برنامه‌سازی کامپیوتر

جلسه یازدهم - بخش اول

فایل در زبان C

طرح کلی

- فایل‌ها
  - فایل
  - روش‌های ذخیره داده‌ها در فایل
  - انواع فایل و تفاوت‌های آن‌ها
    - متنی
    - باینری
  - سازمان فایل‌ها
    - ترتیبی
    - تصادفی
  - اشاره‌گر فایل
  - بازکردن فایل
  - بستن فایل

مقدمه

- داده‌هایی که در برنامه‌های C از آن‌ها استفاده می‌کنیم، در متغییرهای معمولی، آرایه‌ها و ساختمان‌ها ذخیره شده و مورد پردازش قرار می‌گیرند.
- متغییرها، آرایه‌ها و ساختمان‌ها روی حافظه RAM قرار می‌گیرند، لذا پس از قطع جریان برق (و یا خاموش شدن کامپیوتر) و یا خروج از برنامه، داده‌هایی که در آن‌ها ذخیره شده‌اند از بین می‌روند و برای استفاده مجدد از اطلاعات باید آن‌ها را دوباره وارد کرد. (مانند نمرات دانشجویان)
- برای رفع این مشکل و ذخیره دائمی اطلاعات برنامه‌ها، نوعی ساختمان داده به نام **فایل** مورد استفاده قرار می گیرد.
- فایل بر روی حافظه جانبی مثل دیسک، نوار و . . . تشکیل می‌شود. چون اطلاعات موجود در روی حافظه جانبی با قطع جریان برق، قطع اجرای برنامه و یا دلایلی از این قبیل از بین نمی‌روند، این اطلاعات بصورت دائمی ذخیره شده و به دفعات زیادی قابل استفاده هستند.

فایل

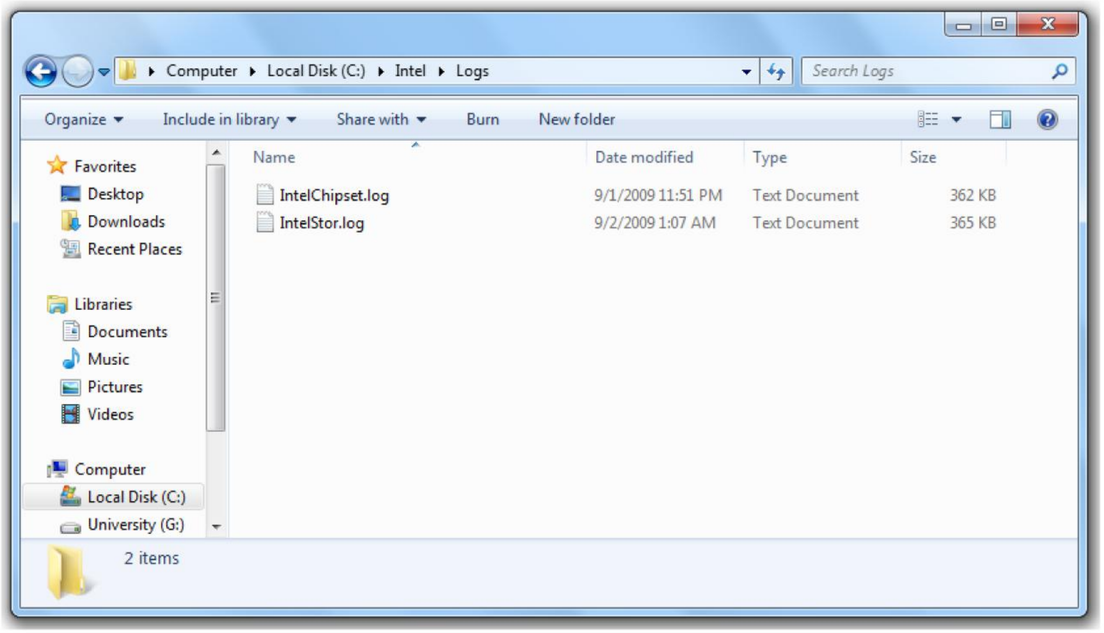
- هر فایل شامل مجموعه ای از داده‌های مرتبط به هم است. مانند داده‌های مربوط به کلیه دانشجویان یک دانشگاه.
- داده‌های مربوط به هر یک از اجزای فایل، یک **رکورد** نام دارد. به عنوان مثال، در یک دانشگاه داده‌های مربوط به هر دانشجو تشکیل یک رکورد را می‌دهند (). لذا می‌توان گفت که

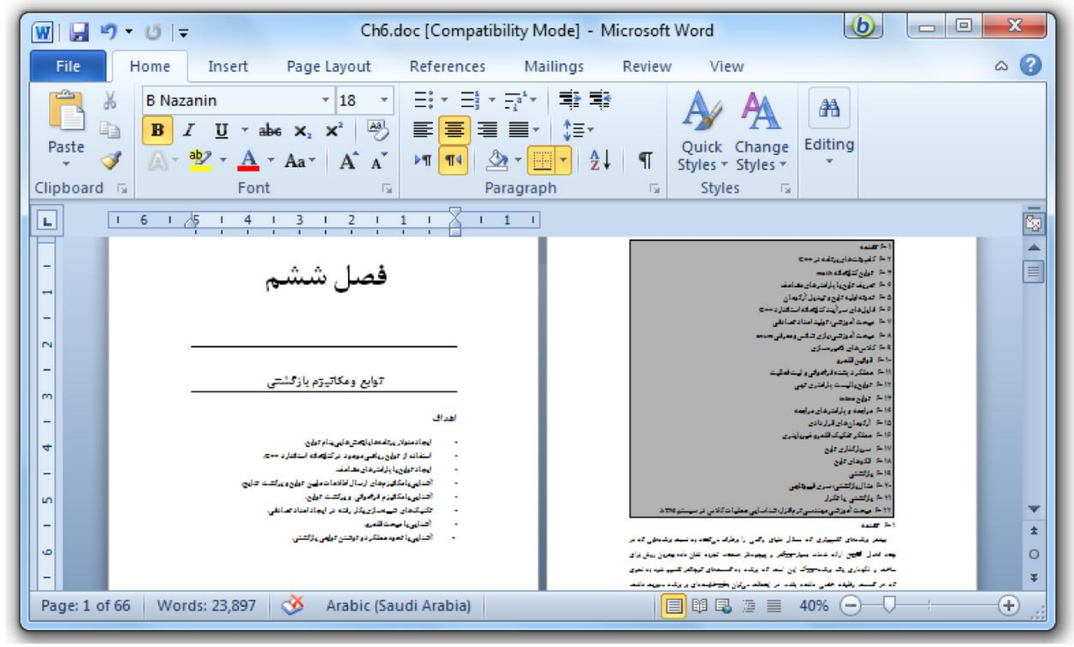
هر فایل، مجموعه‌ای از چند رکورد است.

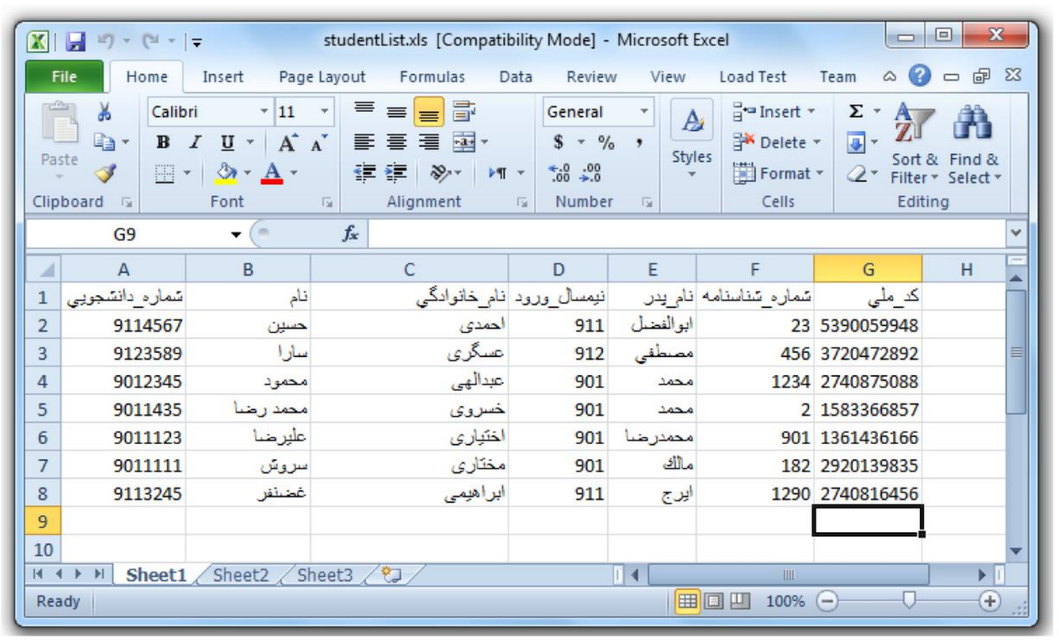
- اگر بازهم دقیق‌تر به فایل دانشجویان دانشگاه پرداخته شود، مشاهده می‌گردد که هر دانشجو ممکن است چند قلم داده داشته باشد: مثل نام دانشجو، تعداد واحدهایی که گذرانده، نمره هر درس و ... . به هریک از اجزای یک رکورد، **فیلد** گفته می‌شود. لذا می‌توان گفت که

