیجه‌ی اجرای آن لیست جدیدی است که از اضافه کردن عنصر s به انتهای لیست list حاصل می‌شود. به طور مثال

  gap> A:=[2,5,10,16];
  gap> Add(A,10);
  gap> A; ⟶[ 2, 5, 10, 16, 10 ]

# چند تابع کاربردی برای لیست‌ها

- Append(list1,list2)

  که نتیجه‌ی اجرای آن لیست جدیدی است که ازاضافه کردن عناصر لیست list۲ به انتهای لیست list۱ حاصل می‌شود. به طور مثال

```
gap> Append(A,[1,2,3]);
gap> A;⟶ [ 2, 5, 10, 16, 10, 1, 2, 3 ]
```

# چند تابع کاربردی برای لیست‌ها

- Append(list1,list2)

  که نتیجه‌ی اجرای آن لیست جدیدی است که ازاضافه کردن عناصر لیست list۲ به انتهای لیست list۱ حاصل می‌شود. به طور مثال

  gap> Append(A,[1,2,3]);
  gap> A;⟶ [ 2, 5, 10, 16, 10, 1, 2, 3 ]

- Union(list1,list2), Intersection(list1,list2), Difference(list1,list2)

  که به ترتیب اجتماع، اشتراک و تفاضل دو لیست را محاسبه می‌کند. همچنین دستور Sum(list) حاصل جمع عناصر لیست list را محاسبه می‌کند.

# چند تابع کاربردی برای لیست‌ها

- Append(list1,list2)

  که نتیجه‌ی اجرای آن لیست جدیدی است که ازاضافه کردن عناصر لیست list۲ به انتهای لیست list۱ حاصل می‌شود. به طور مثال

  gap> Append(A,[1,2,3]);
  gap> A;⟶ [ 2, 5, 10, 16, 10, 1, 2, 3 ]

- Union(list1,list2), Intersection(list1,list2), Difference(list1,list2)

  که به ترتیب اجتماع، اشتراک و تفاضل دو لیست را محاسبه می‌کند. همچنین دستور Sum(list) حاصل جمع عناصر لیست list را محاسبه می‌کند.

- gap> Difference(Union(A,B),Intersection(A,B));

  دستور ساده‌ی فوق تفاضل متقارن دو مجموعه‌ی A و B را محاسبه می‌کند.

# چند تابع کاربردی برای لیست‌ها

- Append(list1,list2)

  که نتیجه‌ی اجرای آن لیست جدیدی است که ازاضافه کردن عناصر لیست list۲ به انتهای لیست list۱ حاصل می‌شود. به طور مثال

  gap> Append(A,[1,2,3]);
  gap> A;⟶ [ 2, 5, 10, 16, 10, 1, 2, 3 ]

- Union(list1,list2), Intersection(list1,list2), Difference(list1,list2)

  که به ترتیب اجتماع، اشتراک و تفاضل دو لیست را محاسبه می‌کند. همچنین دستور Sum(list) حاصل جمع عناصر لیست list را محاسبه می‌کند.

- gap> Difference(Union(A,B),Intersection(A,B));

  دستور ساده‌ی فوق تفاضل متقارن دو مجموعه‌ی A و B را محاسبه می‌کند.

- gap> Cartesian(A,B);

  که نتیجه‌ی اجرای این دستور ضرب دکارتی دو مجموعه‌ی $A$ و $B$ است.

# چند تابع کاربردی برای لیست‌ها

- ForAll(list,function)

  این دستور بررسی می‌کند که آیا نتیجه‌ی عمل تابع function بر همه‌ی عناصر list صحیح است یا خیر. به طور مثال

  gap> ForAll( [1..20], IsPrime );⟶ false
  gap> ForAll( [2,3,4,5,8,9], IsPrimePowerInt );⟶ true

# چند تابع کاربردی برای لیست‌ها

- ForAll(list,function)

  این دستور بررسی می‌کند که آیا نتیجه‌ی عمل تابع function بر همه‌ی عناصر list صحیح است یا خیر. به طور مثال

  gap> ForAll( [1..20], IsPrime );⟶ false
  gap> ForAll( [2,3,4,5,8,9], IsPrimePowerInt );⟶ true

- ForAny(list,function)

  این دستور بررسی می‌کند که آیا عنصری از لیست list وجود دارد که نتیجه‌ی عمل تابع function بر آن صحیح باشد. به طور مثال

  gap> ForAny( [1..20], IsPrime );⟶ true
  gap> ForAny( [2,3,4,5,8,9], IsPrimePowerInt );⟶ true

- تمرین: حداقل فاصله‌ی همینگ بین عناصر متمایز گروه جایگشتی تولید شده توسط دو عنصر (۱۲۳) و (۲۳۴) به دست بیاورید.

