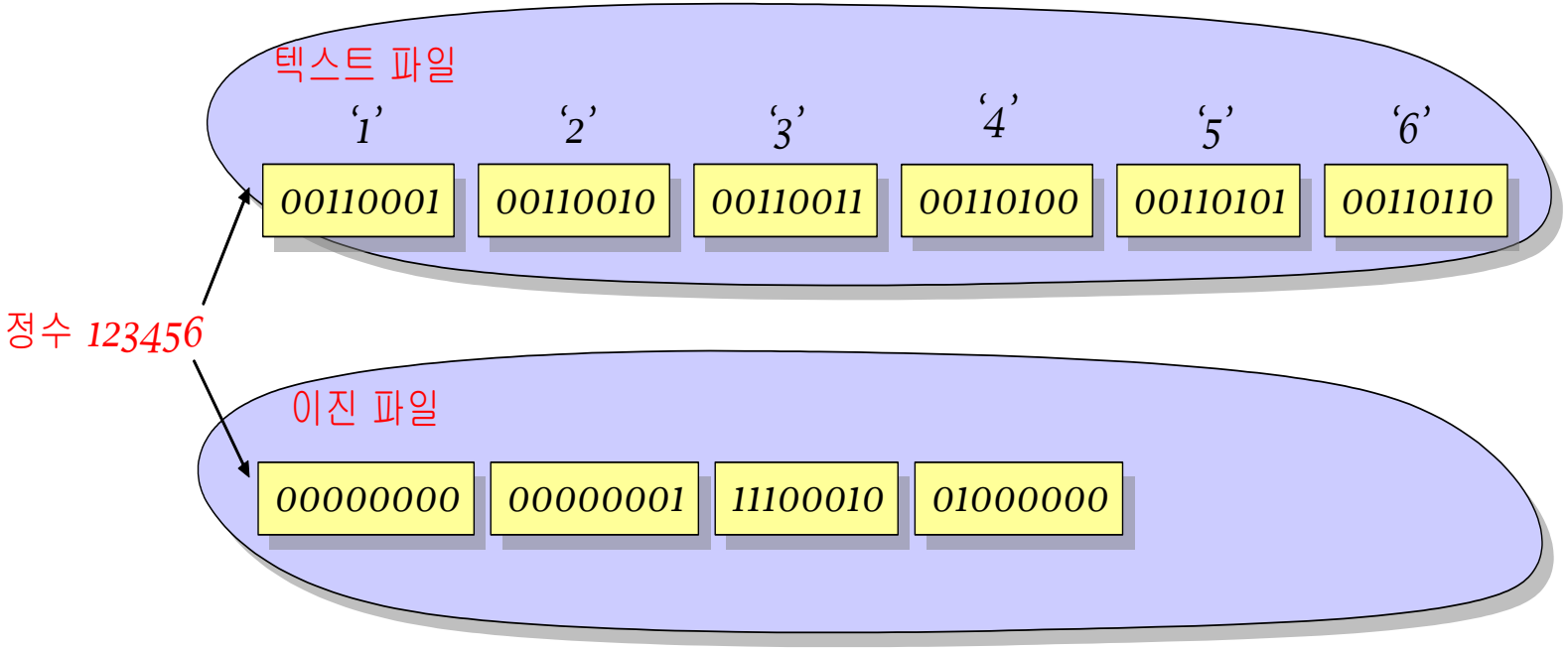
파일 입출력 Part 2

이진 파일 입출력

- 텍스트 파일과 이진 파일의 차이점
  - 텍스트 파일: 모든 데이터가 아스키 코드로 변환되어서 저장됨
  - 이진 파일: 컴퓨터에서 데이터를 표현하는 방식 그대로 저장