هر رکورد مجموعه‌ای از چند فیلد است.

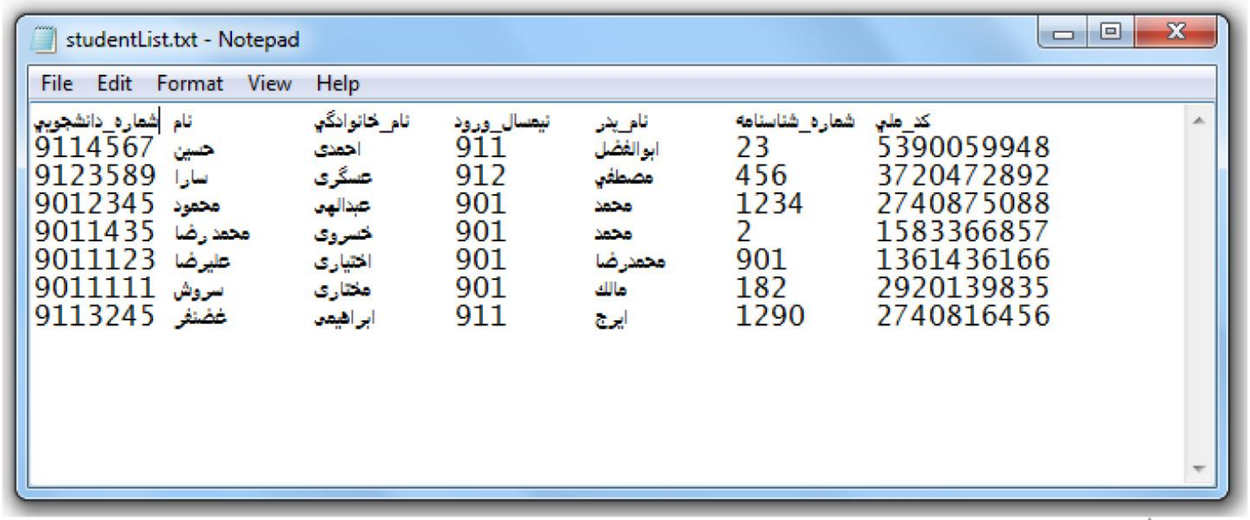
مثال







شماره_دانشجويي	نام	نام_خانوادگي	نيمسال_ورود	نام_پدر	شماره_شناسنامه	كد_ملي
9114567	حسين	احمدی	911	ابوالفضل	23	5390059948
9123589	سارا	عسگری	912	مصطفي	456	3720472892
9012345	محمود	عبدالهی	901	محمد	1234	2740875088
9011435	محمد رضا	خسروی	901	محمد	2	1583366857
9011123	عليرضا	اختياری	901	محمدرضا	901	1361436166
9011111	سروش	مختاری	901	مالك	182	2920139835
9113245	غضنفر	ابراهيمي	911	ايرج	1290	2740816456



روش‌های ذخیره داده‌ها در فایل

داده‌ها ممکن است به ۴ روش در فایل ذخیره شده سپس بازیابی شوند:

۱) داده‌ها، کاراکتر به کاراکتر در فایل نوشته شده سپس کاراکتر به کاراکتر از فایل خوانده شوند.

۲) داده‌ها بصورت رشته‌ای از کاراکترها در فایل نوشته شده به صورت رشته‌ای از کاراکترها دستیابی شوند.

۳) داده‌ها در حین نوشتن برروی فایل، با فرمت خاصی نوشته شده سپس با همان فرمت خوانده شوند (عددی، کاراکتری، رشته‌ای).

۴) داده‌ها به شکل ساختمان (رکورد) بر روی فایل نوشته شده سپس به صورت ساختمان از فایل خوانده شوند.

انواع فایل از نظر نوع اطلاعات

- داده‌ها ممکن است در فایل به دو صورت ذخیره شوند:
  - اسکی یا متن
  - باینری
- این دو روش در موارد زیر با یکدیگر تفاوت دارند:
  - نحوه ذخیره شدن اعداد بر روی دیسک
  - تعیین انتهای خط
  - تعیین انتهای فایل

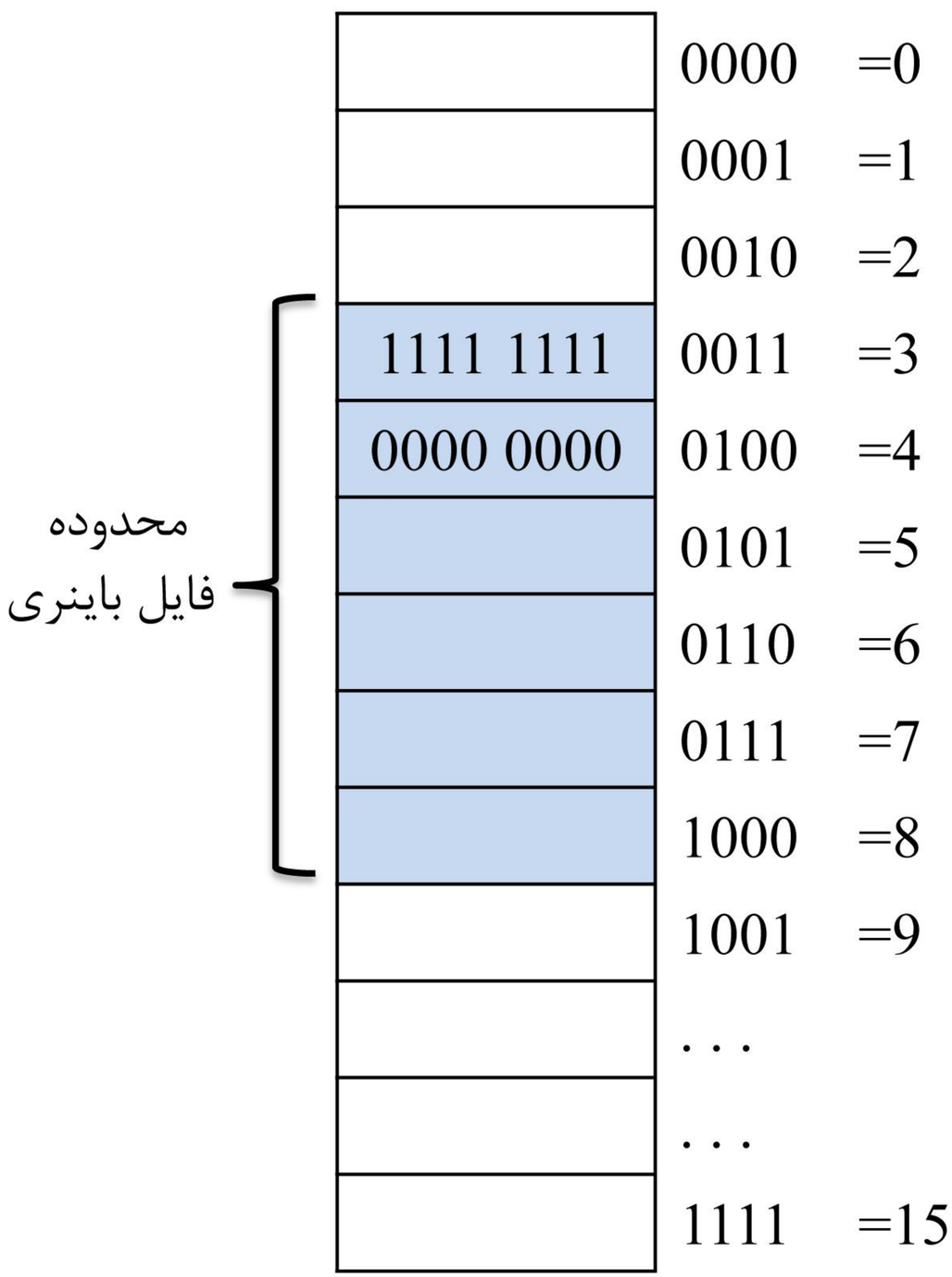
تفاوت متنی و باینری (۱)