- تمرین: حداقل فاصله‌ی همینگ بین عناصر متمایز گروه جایگشتی تولید شده توسط دو عنصر (۱۲۳) و (۲۳۴) به دست بیاورید.
با توجه به اینکه به ازای هر دو عنصر $a$ و $b$ از یک گروه $ab^{-1}$ نیز یک عنصر از آن گروه است، پس کافی است اندازه‌ی نقاط انتقال یافته توسط هر کدام از عناصر غیر بدیهی (چون عناصر متمایز هستند) را به دست بیاوریم.

- تمرین: حداقل فاصله‌ی همینگ بین عناصر متمایز گروه جایگشتی تولید شده توسط دو عنصر (۱۲۳) و (۲۳۴) به دست بیاورید.
  با توجه به اینکه به ازای هر دو عنصر $a$ و $b$ از یک گروه $ab^{-1}$ نیز یک عنصر از آن گروه است، پس کافی است اندازه‌ی نقاط انتقال یافته توسط هر کدام از عناصر غیر بدیهی (چون عناصر متمایز هستند) را به دست بیاوریم.

```
gap> G:=Group((1,2,3),(2,3,4));
gap> S:=List(AsList(G),i->NrMovedPoints(i));
gap> Minimum(Difference(S,[0]));⟶ 3
```

- تمرین: حداقل فاصله‌ی همینگ بین عناصر متمایز گروه جایگشتی تولید شده توسط دو عنصر (۱۲۳) و (۲۳۴) به دست بیاورید.
  با توجه به اینکه به ازای هر دو عنصر $a$ و $b$ از یک گروه $ab^{-1}$ نیز یک عنصر از آن گروه است، پس کافی است اندازه‌ی نقاط انتقال یافته توسط هر کدام از عناصر غیر بدیهی (چون عناصر متمایز هستند) را به دست بیاوریم.

```
gap> G:=Group((1,2,3),(2,3,4));
gap> S:=List(AsList(G),i->NrMovedPoints(i));
gap> Minimum(Difference(S,[0]));⟶ 3
```

- تمرین: حداقل فاصله‌ی همینگ بین عناصر متمایز زیر مجموعه‌ی

$$\{(۴۵), (۲۳۵۴), (۲۵۴۳), (۱۲)(۳۵۴), (۱۲۳)(۴۵), (۱۲۵۴), (۱۳۲)(۴۵)\}$$

از $S_5$ به دست بیاورید.

- تمرین: حداقل فاصله‌ی همینگ بین عناصر متمایز گروه جایگشتی تولید شده توسط دو عنصر (۱۲۳) و (۲۳۴) به دست بیاورید.
با توجه به اینکه به ازای هر دو عنصر $a$ و $b$ از یک گروه $ab^{-1}$ نیز یک عنصر از آن گروه است، پس کافی است اندازه‌ی نقاط انتقال یافته توسط هر کدام از عناصر غیر بدیهی (چون عناصر متمایز هستند) را به دست بیاوریم.

```
gap> G:=Group((1,2,3),(2,3,4));
gap> S:=List(AsList(G),i->NrMovedPoints(i));
gap> Minimum(Difference(S,[0]));⟶ 3
```

- تمرین: حداقل فاصله‌ی همینگ بین عناصر متمایز زیر مجموعه‌ی

$$\{(۴۵), (۲۳۵۴), (۲۵۴۳), (۱۲)(۳۵۴), (۱۲۳)(۴۵), (۱۲۵۴), (۱۳۲)(۴۵)\}$$

از $S_5$ به دست بیاورید.

```
T:=[(4,5), (2,3,5,4), (2,5,4,3), (1,2)(3,5,4), (1,2,3)(4,5), (1,2,5,4), (1,3,2)(4,5)];
gap> T1:=Cartesian(T,T);;
gap> T2:=Filtered(T1,i->i[1]<>i[2]);;
gap> Minimum(List(T2,i->NrMovedPoints(i[1]*i[2]^-1)));⟶3
```

اگر بخواهیم فاصله کندال بین دو عنصر $x$ و $y$ را به دست بیاوریم، در صورتی‌که آخرین نسخه گپ (۴۰۱۱۰۱) را نصب کرده باشیم، کافی است ابتدا بسته‌ی "HAP' را بارگذاری کنیم برای این کار از دستور ؛("HAP")LoadPackage استفاده می‌کنیم و سپس از دستور KendallMetric(x،y) استفاده کنیم.

