



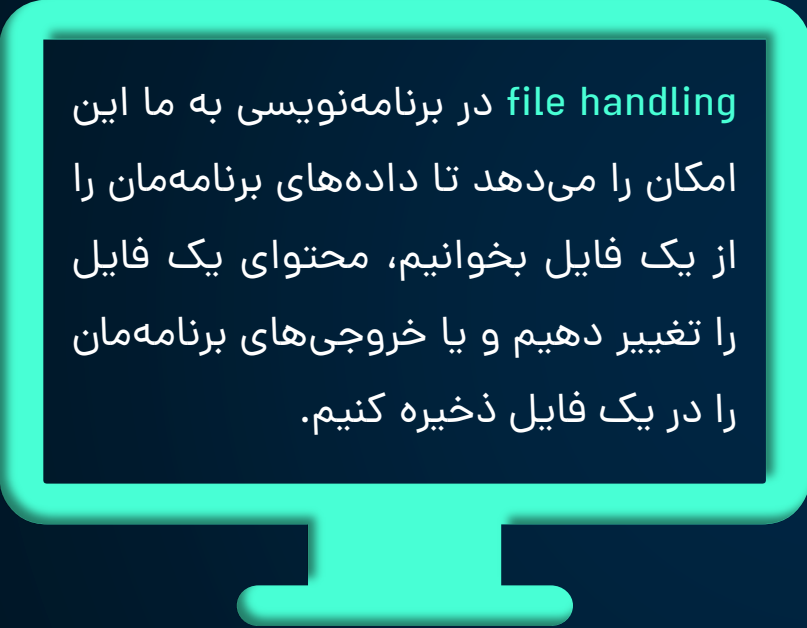
جلسه سیزدهم

کار با فایل

بسم الله الرحمن الرحيم



کارگاه مبانی برنامه‌نویسی - دانشکده مهندسی کامپیوتر دانشگاه امیرکبیر

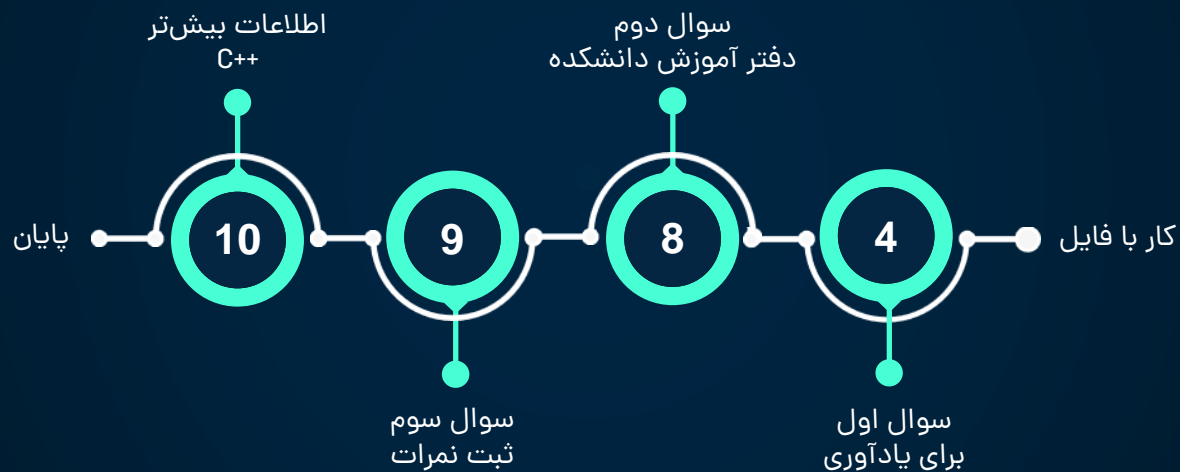


`file handling` در برنامه‌نویسی به ما این امکان را می‌دهد تا داده‌های برنامه‌مان را از یک فایل بخوانیم، محتوای یک فایل را تغییر دهیم و یا خروجی‌های برنامه‌مان را در یک فایل ذخیره کنیم.