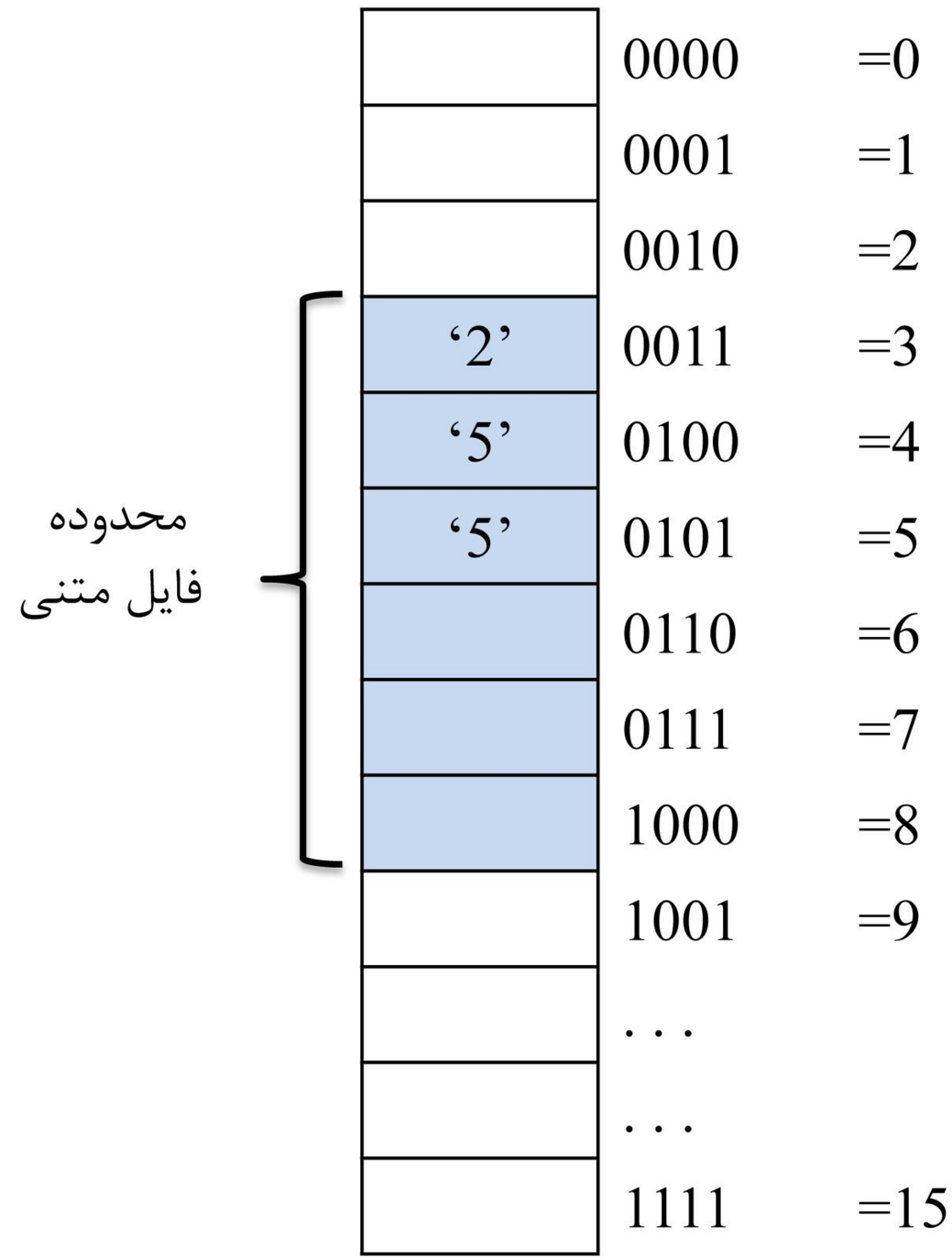
در فایل متنی، اعداد به صورت رشته‌ای از کاراکترها ذخیره می‌شوند ولی در فایل باینری اعداد به همان صورتی که در حافظه قرار می‌گیرند بر روی دیسک ذخیره می‌شوند.

مثال:

در فایل متنی، عدد ۲۵۵ سه بایت را اشغال می‌کند. زیرا هر رقم آن، به صورت یک کاراکتر درنظر گرفته می‌شود. ولی در فایل باینری این عدد در ۲ بایت ذخیره می‌گردد (چون عدد ۲۵۵ یک عدد صحیح است و اعداد صحیح در دو بایت ذخیره می‌شوند).

مثال


محدوده
فایل باینری
1111 1111
0000 0000
0000 =0
0001 =1
0010 =2
0011 =3
0100 =4
0101 =5
0110 =6
0111 =7
1000 =8
1001 =9
. . .
. . .
1111 =15


محدوده
فایل متنی
‘2’
‘5’
‘5’
0000 =0
0001 =1
0010 =2
0011 =3
0100 =4
0101 =5
0110 =6
0111 =7
1000 =8
1001 =9
. . .
. . .
1111 =15

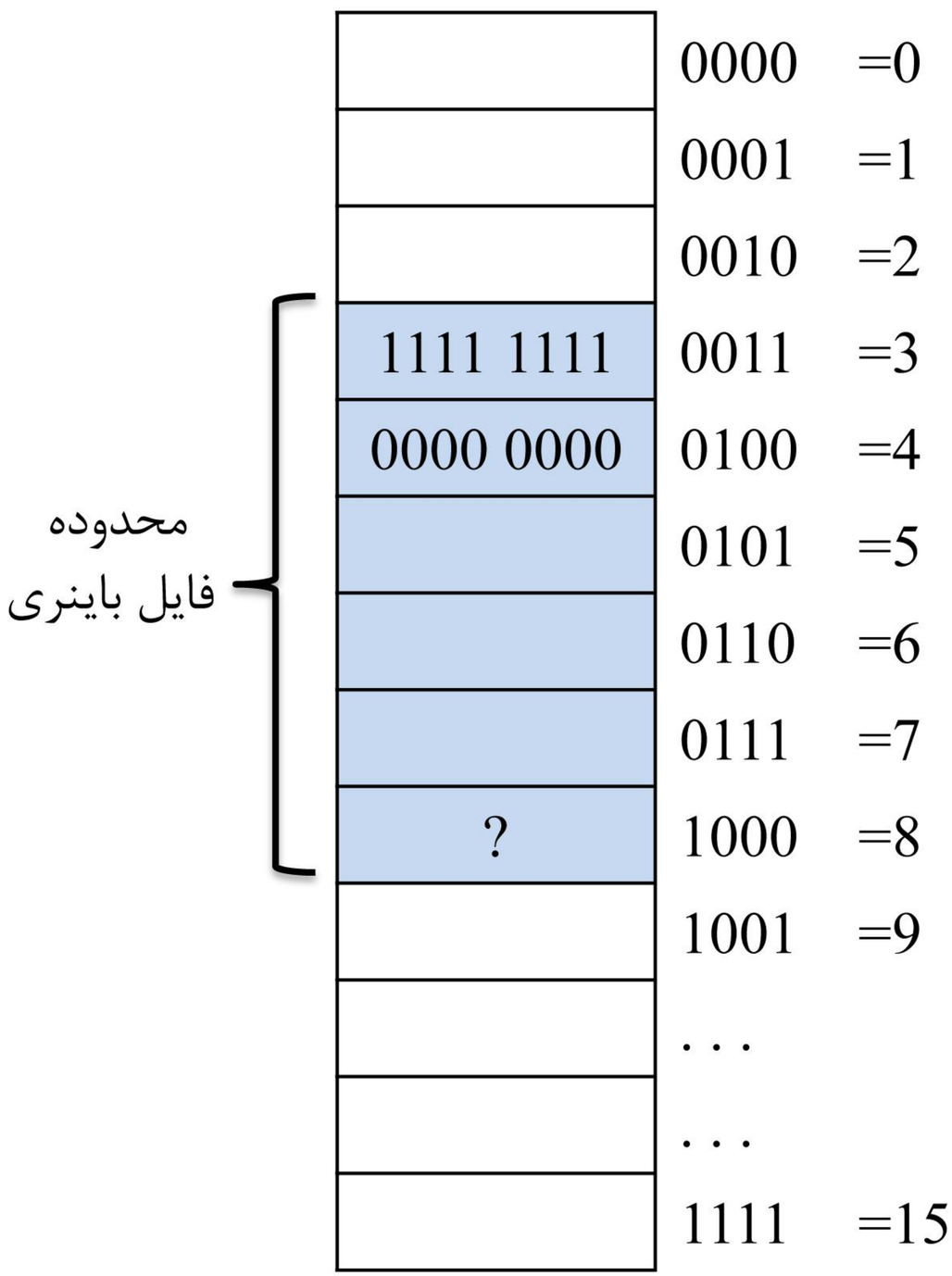
تفاوت متنی و باینری (۲)