- تمرین: مرکز گروه $GL(۲,۸)$ (گروه خطی عام که همان ماتریسهای وارونپذیر $۲ \times ۲$ روی حلقه‌ی ۸ عضوی است) را به دست آورید.

- تمرین: مرکز گروه $GL(۲,۸)$ (گروه خطی عام که همان ماتریسهای وارونپذیر $۲ \times ۲$ روی حلقه‌ی ۸ عضوی است) را به دست آورید.
- به دست آوردن مرکز گروه $G$ در گپ با تابع Centre(G) تعریف شده است ولی ما در اینجا با استفاده از روش‌های مختلفی آن را بررسی خواهیم کرد.

- تمرین: مرکز گروه $GL(۲,۸)$ (گروه خطی عام که همان ماتریسهای وارونپذیر $۲ \times ۲$ روی حلقه‌ی ۸ عضوی است) را به دست آورید.
- به دست آوردن مرکز گروه $G$ در گپ با تابع Centre(G) تعریف شده است ولی ما در اینجا با استفاده از روش‌های مختلفی آن را بررسی خواهیم کرد.
- می‌دانیم که مرکز یک گروه برابر است با اشتراک مرکزسازهای همه‌ی عناصر آن گروه جایی که مرکز ساز عنصر $x$ از گروه $G$ عبارت است از

$$C_G(x) := \{g \in G \,|\, gx = xg\}.$$

# حلقه ی for

حلقه *for* با کلمه *for* شروع می‌شود. در خط اول دامنه تغییرات متغیر را نوشته و در آخر *do* می‌نویسیم. حلقه با *od*; خاتمه می‌یابد.

```
gap> for i in [1,2,3] do
Print(i,”-”,i^2,”-”);
od;⟶ 1-1-2-4-3-9-
```

# حلقه ی for

حلقه *for* با کلمه *for* شروع می‌شود. در خط اول دامنه تغییرات متغیر را نوشته و در آخر *do* می‌نویسیم. حلقه با *od*; خاتمه می‌یابد.

```
gap> for i in [1,2,3] do
Print(i,”-”,i^2,”-”);
od;⟶ 1-1-2-4-3-9-
```

```
gap> G:=GL(2,8);;
gap> centre:=AsList(G);;
gap> for i in G do
> H:=Filtered(T,j->j*i=i*j);
> centre:=Intersection(centre,H);
> od;
```

روش‌های دیگری برای حلقه زدن نیز وجود دارند. به عنوان مثال، ما می‌توانیم حلقه را روی زیرمجموعه‌ای مناسب از اعداد صحیح انجام داده و centre را مانند یک آرایه بپذیریم. برنامه زیر را ملاحظه کنید.

```
gap> G:=GL(2,8);;
gap> T:=AsList(G);;
gap> centre:=T;;
gap> for i in [1..Length(T)] do
> H:=Filtered(T,j->j*T[i]=T[i]*j);
> centre:=Intersection(centre,H);
> od;
```

# شکل عام حلقه‌ی while

while bool-expr do statements od;

حلقه‌ی while همه‌ی دستورات مربوط به statements را تا زمانی که عبارت بولی bool-expr درست باشد اجرا می‌کند. به مثال زیر توجه کنید:

```
gap> i := 0;; s := 0;;
gap> while s <= 200 do
> i := i + 1; s := s + i^ 2;
> od;
gap> s;⟶ 204
```

حال برنامه‌ی مربوط به مرکز یک گروه را با استفاده از حلقه‌ی while می‌نویسیم.

```
gap> G:=GL(2,8);;
gap> i:=1;;
gap> centre:=T;;
gap> while i<=Length(T) do
> centre:=Intersection(Filtered(T,j->j*T[i]=T[i]*j),centre);
> i:=i+1;
> od;
```

# شکل عام دستور شرطی if

```
if bool-expr1 then statements1
elif bool-expr2 then statements2
else statements3
fi;
```

دستور if به ما اجازه می دهد تا بسته به مقدار برخی از عبارت های بولی، دستورات را اجرا کنیم. دستور بالا به شرح زیر انجام می شود.