گاهی در برنامه‌نویسی پیش می‌آید که نسبت به ذخیره کردن خروجی‌های برنامه احساس نیاز می‌کنیم. یا ترجیح می‌دهیم به جای هر بار وارد کردن ورودی‌های برنامه، یک جا لیست ورودی‌ها را داشته باشیم و فقط برنامه را اجرا کنیم. برنامه‌نویسان برای رفع این نیازمندی‌ها از فایل‌ها استفاده می‌کنند.

برای آشنایی و یاد گرفتن این توانایی این جلسه از کارگاه به مبحث کار با فایل‌ها اختصاص داده شده است.

فهرست



سوال اول: برای یادآوری

سلام سلام به همگی ما دوباره اومدیم. این بار دیگه از همون اول دستورکار رو به تسخیر خودمون درآوردیم (((=



تا الان هرکاری می‌کردیم و هر برنامه‌ای می‌نوشتیم نتیجه‌ش رو همون لحظه می‌دیدیم و بعد از اجرای دوباره برنامه، نتیجه‌ی قبل پاک می‌شد. اما این روند برای ادامه‌ی دنیای برنامه‌نویسی شاید کافی نباشه و گاهی لازمه جایی باشه که نتیجه‌ی برنامه رو اونجا ذخیره کنیم. ما می‌خوایم این جلسه با کمک **File** ها این کار رو انجام بدیم.



برای کار با فایل باید ابتدا یک اشاره‌گر از نوع فایل بسازیم و با دستور **fopen** فایل رو باز کنیم تا بتونیم به محتوایش دسترسی داشته باشیم.

```
FILE *testfile = fopen("test.txt", "rb");
```



تابع `fopen` طبق مثال صفحه‌ی قبل دو ورودی دریافت می‌کند که یکی از اون‌ها اسم فایل و از نوع `char *` است و ورودی دوم نوع رفتار و استفاده از فایل رو مشخص می‌کند.

"r"	خواندن از فایل متنی
"w"	نوشتن در فایل متنی
"a"	اضافه کردن به انتهای فایل متنی
"rb"	خواندن از فایل به صورت باینری
"wb"	نوشتن در فایل به صورت باینری
"ab"	اضافه کردن به انتهای فایل به صورت باینری



یه سری از ابتدایی‌ترین توابعی که باید برای کار با فایل ازشون استفاده کنیم `fwrite` و `fprintf` برای نوشتن در فایل و `fscanf` و `fread` برای خواندن محتوای فایل هستن.



تابع‌های `fscanf` و `fprintf` حالت خاصی از توابعی هستند که قبلاً از شون استفاده می‌کردیم که در کار با فایل، ورودی اول اشاره‌گری از نوع فایل هست.



اما تابع‌های `fread` و `fwrite` رو باید دقیق‌تر بررسی کنیم. این نکته رو یامون باشه که این توابع یک قطعه یا block از اطلاعات رو از یک فایل می‌خوانن یا توی فایل می‌نویسن. ورودی اول این توابع اشاره‌گر به ابتدای یک آرایه، ورودی دوم اندازه هر قسمت از block، ورودی سوم طول block و ورودی چهارم اشاره‌گری از نوع File هست.

مثلاً کد زیر رو ببینین:

```
FILE *testfile = fopen("test.txt", "wb");  
char *str = "Hello World";  
fwrite(str, sizeof(char), 11, testfile);  
fclose(testfile);
```



حالا با توجه به دانشی که از فایل‌ها دارین رفتار قطعه کد قبلی رو قبل از اجرا پیش‌بینی کنین.



توجه توجه! به خط آخر کد یعنی `fclose` کردن هر فایلی که باز کردید توجه ویژه‌ای داشته باشین که جلوتر رعایتش کنین.

دوباره توجه توجه! برای ورودی اول `fopen` حواستون باشه که فایل رو درست آدرس‌دهی کنین.