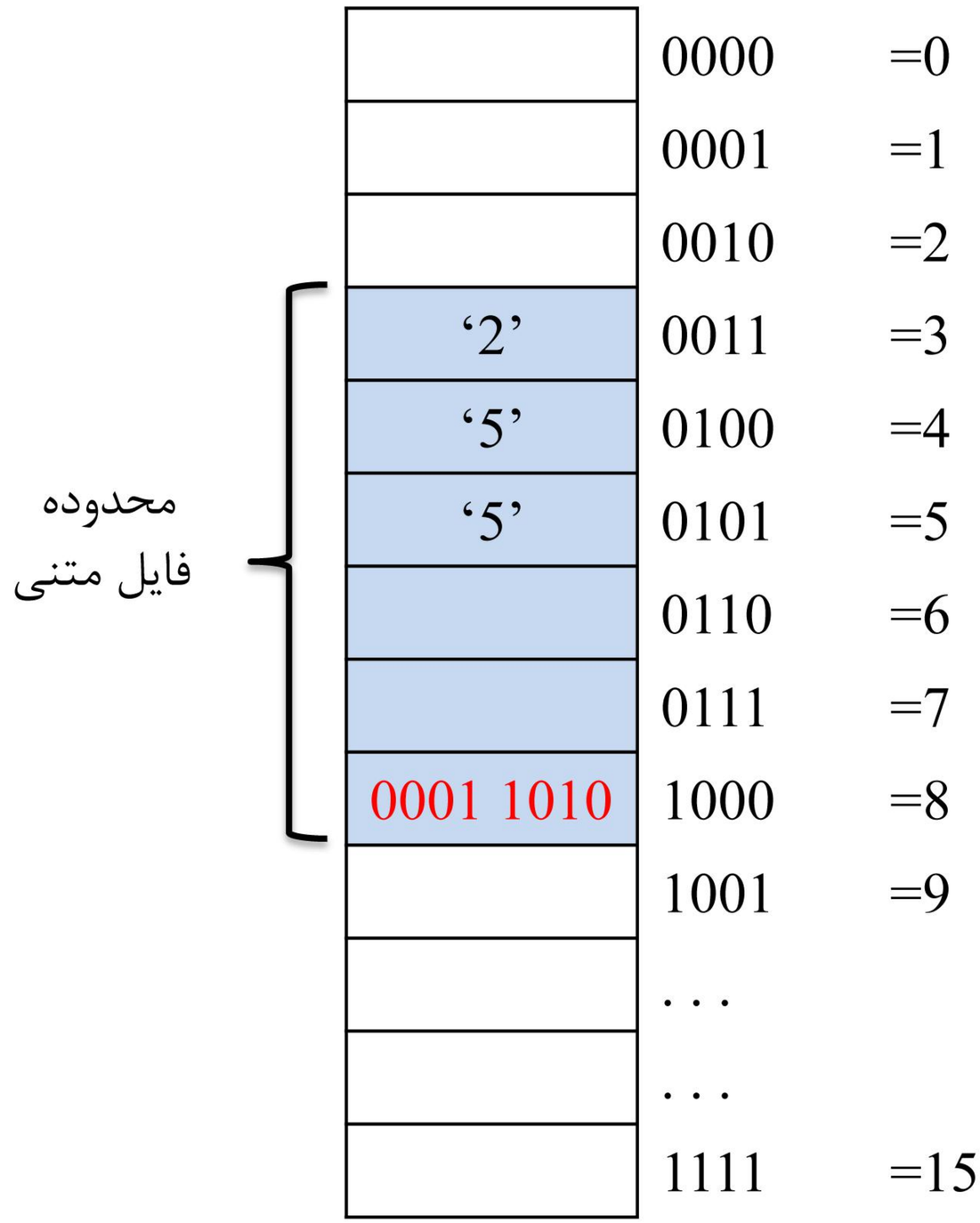
در فایل متنی، کاراکتری که پایان خط را مشخص می‌کند، در حین ذخیره‌شدن بر روی دیسک، باید به کاراکترهای CR/LF (Carriage Return Line Feed) تبدیل شود و در حین خوانده شدن، عکس این عمل باید صورت گیرد: یعنی کاراکترهای CR/LF باید به کاراکتر تعیین کننده پایان خط تبدیل شوند و بدیهی است که این تبدیلات مستلزم وقت است، لذا دسترسی به اطلاعات موجود در فایل‌های متنی کندتر از فایل‌های باینری است.

تفاوت متنی و باینری (۳)

- اختلاف دیگر فایل‌های متنی و باینری در تشخیص انتهای فایل است. در هر دو روش ذخیره‌ی فایل‌ها، طول فایل توسط سیستم نگه‌داری می‌شود و انتهای فایل با توجه به طول فایل مشخص می‌گردد.
- در حالت متنی کاراکتر 1A (در مبنای ۱۶) و یا 26 (در مبنای ۱۰) مشخص کننده‌ی انتهای فایل است (این کاراکتر با فشار دادن کلید CTRL به همراه کلید Z تولید می‌شود). در حین خواندن داده‌ها از روی فایل متنی، وقتی کنترل به این کاراکتر رسید بیانگر این است که داده‌های موجود در فایل تمام شده‌اند.
- در فایل باینری ممکن است عدد 1A (در مبنای ۱۶) و یا 26 (در مبنای ۱۰) جزئی از اطلاعات بوده، بیانگر انتهای فایل نباشد. لذا نحوه تشخیص انتهای فایل در فایل باینری با فایل متنی متفاوت است.

مثال


محدوده
فایل باینری
1111 1111
0000 0000
?
0000 =0
0001 =1
0010 =2
0011 =3
0100 =4
0101 =5
0110 =6
0111 =7
1000 =8
1001 =9
. . .
. . .
1111 =15


محدوده
فایل متنی
‘2’
‘5’
‘5’
0001 1010
0000 =0
0001 =1
0010 =2
0011 =3
0100 =4
0101 =5
0110 =6
0111 =7
1000 =8
1001 =9
. . .
. . .
1111 =15

سازمان فایل‌ها

- منظور از سازمان فایل این است که اطلاعات در فایل چگونه ذخیره می‌شوند و سپس به چه روش‌هایی مورد بازیابی قرار می‌گیرند. به عبارت دیگر

قانون حاکم بر نحوه ذخیره و بازیابی داده‌ها را در فایل، سازمان فایل گویند.

- دو سازمان فایل اصلی:
  1. سازمان فایل ترتیبی (sequential)
  2. سازمان فایل تصادفی (random)

سازمان فایل ترتیبی

- در سازمان فایل ترتیبی، رکوردها به همان ترتیبی که از ورودی خوانده می‌شوند در فایل قرار می‌گیرند و در هنگام بازیابی، به همان ترتیبی که در فایل ذخیره شده‌اند مورد بررسی قرار می‌گیرند.