ابتدا عبارت bool-expr۱ ارزیابی می‌شود. اگر عبارت را درست ارزیابی کند دستورات statements۱ اجرا می شود و اجرای دستور if کامل می شود. ولی اگر عبارت bool-expr۱ غلط باشد عبارات bool-expr۲ به دنبال elif به نوبه خود ارزیابی می شوند اگر درست باشد دستورات statements۲ اجرا می شود و اجرای دستور if کامل می شود و در غیر این صورت دستور statements۳ اجرا خواهد شد.

**نکته:** ممکن است هر تعداد الیف وجود داشته باشد (ممکن است اصلا وجود نداشته باشد) و به همین ترتیب ارزیابی می‌شوند.

**نکته:** قسمت else اختیاری است و ممکن است اصلا وجود نداشته باشد.

به مثال زیر که مقدار تابع علامت را برای عدد ۱۰− محاسبه می‌کند، توجه کنید.

```
gap> i := -10;;
gap> if 0 < i then
> s := 1;
> elif i < 0 then
> s := -1;
> else
> s := 0;
> fi;
gap> s; ⟶ -1
```

حال می‌خواهیم برنامه‌ی محاسبه‌ی مرکز گروه $GL(۲,۸)$ را با کمک دستور if بنویسیم.

```
gap> G:=GL(2,8);;
gap> centre:=[];
gap> for i in G do
> if Length(Filtered(AsList(G),j->j*i=i*j))=Size(G) then Add(centre,i);
> fi;
> od;
```

در برنامه‌ی فوق هر عنصری از $G$ که با همه‌ی عناصر گروه جابه‌جا می‌شود به لیست centre اضافه می‌شود.

# شکل عام یک تابع

```
function(arguments)
statements;
end;
```

تابع زیر بزرگترین مقسوم علیه مشترک بین دو عدد را محاسبه می‌کند.

```
gap> gcd:= function(a, b)
> local c;
> while b <> 0 do
> c:= b;
> b:= a mod b;
> a:= c;
> od;
> return c;
> end;⟶ function ( a, b ) ... end
gap> gcd(30, 63);⟶ 3
```

تمرین: تابعی بنویسید که میانگین مرتبه‌ی عناصر یک گروه را محاسبه کند.

```
average:=function(G)
> local A,B;
> B:=AsList(G);
> A:=Sum(List(B,j->Order(j)))/Length(B);
> return(A);
> end;⟶function( G ) ... end
gap> average(DihedralGroup(8));⟶ 19/8
```

تمرین: تابعی بنویسید که میانگین مرتبه‌ی عناصر یک گروه را محاسبه کند.

```
average:=function(G)
> local A,B;
> B:=AsList(G);
> A:=Sum(List(B,j->Order(j)))/Length(B);
> return(A);
> end;⟶function( G ) ... end
gap> average(DihedralGroup(8));⟶ 19/8
```

**تمرین:** برنامه‌ای برای محاسبه واریانس مرتبه عناصر یک گروه تهیه کنید.

تمرین: تابعی بنویسید که میانگین مرتبه‌ی عناصر یک گروه را محاسبه کند.

```
average:=function(G)
> local A,B;
> B:=AsList(G);
> A:=Sum(List(B,j->Order(j)))/Length(B);
> return(A);
> end;⟶function( G ) ... end
gap> average(DihedralGroup(8));⟶ 19/8
```

**تمرین:** برنامه‌ای برای محاسبه واریانس مرتبه عناصر یک گروه تهیه کنید.
**تمرین:** با دستور *Filtered* پایدارساز ۳ را در گروه $A_n$ بیابید.

```
gap> G:=AlternatingGroup(5);;
gap> S:=AsList(G);
gap> F:=Filtered(S,i->3^i=3);
```

