پیش‌نیاز: set یک «درخت دودویی جستجو» است. قبل از شروع، برای آشنایی با الگوریتم‌ها و پیچیدگی زمانی توابع موجود در set، فصل 4 کتاب «طراحی الگوریتم با رویکردی خلاقانه» صفحه‌ی 92 تا 100 را «حتما» مطالعه کنید.

در این جلسه، با set (مجموعه در زبان C++) و قابلیت‌های آن، آشنا می‌شوید.

قابلیت‌هایی که set فراهم می‌کند را می‌توان، با زمان اجرایی کندتر از زمان اجرای توابع موجود در set، به راحتی پیاده‌سازی کرد؛ ولی حل برخی مسائلی که باید زمان اجرای کوتاهی داشته باشند، ملزوم به استفاده از set می‌باشد.

*****

- **برای استفاده از set و قابلیت‌های تعریف شده برای آن، باید تابع کتابخانه‌ای set فراخوانده شود:**

```
#include<set>
```

- **یک مجموعه که اعضای آن از نوع (متغیر) type است؛ این‌گونه تعریف می‌شود:**

```
set<type> name;
```

type می‌تواند int، pair، string، double، char و ... باشد.

به جای name می‌توان، تمام رشته‌هایی که در قانون نامگذاری متغیرهای C++ صدق می‌کند، قرار داد.

- **برای اضافه کردن یک عنصر (به نام key) به مجموعه‌ای به نام s:**

```
s.insert(key);
```

نوع متغیر key باید با نوع مجموعه‌ی s (نوع متغیرهای اعضای s) یکسان باشد. set نمی‌تواند دو عنصر یکسان داشته باشد. به همین خاطر، اگر چند عنصر با مقدارهای مساوی را در یک set، insert کنید؛ فقط یک عنصر با آن مقدار، در مجموعه قرار می‌گیرد. (نوع دیگری از مجموعه در STL موجود است که multiset نام دارد و قابلیت نگه‌داشتن چندین عنصر با مقدار یکسان را دارد. برای آشنایی با multiset به سایت C++ مراجعه کنید.)

- **برای این‌که چک کنیم عنصر a در مجموعه ی s هست یا نه؛ از find استفاده می‌کنیم. find، در صورتی که عنصر a در مجموعه s قرار نداشت، برابر با s.end() خواهد بود. به عنوان مثال، شرط زیر زمانی true خوهد بود که عنصر a در مجموعه‌ی s موجود باشد:**

```
if(s.find(key)!=s.end())
```

- برای حذف کردن عنصری با مقدار a از مجموعه s، از erase استفاده می‌کنیم.

```
s.erase(a);
```

اگر عنصر a در مجموعه ی S موجود نباشد هم، مشکلی ایجاد نمی‌شود.

- یکی از قابلیت‌های پر استفاده‌ای که C++ برای set فراهم کرده است، begin است. با begin می‌توان به کوچکترین عنصر موجود در s دسترسی داشت. (کوچکترین عنصر مجموعه‌هایی از نوع pair یا string، با قواعد مقایسه‌ی دو عنصر از این نوع، که در جلسات قبلی گفته شده، مشخص می‌شود.) s.begin() یک اشاره‌گر به کوچکترین عنصر موجود در مجموعه ی s است. پس *s.begin() برابر با مقدار کوچکترین عنصر موجود در مجموعه ی s خواهد بود. (یادآوری: با قرار دادن * قبل از اشاره‌گرها، مقدار موجود در خانه ای که اشاره‌گر به آن اشاره می‌کند، حاصل می‌شود.)

با این تعاریف، تکه‌کد زیر، عدد 3 را در خروجی چاپ می‌کند:

```
set<int> s;
s.insert(5);
s.insert(3);
s.insert(10);
cout<<*s.begin()<<endl;
```

- مقدار s.empty() در صورتی که مجموعه ی s خالی باشد، true و در غیر این صورت، false خواهد بود. بنابراین، برای چک کردن خالی نبودن مجموعه‌ی s می‌توان نوشت:

```
if(!s.empty())
```

- برای خالی کردن مجموعه ی s، از clear استفاده می‌کنیم:

```
s.clear();
```

تمرین: تابعی بنویسید که همه‌ی اعضای مجموعه‌ی داده شده‌ی S را در خروجی چاپ کند.