مثال:

اگر صدمین رکورد فایل بخواهد مورد دسترسی قرار گیرد، باید ۹۹ رکورد قبل از آن، از فایل خوانده شوند.

- فایل‌های ترتیبی معمولاً دارای یک فیلد کلید هستند و براساس آن، مرتب می‌باشند.
- فیلد کلید، فیلدی است که به عنوان شاخص رکورد مورد استفاده قرار می‌گیرد.

مثال:

در مورد دانشجویان، شماره دانشجویی و در مورد کارمندان شماره کارمندی، فیلد خوبی برای شاخص فرد می‌باشند.

سازمان فایل تصادفی

- در سازمان فایل تصادفی، به هر رکورد یک شماره اختصاص می‌یابد. لذا اگر فایل دارای n رکورد باشد. رکوردها از ۱ تا n شماره‌گذاری خواهند شد.
- وقتی که رکوردی در فایلی با سازمان تصادفی قرار گرفت، محل آن توسط یک الگوریتم پیداکننده آدرس، که با فیلد کلید ارتباط دارد مشخص می‌شود.
- پس دو رکورد با فیلد کلید مساوی، نمی‌توانند در فایل تصادفی وجود داشته باشند.
- در سازمان فایل تصادفی مستقیماً می‌توان به هر رکورد دلخواه دسترسی پیدا کرد (بدون این‌که رکوردهای قبل از آن خوانده شوند).

مثال

فایل ترتیبی

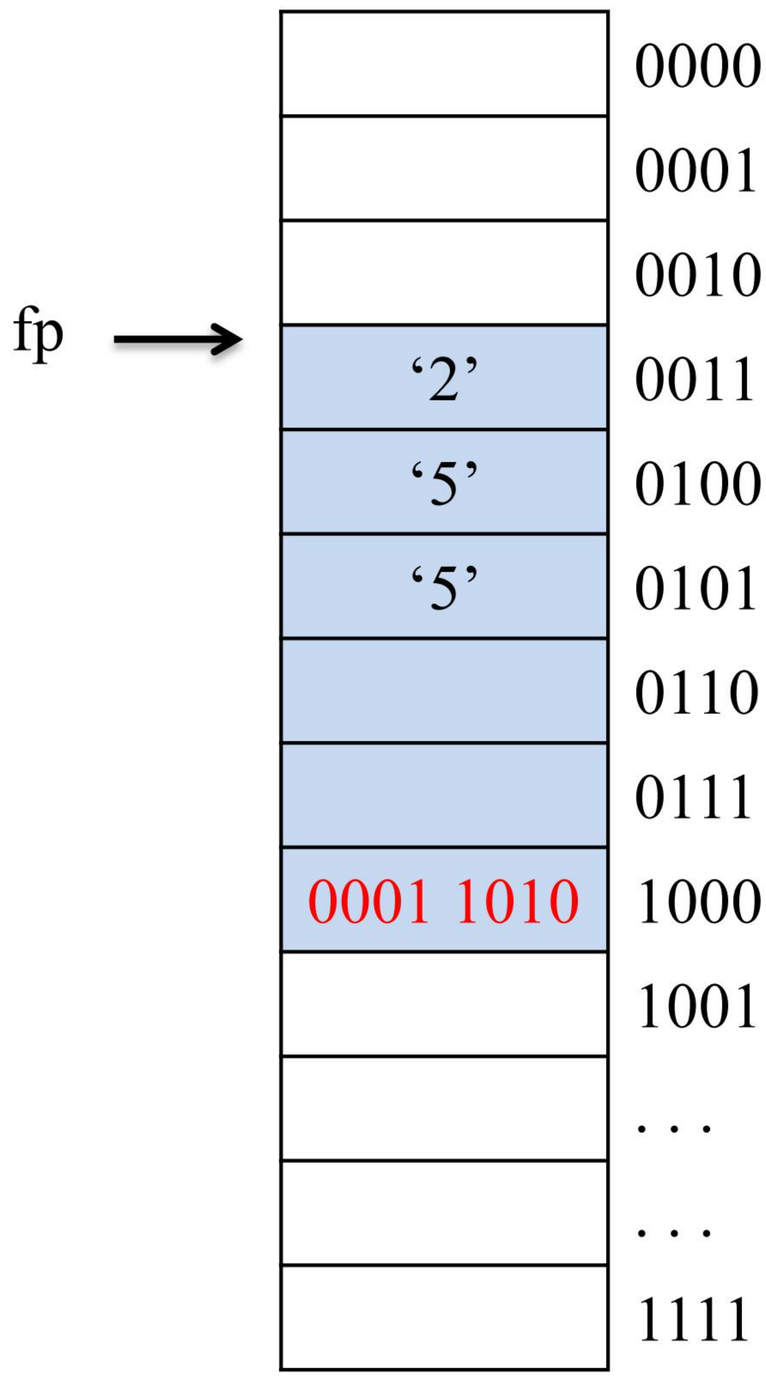
نمره	موضوع درس	نام	شماره شناسایی
20	PASCAL	ALI	12
15	PASCAL	AHMAD	23
18	PASCAL	REZA	34
20	C	JAFAR	56

فایل تصادفی

نمره	موضوع درس	نام	شماره شناسایی	شماره رکورد
20	PASCAL	ALI	12	13
20	C	JAFAR	56	20
15	PASCAL	AHMAD	23	24
18	PASCAL	REZA	34	31

فایل‌ها در زبان C

- برای استفاده از فایل‌ها در زبان C باید به نکات زیر توجه کرد:
- مدیریت فایل‌ها در اختیار سیستم عامل می‌باشد پس باید از سیستم عامل اجازه استفاده از فایل را دریافت کنیم.
  - فایل‌ها را ابتدا باید باز کرد.
  - سپس می‌توان با توجه به عملیات دلخواه، اطلاعاتی را از فایل خوانده و یا در آن نوشت.
  - در انتها نیز باید فایل را بست.
- پس باید مکانیزمی برای شناسایی محل فایل روی حافظه مشخص کرد. برای این کار از اشاره‌گر فایل استفاده می‌شود.
- اشاره‌گر فایل آدرس محل فایل را تعیین می‌کند.



بازکردن فایل (۱)

- هر فایل قبل از این‌که بتواند مورد استفاده قرار گیرد، باید باز شود. مواردی که در حین بازکردن فایل مشخص می‌شوند عبارتند از :
  ۱) نام فایل
  ۲) نوع فایل از نظر ذخیره اطلاعات متنی یا باینری
  ۳) نوع فایل از نظر ورودی-خروجی (آیا فایل فقط به عنوان ورودی است، آیا فقط به عنوان خروجی است یا هم به عنوان ورودی است و هم به عنوان خروجی)
- انواع فایل از نظر ورودی-خروجی:
  - یک فایل ممکن است طوری باز شود که فقط عمل نوشتن اطلاعات برروی آن مجاز باشد. به چنین فایلی، **فایل خروجی** گفته می‌شود.
  - اگر فایل طوری بازگردد که فقط عمل خواندن اطلاعات از آن امکان‌پذیر باشد به چنین فایلی، **فایل ورودی** گفته می‌شود.
  - اگر فایل طوری باز شود که هم عمل نوشتن اطلاعات برروی آن مجاز باشد و هم عمل خواندن اطلاعات از آن، به چنین فایلی، **فایل ورودی-خروجی** گفته می‌شود.
- اگر فایلی قبلاً وجود داشته باشد و به عنوان خروجی بازگردد، اطلاعات قبلی آن از بین می‌رود.