# چند تابع کاربردی در مورد گروه‌ها

$G$ را گروه متناوب $A_5$ قرار می‌دهیم. در این صورت

```
gap> IsAbelian(G);⟶ false
gap> IsSolvable(G);⟶ false
gap> IsCyclic(G);⟶ false
gap> IsNilpotent(G);⟶ false
gap> IsSimple(G);⟶ true
gap> StructureDescription(Group((1,2,3),(3,4,5)));⟶ ”A5”
```

# چند تابع کاربردی در مورد گروه‌ها

$G$ را گروه متناوب $A_5$ قرار می‌دهیم. در این صورت

```
gap> IsAbelian(G);⟶ false
gap> IsSolvable(G);⟶ false
gap> IsCyclic(G);⟶ false
gap> IsNilpotent(G);⟶ false
gap> IsSimple(G);⟶ true
gap> StructureDescription(Group((1,2,3),(3,4,5)));⟶ ”A5”
```

**تمرین:** یک زوج مرتب از $A_4$ را به تصادف انتخاب می‌کنیم. احتمال اینکه زیرگروه ساخته شده توسط این زوج مرتب آبلی، حل‌پذیر، پوچ توان، ساده و یا دوری باشد را حساب کنید.

```
gap> G:=AlternatingGroup(4);;
gap> C:=Cartesian(G,G);;
gap> Length(Filtered(C,i->IsAbelian(Group(i))))/Length(C);⟶ 1/3
```

# اطلاعاتی در مورد کتابخانه گپ

وقتی شما برنامه‌ی گپ را اجرا می‌کنید، به طور خودکار کتابخانه‌ی گپ شامل گروه‌های زیر می‌خواند:

# اطلاعاتی در مورد کتابخانه گپ

وقتی شما برنامه‌ی گپ را اجرا می‌کنید، به طور خودکار کتابخانه‌ی گپ شامل گروه‌های زیر می‌خواند:

۱ گروه‌های دوری، متناوب و متقارن که به ترتیب با دستورهای زیر فراخوانی می‌شوند:

```
CyclicGroup(n);
AlternatingGroup(n);
SymmetricGroup(n);
```

نکته: اگر بخواهیم یک گروه دوری را به صورت جایگشتی فراخوانی کنیم کافی است از دستور زیر استفاده کنیم:

```
CyclicGroup(IsPermGroup,n);
```

# اطلاعاتی در مورد کتابخانه گپ

وقتی شما برنامه‌ی گپ را اجرا می‌کنید، به طور خودکار کتابخانه‌ی گپ شامل گروه‌های زیر می‌خواند:

۱ گروه‌های دوری، متناوب و متقارن که به ترتیب با دستورهای زیر فراخوانی می‌شوند:

```
CyclicGroup(n);
AlternatingGroup(n);
SymmetricGroup(n);
```

نکته: اگر بخواهیم یک گروه دوری را به صورت جایگشتی فراخوانی کنیم کافی است از دستور زیر استفاده کنیم:

```
CyclicGroup(IsPermGroup,n);
```

۲ گروه‌های ماتریسی که شامل گروه‌های زیر است:

# اطلاعاتی در مورد کتابخانه گپ

وقتی شما برنامه‌ی گپ را اجرا می‌کنید، به طور خودکار کتابخانه‌ی گپ شامل گروه‌های زیر می‌خواند:

۱ گروه‌های دوری، متناوب و متقارن که به ترتیب با دستورهای زیر فراخوانی می‌شوند:

```
CyclicGroup(n);
AlternatingGroup(n);
SymmetricGroup(n);
```

نکته: اگر بخواهیم یک گروه دوری را به صورت جایگشتی فراخوانی کنیم کافی است از دستور زیر استفاده کنیم:

```
CyclicGroup(IsPermGroup,n);
```

۲ گروه‌های ماتریسی که شامل گروه‌های زیر است:

- گروه‌های خطی عام که گروه تشکیل شده از همه‌ی ماتریس‌های وارونپذیر $n \times n$ روی عناصر یک میدان یا حلقه است.به مثال‌های زیر توجه کنید:

```
gap>GL(3,4);⟶ GL(3,4)
gap> GeneratorsOfGroup(GL(2,Integers));
⟶ [ [ [ 0, 1 ], [ 1, 0 ] ], [ [ -1, 0 ], [ 0, 1 ] ], [ [ 1, 1 ], [ 0, 1 ] ] ]
```