حالا چند تا سوال می‌خوام بپرسم... اشاره‌گری داشتیم که به ابتدای فایل اشاره می‌کرد، بعد از اجرای کد در چه موقعیتی از فایل قرار می‌گیره؟ اگر فایل `test.txt` در پوشه‌ای در کنار کد موجود نباشه خروجی برنامه چی میشه؟ چرا؟

سوال دوم: دفتر آموزش دانشکده



حالا رسیدیم دوباره به همون کدی که جلسه‌ی قبل زده بودیم. یادتونه آخرین بخش سوپر کلاس بود؟ الان می‌خوایم به جای اینکه برای هر بار اجرا کردن برنامه‌مون مجبور باشیم اطلاعات تمام دانش‌جوهای کلاس رو دونه دونه وارد کنیم، این اطلاعات رو از آموزش دانشکده بگیریم.



فرض کنیم این اطلاعات داخل یه فایل به فرمت csv که شامل اطلاعات ۱۰۰ دانش‌جو هست در اختیارتون گذاشتیم. برنامه‌ی قبلی‌تون رو به این شکل ارتقا بدین که اطلاعات هر دانش‌جو شامل نام و نام‌خانوادگی و شماره دانشجویی اون‌ها رو از فایل بخونه و توی همون لینکدلیست جلسه‌ی قبل ذخیره کنه.



سوال سوم: ثبت نمرات

شما الان یاد گرفتین که چطور می‌شه از توی یه فایل اطلاعاتی رو خوند و باهاشون مثل ورودی‌های عادی برنامه کار کرد.



حالا می‌خوایم اطلاعات خروجی‌مون رو که همیشه روی کنسول می‌داشتیم هم با کمک فایل‌ها ذخیره و مدیریت کنیم. برای این کار قراره همون برنامه‌ی قبلی که اجرا شد، اسم و مشخصات دانش‌جوها به همراه نمرات هر فرد توی این فایل ذخیره بشه. در آخر هم میانگین کل کلاس توی فایل نوشته بشه.



پس خیلی قرار نیست کار سختی رو انجام بدیم. کد برنامه رو که قبلا زدین فقط قراره یه کاری کنین که خروجی‌ها این بار توی یه فایل ذخیره بشن.

اطلاعات بیش‌تر: C++

خب، امیدواریم که توی این ترم با زبان C به خوبی آشنا شده باشین و به مفاهیم مورد نیاز برای کار باهاش هم مسلط شده باشین. یک مساله‌ای که نیاز به دونین و شاید تا الان خودتون هم متوجه شدین، اینه که خیلی وقت‌ها زبان‌های برنامه‌نویسی مختلفی بر اساس نیازمندی‌های مختلف ساخته می‌شن. به عنوان مثال، زبان JavaScript برای پویاسازی صفحات اینترنتی، زبان Kotlin برای برنامه‌نویسی دستگاه‌های اندروید و ... ساخته شدن.



اگه یادتون باشه، جلسه‌های قبل در مورد پارادایم Object Oriented به کم باهاتون صحبت کردیم. در سال ۱۹۸۲، فردی به اسم **Bjarne Stroustrup** با فلسفه‌ی وارد کردن پایه‌ی اصلی برنامه‌نویسی Object Oriented، یعنی Class ها به زبان C، زبان جدیدی به نام C++ رو پایه‌گذاری کرد که هنوز هم به عنوان یکی از اصلی‌ترین زبان‌ها در زمینه‌ی OOP استفاده می‌شه. شاید براتون جالب باشه بدونین که اپراتور ++ که توی زبان C باهاش آشنا شدین، اول توی زبان C++ پیاده‌سازی شده و بعدا به استانداردهای زبان C اضافه شده.





در حین توسعه‌ی این زبان، برنامه‌نویس‌هاش خیلی از مسائلی که فکر می‌کردن پیاده‌سازیش در زبان C مشکل‌ساز هست رو تغییر دادن، در حالی که بعضی کتابخانه‌های زبان C رو هم درونش نگه داشته و بعضا هم می‌شه قسمت‌هایی از زبان C رو با کامپایلرهای C++ کامپایل کرد که بشه از کدهای ترکیبی بین این دو زبان استفاده کرد. ورژن‌های مختلفی از C++ وجود دارن که جدیدترینش ورژن C++17 هست. از این زبان دو استاندارد وجود داره که یکی‌ش مال گروه نرم‌افزاری GNU و اون یکی‌ش برای شرکت مایکروسافت هست.

```
#include <iostream>
#include <string>
using namespace std;
```