بازکردن فایل (۲)

- برای باز کردن فایل از تابع fopen() استفاده می‌گردد. این تابع که در فایل stdio.h قرار دارد به صورت زیر به‌کار می‌رود:

اشاره‌گر فایل = fopen (“filename” , “mode”);

- در این الگو، نام فایل به رشته‌ای اشاره می‌کند که حاوی نام فایل و محل تشکیل یا وجود آن است.
- نام فایل داده، از قانون نام‌گذاری فایل برنامه تبعیت می‌کند و شامل دو قسمتِ نام و پسوند است.
- بهتر است پسوند فایلِ داده، dat انتخاب شود.
- محل تشکیل یا وجود فایل می‌تواند شامل نام درایو و یا هر مسیر موجود روی دیسک باشد (یعنی آدرس کامل فایل در سیستم عامل).
- mode مشخص می‌کند که فایل چگونه باید باز شود (ورودی، خروجی و یا ورودی-خروجی).

mode

جدول مقادیر معتبر برای mode در تابع fopen()

mode	مفهوم
r(rt)	فایلی از نوع text را به عنوان ورودی باز می‌کند.
w(wt)	فایلی از نوع text را به عنوان خروجی باز می‌کند.
a(at)	فایل را طوری باز می‌کند که بتوان اطلاعاتی را به انتهای آن اضافه نمود.
rb	فایلی از نوع باینری را به عنوان ورودی باز می‌کند.
wb	فایلی از نوع باینری را به عنوان خروجی باز می‌کند.
ab	فایل موجود از نوع باینری را طوری باز می‌کند که بتوان اطلاعات را به انتهای آن اضافه نمود.
r+(r+t)	فایل موجود از نوع text را به عنوان ورودی و خروجی باز می‌کند.
w+(w+t)	فایلی از نوع text را به عنوان ورودی و خروجی باز می‌کند.
a+(a+t)	فایل موجود از نوع text را به عنوان ورودی و خروجی باز می‌کند.
r+b	فایل موجود از نوع باینری را به عنوان ورودی و خروجی باز می‌کند.
w+b	فایل از نوع باینری را به عنوان ورودی و خروجی باز می‌کند.
a+b	فایل از نوع باینری را به عنوان ورودی و خروجی باز می‌کند. (این فایل می‌تواند قبلاً وجود داشته باشد)

بازکردن فایل(۳)

- برای بازکردن فایل باید یک اشاره‌گر از نوع فایل تعریف گردد تا به فایلی که توسط تابع fopen باز می‌شود اشاره نماید. اگر فایل به دلایلی باز نشود این اشاره‌گر برابر با NULL خواهد بود. مثال:

```
FILE   *fp;
fp = fopen("C:\\test.dat", "w");
```

- FILE ماکرویی در فایل stdio.h است.
- برای تشخیص این‌که آیا فایل با موفقیت باز شده است یا خیر می‌توان اشاره‌گر فایل را با NULL مقایسه کرد. (NULL ماکرویی است که در فایل stdio.h تعریف شده است و با حروف بزرگ به کار می‌رود)، اگر فایل برابر با NULL باشد بدین معنی است که فایل بازنشده است:

```
FILE   *fp;
if ((fp=fopen("C:\\test.dat" , "w")) == NULL)
{
  printf("can not open file \n");
  exit(0);
}
```

بستن فایل

پس از اینکه برنامه‌نویس کارش را با فایل تمام کرد، باید آن را ببندد. بستن فایل توسط تابع fclose() انجام می‌شود که بصورت زیر مورد استفاده قرار می‌گیرد:

fclose(اشاره‌گرفایل);

مثال:

fclose(fp);

این دستور فایلی را که fp به آن اشار می‌کند، می‌بندد.

- اگر چندین فایل به‌طور همزمان در برنامه باز باشند می‌توان آن‌ها را با تابع fcloseall() بست. این تابع بصورت زیر به کار می‌رود:

fcloseall();

مثال

```
#include <stdio.h>

int main()
{
FILE   *fp;
if ((fp=fopen("C:\\test.dat" , "w")) == NULL)
{
  printf("can not open file \n");
  exit(0);
}

عملیات‌های دلخواه روی فایل (خواندن و نوشتن)

fclose(fp);
return 0;
}
```

طرح کلی برنامه‌هایی که از فایل استفاده می‌کنند.

پایان بخش اول- جلسه یازدهم

برنامه‌سازی کامپیوتر

جلسه یازدهم - بخش دوم

فایل در زبان C

طرح کلی

- فایل‌ها
  - ورودی-خروجی کاراکترها
  - ورودی-خروجی رشته‌ها
  - ورودی-خروجی همراه با فرمت
  - فایل‌های ورودی و خروجی
  - بازگشت به ابتدای فایل
  - حذف فایل
  - دستگاه‌های ورودی-خروجی استاندارد

ورودی-خروجی کاراکترها

- برای نوشتن یک کاراکتر در فایل، از توابع putc() و fputc() استفاده می‌شود. عملکرد این دو تابع یکسان است. تابع putc() در گونه‌های جدید C و fputc() در گونه‌های قدیمی C وجود داشته است. سرعت تابع putc() از تابع fputc() بیش‌تر است. الگوی تابع putc() بصورت زیر است:

putc(اشاره‌گر فایل , کاراکتر);

- برای خواندن کاراکترها از فایل، می‌توان از دو تابع getc() و fgetc() استفاده نمود. نحوه بکارگیری این دو تابع نیز یکسان است. تابع fgetc() در گونه‌های قدیمی C و getc() در گونه‌های جدید C وجود دارد. سرعت اجرای تابع getc() از تابع fgetc() بیش‌تر است. لذا از تابع getc() استفاده می‌شود. الگوی این تابع به صورت زیر است:

```
char ch;
ch = getc(اشاره‌گر فایل);
```

مثال

مثال: برنامه‌ای بنویسید که جمله “This is a statement ! ” را در یک فایل ذخیره کند.

```
#include <stdio.h>
int main()
{
   char msg[100]=“This is a statement ! ”;
   int i=0;
   FILE  *fp;
   fp = fopen(“C:\\test.dat” , “w”);
   while(msg[i] != ‘\0’)
   {
      putc(msg[i] , fp);
      i ++ ;
    }
   fclose(fp);
   return 0;
}
```

نکته (۱)

- وقتی کاراکترهایی بر روی فایل نوشته می‌شوند، باید مکان بعدی، که کاراکتر آتی در آنجا قرار می‌گیرد مشخص باشد.
- همچنین وقتی که کاراکترهایی از فایل خوانده می‌شوند باید مشخص باشد که تاکنون تا کجای فایل خوانده شده است و کاراکتر بعدی از کجا باید خوانده شود.
- برای این منظور، سیستم از یک متغییر به نام موقعیت‌سنج فایل (file position) استفاده می‌کند که با هر دستور خواندن و یا نوشتن برروی فایل، مقدار این متغییر به طور اتوماتیک تغییر می‌کند، تا موقعیت فعلی فایل را مشخص نماید.
- پس عمل نوشتن برروی فایل و عمل خواندن از روی آن، از جایی که متغییر موقعیت‌سنج فایل نشان می‌دهد، شروع می‌شود.

نکته (۲)

در هنگام خواندن داده‌ها از فایل باید بتوان انتهای فایل را تست نمود، یعنی در برنامه باید بتوان این تست را انجام داد که اگر در حین خواندن داده‌ها از فایل، موقعیت‌سنج فایل به انتهای فایل رسید دستور خواندن بعدی صادر نگردد. چرا که در غیر این صورت، سیستم، پیام خطایی را مبنی بر نبودن اطلاعات در فایل صادر می‌کند. دو روش برای این کار وجود دارد:

- **روش اول :** در حین خواندن داده‌ها از فایل متنی، پس از رسیدن به انتهای فایل، تابع getc() یا fgetc() علامت EOF را برمی‌گرداند. لذا در هنگام خواندن داده‌ها از فایل متنی می‌توان آنقدر به عمل خواندن ادامه داد، تا این‌که کاراکتر خوانده شده برابر با EOF شود.

```
while( (ch=getc(fp)) != EOF) { . . . }
```

- **روش دوم :** تابع feof() برای تشخیص انتهای فایل‌های باینری و متنی استفاده می‌شود. اگر اشاره‌گر فایلی که feof() به آن اشاره می‌کند به انتهای فایل رسیده باشد، این تابع ارزش درستی وگرنه ارزش نادرستی را برمی‌گرداند.

feof(اشاره‌گر فایل);

مثال

مثال: برنامه‌ای بنویسید که محتویات فایل C:\test.dat را در صفحه نمایش چاپ کند.

```
#include <stdio.h>
int main()
{
   char ch;
   FILE  *fp;
   fp = fopen("C:\\test.dat" , "r");
   while(!feof(fp))
   {
      ch = getc(fp);
      printf("%c", ch);
    }
   fclose(fp);
   return 0;
}
```

ورودی-خروجی رشته‌ها

- برای نوشتن رشته‌ها در فایل، از تابع fputs() و برای خواندن رشته‌ها از فایل، از تابع fgets() استفاده می‌شود:

fputs(اشاره‌گر فایل , رشته کاراکتری);
fgets(متغییر رشته‌ای, طول , اشاره‌گر فایل);

- نحوه عمل تابع fgets() به این صورت است که، از ابتدای فایل شروع به خواندن می‌کند تا به انتهای یک خط برسد و یا رشته‌ای با طول مشخص شده را از فایل بخواند.
- برخلاف تابع gets()، در تابع fgets() کاراکتری که انتهای خط را مشخص می‌کند جزء رشته‌ای خواهد بود که این تابع از فایل می‌خواند.یعنی تابع gets() خودِ ENTER را در رشته قرار نمی دهد اما تابع fgets() خودِ ENTER را نیز در رشته قرار خواهد داد.