# اطلاعاتی در مورد کتابخانه گپ

# اطلاعاتی در مورد کتابخانه گپ

- گروه‌های خطی خاص که گروه تشکیل شده از همه‌ی ماتریس‌های وارونپذیر $n \times n$ روی عناصر یک میدان یا حلقه با دترمینان ۱ است. به مثال‌های زیر توجه کنید:

```
gap> SL(3,4);
SL(3,4)
```

# اطلاعاتی در مورد کتابخانه گپ

- گروه‌های خطی خاص که گروه تشکیل شده از همه‌ی ماتریس‌های وارونپذیر $n \times n$ روی عناصر یک میدان یا حلقه با دترمینان ۱ است. به مثال‌های زیر توجه کنید:

```
gap> SL(3,4);
```
$\longrightarrow$ SL(3,4)

- گروه‌های ماتریسی یکانی عام و یکانی خاص هستند که با دستورهای زیر فراخوانی می‌شوند:

```
GeneralUnitaryGroup(3,5);
SpecialUnitaryGroup(3,5);
```

# اطلاعاتی در مورد کتابخانه گپ

۳ـ گروه‌های تام: گروه $G$ را یک گروه تام گوییم هرگاه $G = G'$. در حالت کلی گروه‌های تام مشخص نشده اند ولی برنامه‌ی گپ فهرست تمام این گروه‌ها را تا مرتبه‌ی ۱۰۰۰۰۰۰ دارد به جز ۸ مورد تعیین نشده‌ی زیر:

۶۱۱۴۴۰, ۱۲۲۸۸۰, ۱۷۲۰۳۲, ۲۴۵۷۶۰, ۳۴۴۰۶۴, ۴۹۱۵۲۰, ۶۸۸۱۲۸, ۹۸۳۰۴۰

البته در حالت‌های زیر هم کتابخانه‌ی گپ کامل نشده است:

۴۹۱۵۲۰, ۳۶۸۶۴۰, ۸۶۰۱۶

دستور زیر همه‌ی این اعداد را مشخص می‌کند:

# اطلاعاتی در مورد کتابخانه گپ

۳ـ گروه‌های تام: گروه $G$ را یک گروه تام گوییم هرگاه $G = G'$. در حالت کلی گروه‌های تام مشخص نشده اند ولی برنامه‌ی گپ فهرست تمام این گروه‌ها را تا مرتبه‌ی ۱۰۰۰۰۰۰ دارد به جز ۸ مورد تعیین نشده‌ی زیر:

۶۱۱۴۴۰, ۱۲۲۸۸۰, ۱۷۲۰۳۲, ۲۴۵۷۶۰, ۳۴۴۰۶۴, ۴۹۱۵۲۰, ۶۸۸۱۲۸, ۹۸۳۰۴۰

البته در حالت‌های زیر هم کتابخانه‌ی گپ کامل نشده است:

۴۹۱۵۲۰, ۳۶۸۶۴۰, ۸۶۰۱۶

دستور زیر همه‌ی این اعداد را مشخص می‌کند:

```
gap> SizesPerfectGroups();
```

# اطلاعاتی در مورد کتابخانه گپ

۳ـ گروه‌های تام: گروه $G$ را یک گروه تام گوییم هرگاه $G = G'$. در حالت کلی گروه‌های تام مشخص نشده اند ولی برنامه‌ی گپ فهرست تمام این گروه‌ها را تا مرتبه‌ی ۱۰۰۰۰۰۰ دارد به جز ۸ مورد تعیین نشده‌ی زیر:

۶۱۱۴۴۰, ۱۲۲۸۸۰, ۱۷۲۰۳۲, ۲۴۵۷۶۰, ۳۴۴۰۶۴, ۴۹۱۵۲۰, ۶۸۸۱۲۸, ۹۸۳۰۴۰

البته در حالت‌های زیر هم کتابخانه‌ی گپ کامل نشده است:

۴۹۱۵۲۰, ۳۶۸۶۴۰, ۸۶۰۱۶

دستور زیر همه‌ی این اعداد را مشخص می‌کند:

```
gap> SizesPerfectGroups();
```

دستور زیر تعداد گروه‌های تام از مرتبه‌ی $n$ را مشخص می‌کند:

```
gap> NrPerfectGroups(737280);
```