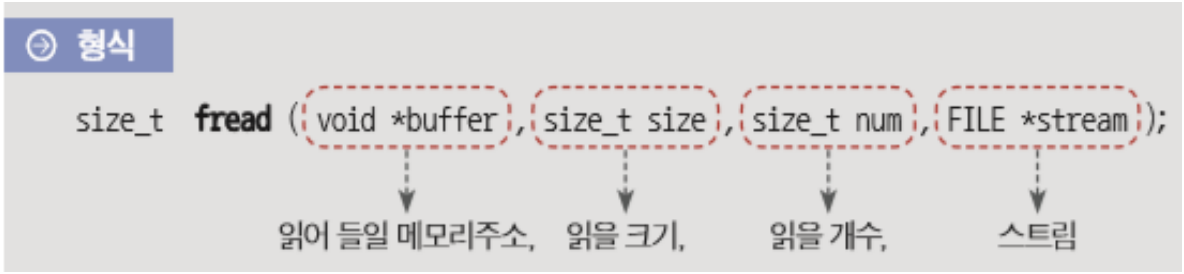
이진 파일의 생성

파일 모드	설명
“rb"	읽기 모드 + 이진 파일 모드
“wb"	쓰기 모드 + 이진 파일 모드
“ab"	추가 모드 + 이진 파일 모드
“rb+"	읽고 쓰기 모드 + 이진 파일 모드
"wb+"	쓰고 읽기 모드 + 이진 파일 모드

```
1.  int main(void)
2.  {
3.          FILE *fp = NULL;
4.
5.          fp = fopen("binary.txt", "rb");
6.
7.          if( fp == NULL )
8.                  printf("이진 파일 열기에 실패하였습니다.₩n");
9.          else
10.                 printf("이진 파일 열기에 성공하였습니다.₩n");
11.
12.         if( fp != NULL ) fclose(fp);
13. }
```

이진 파일 읽기

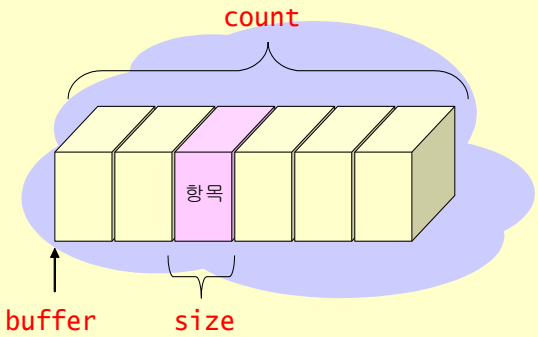
```
size_t fread( void *buffer,  size_t  size, size_t  num, FILE *fp );
```



- fread()

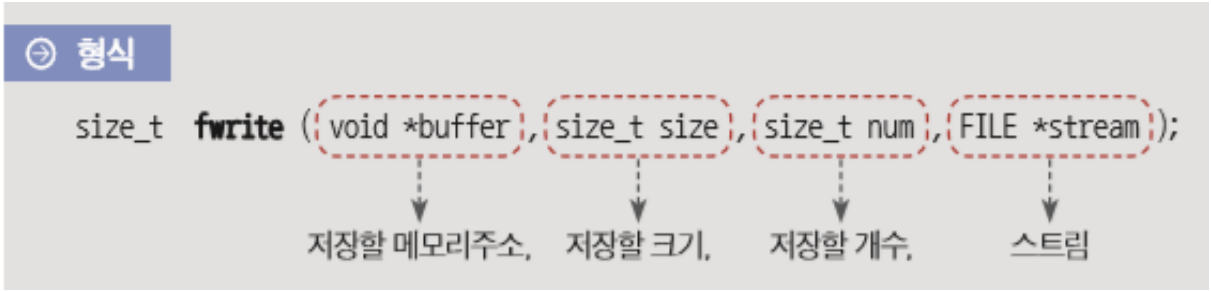
- fread() 함수는 stream과 관련된 파일에서 num개의 개체를(개체는 size 길이) buffer 가 가리키는 버퍼로 읽어 들임.
- 실제 읽은 개체의 수를 반환함.

```
1.  #include <stdio.h>
2.  #define SIZE 1000

3.  int main(void)
4.  {
5.          float buffer[SIZE];
6.          FILE *fp = NULL;
7.          size_t size;

8.          fp = fopen("binary.txt", "rb");
9.          if( fp == NULL )
10.         {
11.                 fprintf(stderr, "binary.txt 파일을 열 수 없습니다.");
12.                 exit(1);
13.         }
14.         size = fread(buffer, sizeof(float), SIZE, fp);
15.         if( size != SIZE )
16.         {
17.                 fprintf(stderr, "읽기 동작 중 오류가 발생했습니다.₩n");
18.         }
19.         fclose(fp);

20.         return 0;
21. }
```



이진 파일 쓰기

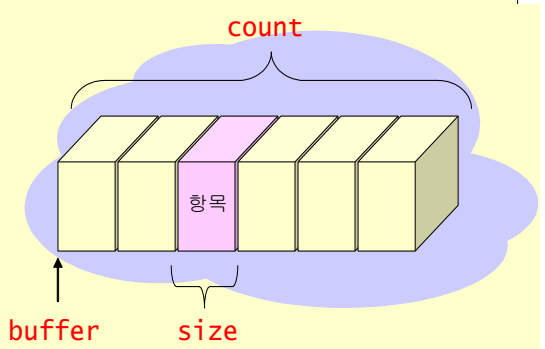
```
size_t fwrite( void *buffer, size_t size, size_t num,    FILE *fp);
```