حالا وقتشه بریم یه کم با کد C++ آشنا بشیم:



```
int main() {
    int num1 = 70;
    double num2 = 256.783;
    char ch = 'A';
    int num3;
    string s;
    cin >> num3 >> s;
    cout << num1 << " " << num2 << " " << num1 + num3 << endl << s << endl <<
    "character : " << ch << endl;
    return 0;
}
```



اولین تفاوتی که به چشم میاد استفاده از کتابخونه‌ی `iostream` مخفف (`input/output` stream) به جای `stdio` هست، این کتابخونه حاوی توابع `cin` و `cout` هست که با استفاده از عملگرهای `<<` و `>>` کار می‌کنن.

نکته‌ی مهم این دو تابع `generic` بودنشون هست، به این معنا که انواع تایپ‌های مختلف رو می‌شه به عنوان ورودی یا خروجی به این توابع به بافر سیستم پاس داد.



همون‌طور که می‌بینین `C++` با کتابخونه‌ی `string` هم تایپ `string` رو تعریف کرده که همین مسئله براش صادق‌ه. (تعریف عملگر ورودی یا خروجی برای یه تایپ خاص با خاصیتی از `C++` به اسم `Operator Overloading` انجام می‌شه که توضیحش رو می‌تونید خودتون سرچ کنید و درباره‌ش بیشتر بخونین).



و در آخر، مفهوم namespace در زبان C++، برای از بین بردن name-conflict ها استفاده شده. به این معنی که وقتی حجم کتابخانه‌ها مون خیلی زیاد بشن، توابع مختلفی برای کارهای مختلف ممکنه داشته باشیم که اسم یکسان دارن، برای این‌که بخوایم نشون بدیم دقیقا کدوم کاربرد مد نظر مونه، این توابع رو درون namespace های مختلفی قرار می‌دیم و از هر کدوم با ارجاع دادن به خودش استفاده می‌کنیم. (فرض کنید به عنوان یه جداساز توی کتابخانه‌ها استفاده می‌شه) این‌جا وقتی از `using namespace std` استفاده کردیم یعنی داریم به برنامه می‌گیم کلا توابع موجود توی کتابخانه‌ی `std` رو جز `scope` اصلی برنامه حساب کن، جوری که انگار همون بالای برنامه تعریف شدن. راه دقیق‌تر (ولی طولانی‌تر) استفاده از کتابخانه‌ها اینه که با اپراتور :: بهشون ارجاع بدیم.

به عنوان مثال، اگه بخوایم این‌طوری با `cin` کار کنیم، داریم:

```
#include <iostream>
#include <string>
```

```
int main() {
    int num1 = 70;
    std::cin >> num1;
    std::cout << num1 << std::endl;
    return 0;
}
```

همونطور که می‌بینید، عبارت endl هم



یک کاراکتر تعریف شده در namespace

std هست که به معنای کاراکتر \n است.

اینجا ما وارد جزئیات OOP و پیاده سازی در C++ نمی‌شیم، چون که قراره مفهومش رو در درس

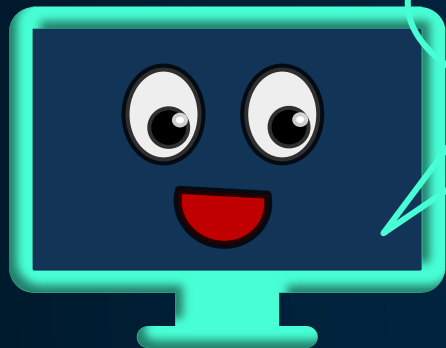


برنامه‌نویسی پیش‌رفته یاد بگیرین و خود زبان C++ رو هم در صورت نیاز می‌تونین از منابع

اینترنتی یاد بگیرین.

با آرزوی موفقیت و سلامتی برای همه و در آخر هم خدانگه‌دار ☺





این سری اومدیم
تنها نباشی با هم
خداحافظی کنیم از
بچه‌ها