# اطلاعاتی در مورد کتابخانه گپ

۳ـ گروه‌های تام: گروه $G$ را یک گروه تام گوییم هرگاه $G = G'$. در حالت کلی گروه‌های تام مشخص نشده اند ولی برنامه‌ی گپ فهرست تمام این گروه‌ها را تا مرتبه‌ی ۱۰۰۰۰۰۰ دارد به جز ۸ مورد تعیین نشده‌ی زیر:

۶۱۱۴۴۰, ۱۲۲۸۸۰, ۱۷۲۰۳۲, ۲۴۵۷۶۰, ۳۴۴۰۶۴, ۴۹۱۵۲۰, ۶۸۸۱۲۸, ۹۸۳۰۴۰

البته در حالت‌های زیر هم کتابخانه‌ی گپ کامل نشده است:

۴۹۱۵۲۰, ۳۶۸۶۴۰, ۸۶۰۱۶

دستور زیر همه‌ی این اعداد را مشخص می‌کند:

```
gap> SizesPerfectGroups();
```

دستور زیر تعداد گروه‌های تام از مرتبه‌ی $n$ را مشخص می‌کند:

```
gap> NrPerfectGroups(737280);
```

دستور زیر هم $i$ـامین گروه کامل از مرتبه‌ی $n$ را مشخص می‌کند:

```
gap> PerfectGroup(n,i);
```

# اطلاعاتی در مورد کتابخانه گپ

۴ـ گروه‌های کوچک :(SmallGroups) که شامل گروه‌های زیر است:

# اطلاعاتی در مورد کتابخانه گپ

۴ـ گروه‌های کوچک :(SmallGroups) که شامل گروه‌های زیر است:

# اطلاعاتی در مورد کتابخانه گپ

۴ـ گروه‌های کوچک :(SmallGroups) که شامل گروه‌های زیر است:

- همه‌ی گروه‌ها تا مرتبه‌ی ۲۰۰۰ به جز ۱۰۲۴.

# اطلاعاتی در مورد کتابخانه گپ

۴ـ گروه‌های کوچک :(SmallGroups) که شامل گروه‌های زیر است:

- همه‌ی گروه‌ها تا مرتبه‌ی ۲۰۰۰ به جز ۱۰۲۴.
- گروه‌هایی تا مرتبه‌ی ۵۰۰۰ که مرتبه‌ی آن‌ها بر توان سوم هیچ عدد اولی تقسیم پذیر نیست.

# اطلاعاتی در مورد کتابخانه گپ

۴ـ گروه‌های کوچک :(SmallGroups) که شامل گروه‌های زیر است:

- همه‌ی گروه‌ها تا مرتبه‌ی ۲۰۰۰ به جز ۱۰۲۴.
- گروه‌هایی تا مرتبه‌ی ۵۰۰۰ که مرتبه‌ی آن‌ها بر توان سوم هیچ عدد اولی تقسیم پذیر نیست.
- گروه‌هایی از مرتبه‌ی $p^n$ که $p$ عددی اول و $n \leq 6$.
- گروه‌هایی از مرتبه‌ی $pq^n$ که $p$ عددی اول و $q^n$ مقسوعلیهی از $2^8$، $3^6$، $5^5$ و یا $7^4$ باشد.

- دستور زیر گروه $i$ـام از مرتبه‌ی $n$ را مشخص می‌کند:

```
gap> SmallGroup(n,i);
```

- دستور زیر گروه $i$ـام از مرتبه‌ی $n$ را مشخص می‌کند:

```
gap> SmallGroup(n,i);
```

- دستور زیر تعداد گروه‌های کوچک از مرتبه‌ی $n$ را مشخص می‌کند:

```
gap> NrSmallGroups(n);
```

- دستور زیر گروه $i$ـام از مرتبه‌ی $n$ را مشخص می‌کند:

```
gap> SmallGroup(n,i);
```

- دستور زیر تعداد گروه‌های کوچک از مرتبه‌ی $n$ را مشخص می‌کند:

```
gap> NrSmallGroups(n);
```

- دستور زیربرای به دست آوردن اطلاعات در مورد گروه‌های مرتبه‌ی $n$ بسیار مفید است.

```
gap> SmallGroupsInformation(n);
```

- دستور زیر گروه $i$ـام از مرتبه‌ی $n$ را مشخص می‌کند:

```
gap> SmallGroup(n,i);
```