مثال

مثال: برنامه‌ای بنویسید که جمله “This is a statement ! ” را در یک فایل ذخیره کند.

```
#include <stdio.h>
int main()
{
   char msg[100]=“This is a statement ! ”;
   int i=0;
   FILE  *fp;
   fp = fopen(“C:\\test.dat” , “w”);
   while(msg[i] != ‘\0’)
   {
      putc(msg[i] , fp);
      i ++ ;
   }
   fclose(fp);
   return 0;
}
```

```
#include <stdio.h>
int main()
{
   char msg[100]=“This is a statement ! ”;

   FILE  *fp;
   fp = fopen(“C:\\test.dat” , “w”);

   fputs(msg,fp);

   fclose(fp);
   return 0;
}
```

مثال

مثال: برنامه‌ای بنویسید که محتویات فایل C:\test.dat را در صفحه نمایش چاپ کند.

```
#include <stdio.h>
int main()
{
   char ch;
   FILE  *fp;
   fp = fopen("C:\\test.dat" , "r");
   while(!feof(fp))
   {
      ch = getc(fp);
      printf("%c", ch);
   }
   fclose(fp);
   return 0;
}
```

```
#include <stdio.h>
int main()
{
   char ch[100];
   FILE  *fp;
   fp = fopen("C:\\test.dat" , "r");
   while(!feof(fp))
   {
      fgets(ch,100,fp);
      printf("%s", ch);
   }

   fclose(fp);
   return 0;
}
```

ورودی و خروجی همراه با فرمت

برای نوشتن داده‌ها در فایل یا خواندن داده‌ها از فایل با فرمت خاص از دو تابع fprintf() و fscanf() استفاده می‌شود. این دو تابع دقیقاً کار توابع printf() و scanf() را در ورودی-خروجی معمولی صفحه نمایش، انجام می‌دهند. این توابع در سرفایل stdio.h قرار دارند و بصورت زیر مورد استفاده قرار می‌گیرند:

fprintf(اشاره‌گر فایل , "رشته کنترلی" , پارامترها);
fscanf(اشاره‌گر فایل , "رشته کنترلی" , پارامترها);

- اشاره‌گر فایل، اشاره‌گری است که مشخص می‌کند اعمال این توابع باید برروی چه فایلی انجام شود.
- رشته کنترلی مشخص می‌کند که داده‌ها (پارامترها) باید با چه فرمتی نوشته و یا خوانده شوند.

مثال:

```
FILE  *fp;
int m=10, n; char ch1='A',ch2;
fp=fopen("C:\\test.dat" , "w+");
. . .
fprintf(fp , "%d , %c" , m , ch1);
fscanf(fp , "%d   %c" , &n , &ch2);
```

نکته

- در مورد توابع fprintf() و fscanf() باید توجه داشت که علیرغم اینکه ورودی-خروجی با این دو تابع آسان است اما اطلاعات به همان صورتی که در صفحه نمایش ظاهر می‌شوند در فایل ذخیره می‌گردند.

مثال:

عدد ۲۵۵ که ۳ بایت را در صفحه نمایش اشغال می‌کند، اگر توسط تابع fprintf() بر روی فایل نوشته شود، در فایل نیز ۳ بایت را اشغال خواهد کرد (در حالی که عدد ۲۵۵ یک عدد صحیح است و می‌تواند در دو بایت ذخیره شود). یعنی هر رقم به صورت یک کاراکتر تلقی می‌گردد.

اگر این عدد توسط تابع fscanf() از روی فایل خوانده شود، باید عمل تبدیل کاراکتر به عدد صورت گیرد که مستلزم صرف وقت است و زمانی که تعداد خواندن داده‌ها از فایل زیاد باشد، زمان زیادی صرف خواهد شد.

برای حل این مشکل از دو تابع fread() و fwrite() استفاده می‌شود.

مثال(۱)

مثال: برنامه‌ای بنویسید که لیست دانشجویان را از کاربر دریافت کرده و در یک فایل ذخیره نماید.

```
#include <stdio.h>
int main() {
   char name[100], ch;
   int i=0;
   FILE  *fp;
   fp = fopen("C:\\test.dat" , "w");

   do{
      i++;
      printf("\nEnter %d th student name: ", i);
      gets(name);
      fprintf(fp,"%d %s\n", i , name);
      printf("Do you want to continue(y/n)? ");
      ch=getche();
   }while (ch=='y');
   fclose(fp);
   getch();
   return 0;
}
```

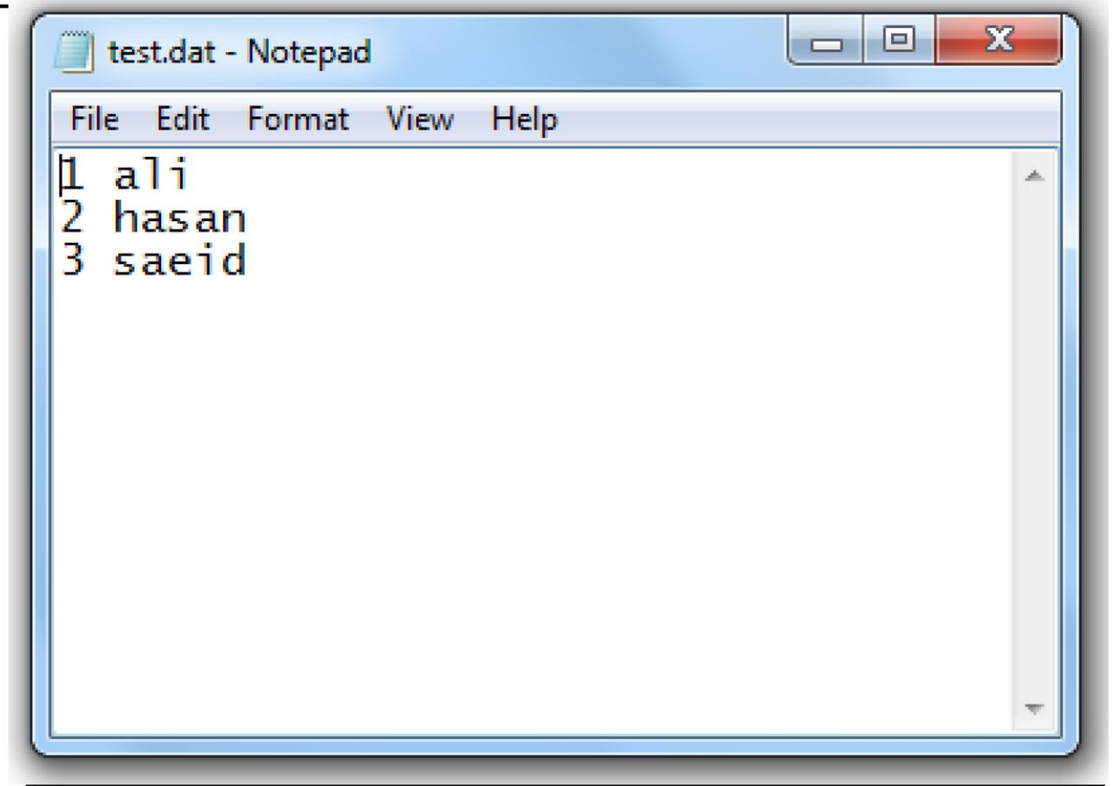
```
Enter 1 th student name: ali
Do you want to continue(y/n)? y
Enter 2 th student name: hasan
Do you want to continue(y/n)? y
Enter 3 th student name: saeid
Do you want to continue(y/n)? n
```