- دستور زیر تعداد گروه‌های کوچک از مرتبه‌ی $n$ را مشخص می‌کند:

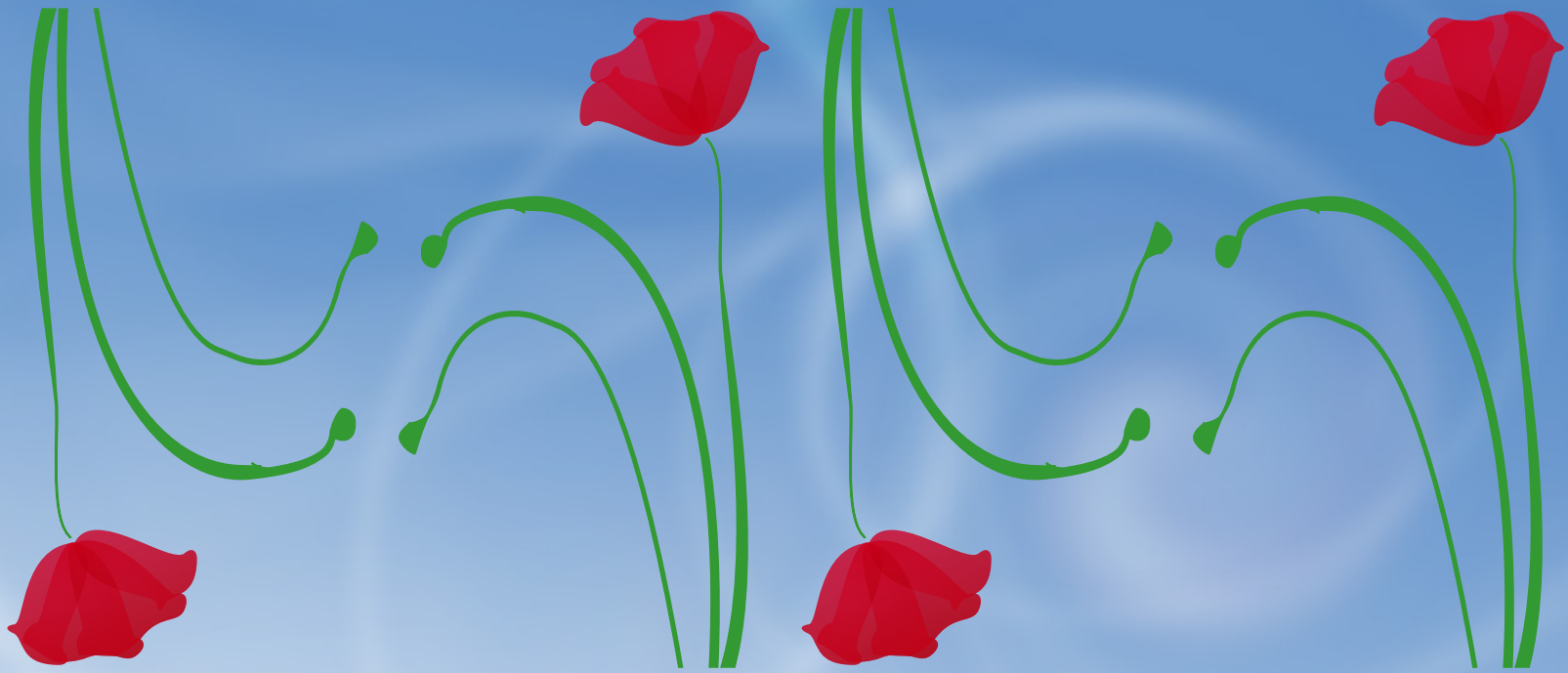
```
gap> NrSmallGroups(n);
```

- دستور زیربرای به دست آوردن اطلاعات در مورد گروه‌های مرتبه‌ی $n$ بسیار مفید است.

```
gap> SmallGroupsInformation(n);
```

- اگر بخواهیم همه‌ی گروه‌های حل‌پذیر از مرتبه‌ی ۲۰ را به دست بیاوریم کافی است از دستور زیر استفاده کنیم:

```
gap> AllSmallGroups(Size,20,IsSolvable);
<pc group of size 20 with 3 generators>,
<pc group of size 20 with 3 generators>,
<pc group of size 20 with 3 generators>,
<pc group of size 20 with 3 generators> .
```



# با تشکر از حسن توجه شما