مثال(۲)

مثال: برنامه‌ای بنویسید که لیست دانشجویان را از فایل خوانده و به ترتیب در صفحه خروجی نمایش دهد.

```
#include <stdio.h>
int main() {
   char name[100];
   int i;
   FILE  *fp;
   fp = fopen("C:\\test.dat" , "r");

   while (!feof(fp)){
       fscanf(fp,"%d %s\n", &i , &name);
       printf("\n%d %s", i , name);
   }
   fclose(fp);
   getch();
   return 0;
}
```




```
1 ali
2 hasan
3 saeid
```

فایل‌های ورودی-خروجی

برای اینکه بتوان فایل را هم به عنوان ورودی و هم خروجی درنظر گرفت، کافی است در تابع fopen() به جای mode از یکی از عبارات زیر استفاده نمود.

r+ یا r+t	بازکردن فایل متنی موجود به عنوان ورودی و خروجی
w+ یا w+t	ایجاد یک فایل متنی به عنوان ورودی و خروجی
a+ یا a+t	ایجاد فایل متنی و یا بازکردن فایل متنی موجود به عنوان ورودی و خروجی
r+b	بازکردن فایل باینری موجود به عنوان ورودی و خروجی
w+b	ایجاد یک فایل باینری به عنوان ورودی و خروجی
a+b	ایجاد و یا بازکردن فایل موجود باینری به عنوان ورودی و خروجی

بازگشت به ابتدای فایل

- در حین کار با فایل‌ها (نوشتن اطلاعات بر روی آن‌ها و یا خواندن اطلاعات از آن‌ها) برای برگشت به ابتدای فایل (تغییر "موقعیت سنج فایل"، به طوری که به ابتدای فایل اشاره کند) باید فایل را بسته و مجدداً آن را باز نمود.
- شاید در فایل‌هایی که فقط به عنوان خروجی و یا فقط به عنوان ورودی باز می‌شوند، نیاز به برگشت به ابتدای فایل (بدون بستن و بازکردن مجدد آن) احساس نشود، ولی این امر در مورد فایل‌های ورودی و خروجی یک نیاز ضروری است. برای این منظور، از تابعی به نام rewind() استفاده می‌گردد.
- الگوی تابع rewind() در فایل stdio.h قرار داشته و بصورت زیر است:

rewind(اشاره‌گر فایل);

مثال (۱)

مثال: برنامه‌ای بنویسید که جمله “This is a statement ! ” را در یک فایل ذخیره کند و در ادامه، محتویات این فایل را خوانده و در صفحه نمایش چاپ کند.

```
#include <stdio.h>
int main()
{
  char msg[100]=“This is a statement ! ”;
  int i=0;
  FILE  *fp;
  char ch;

  fp = fopen(“C:\\test.dat” , “w”);
  while(msg[i] != ‘\0’)
  {
     putc(msg[i] , fp);
     i ++ ;
   }
```

```
  fclose(fp);
  fp = fopen(“C:\\test.dat” , “r”);

  while(!feof(fp))
  {
     ch = getc(fp);
     printf(“%c”, ch);
   }

  fclose(fp);

  return 0;
}
```

مثال (۲)

مثال: برنامه‌ای بنویسید که جمله “This is a statement ! ” را در یک فایل ذخیره کند و در ادامه، محتویات این فایل را خوانده و در صفحه نمایش چاپ کند.

```
#include <stdio.h>
int main()
{
  char msg[100]=“This is a statement ! ”;
  int i=0;
  FILE  *fp;
  char ch;

  fp = fopen(“C:\\test.dat” , “w+”);
  while(msg[i] != ‘\0’)
  {
     putc(msg[i] , fp);
     i ++ ;
   }
```

```
  rewind(fp);

  while(!feof(fp))
  {
     ch = getc(fp);
     printf(“%c”, ch);
   }

  fclose(fp);

  return 0;
}
```

حذف فایل

- برای حذف فایل‌های غیرضروری می‌توان از تابع ()remove استفاده کرد. الگوی این تابع در فایل stdio.h قرار دارد:

remove(“آدرس و نام فایل”);

- اگر عمل تابع با موفقیت انجام شود مقدار صفر وگرنه مقداری غیر از صفر را برمی‌گرداند.

مثال:

```
#include <stdio.h>
int main()
{
   if (remove("C:\\test.dat"))
      printf("I can't delete file.");
   else
      printf("File deleted successfully.");
getch();
   return 0;
}
```

دستگاه‌های ورودی و خروجی استاندارد

- در زبان C فایل داده می‌تواند هر دستگاهی مثل صفحه نمایش، صفحه کلید، چاپگر، ترمینال، دیسک، نوار و ... باشد.
- وقتی اجرای یک برنامه به زبان C آغاز می‌شود، ۵ فایل به‌طور اتوماتیک باز می‌شوند که اشاره‌گرهای آن‌ها در جدول زیر آمده است.
- این اشاره‌گرها بصورت اتوماتیک نیز بسته می‌شوند و نیازی به این نیست که برنامه‌نویس آن‌ها را ببندد.

اشاره‌گر دستگاه (فایل)	نام دستگاه (فایل)
stdin	دستگاه ورودی استاندارد (صفحه کلید)
stdout	دستگاه خروجی استاندارد (صفحه نمایش)
stderr	دستگاه استاندارد جهت ثبت پیام‌های خطا (صفحه نمایش)
stdprn	دستگاه استاندارد چاپ (چاپگر موازی)
stdaux	پورت سری (serial port)

- مثال:

putc(ch, stdout);	کاراکتر ch در صفحه نمایش چاپ می‌شود.
fprintf(stdout, “%d , %d” , a , b);	متغییرهای a و b در صفحه نمایش نوشته می‌شوند.
fscanf(stdin , “%d %d” , &x , &y);	متغییرهای x و y از صفحه کلید خوانده می‌شوند.

جمع‌بندی

- فایل‌ها
  - فایل
  - روش‌های ذخیره داده‌ها در فایل
  - انواع فایل و تفاوت‌های آن‌ها
    - متنی
    - باینری
  - سازمان فایل‌ها
    - ترتیبی
    - تصادفی
  - اشاره‌گر فایل
  - بازکردن فایل
  - بستن فایل
  - ورودی-خروجی کاراکترها
  - ورودی-خروجی رشته‌ها
  - ورودی-خروجی همراه با فرمت
  - فایل‌های ورودی و خروجی
  - بازگشت به ابتدای فایل
  - حذف فایل
  - دستگاه‌های ورودی-خروجی استاندارد

تمرینات تکمیلی

برنامه‌های زیر را به ترتیب با استفاده از یکی از کامپایلرهای زبان C طراحی، پیاده‌سازی و اجرا کنید.

۱) برنامه‌ای بنویسید که تمامی حروف انگلیسی بزرگ و کوچک را در یک فایل قرار دهد. (در یک سطر حروف بزرگ و در سطر بعدی حروف کوچک قرار گیرند)

۲) برنامه‌ای بنویسید که فایلی را بصورت متنی باز کرده و متن دریافتی از کاربر را در آن ذخیره کند. آخرین کاراکتری که توسط کاربر وارد می‌شود کاراکتر & است.

۳) برنامه‌ای بنویسید که حروف انگلیسی را به همراه کد اسکی آن‌ها در یک فایل ذخیره کند. در هر سطر یک حرف انگلیسی و در مقابل آن کد اسکی آن نوشته شود.

۴) تابعی بنویسید که نام دو فایل را گرفته و محتویات فایل اول را به فایل دوم اضافه کند.

۵) تابعی بنویسد که نام فایلی را دریافت کرده و تعداد کاراکترهای آن را شمرده و بازگرداند.

۶) تابعی بنویسد که نام فایلی را دریافت کرده و تعداد سطرهای آن را شمرده و بازگرداند.