- fwrite()

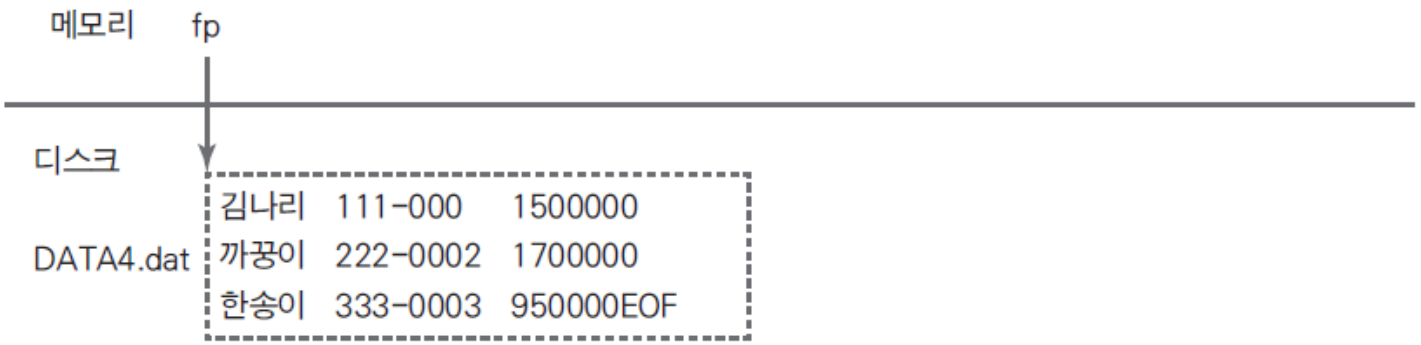
- fwrite() 함수는 buffer가 가리키는 버퍼에서 num개의 개체를(개체는 size 길이) stream과 관련된 파일에 씀.

```
1.  #include <stdio.h>

2.  int main(void)
3.  {
4.          int buffer[] = { 10, 20, 30, 40, 50 };
5.          FILE *fp = NULL;
6.          size_t i, size, count;

7.          fp = fopen("binary.txt", "wb");
8.          if( fp == NULL )
9.          {
10.                 fprintf(stderr, "binary.txt 파일을 열 수 없습니다.");
11.                 exit(1);
12.         }

13.         size = sizeof(buffer[0]);
14.         count = sizeof(buffer) / sizeof(buffer[0]);

15.         i = fwrite(buffer, size, count, fp);
16.         return 0;
17. }
```



예제 C 프로그램은 자료구조 표현 시 일반적으로 구조체를 가장 많이 사용하게 된다. 구조체 배열의 데이터를 파일에 저장하고 읽어보자.

	name	telno	salary
emps[0]	김나리	111-000	1500000
emps[0]	까꿍이	222-0002	1700000
emps[0]	한송이	333-0003	950000



이진 입출력, 13_4

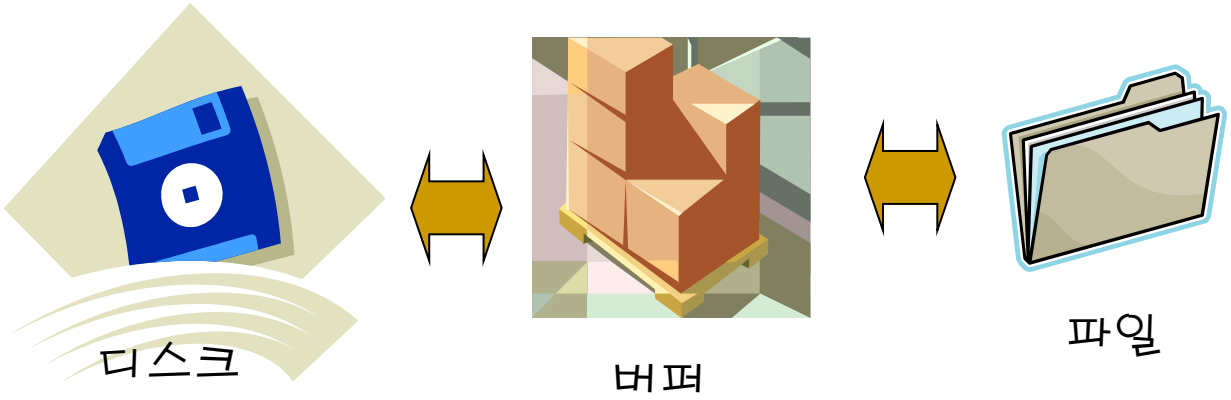
```
4   struct EMP {  char name[20];
5                 char telno[20];
6                 int salary;
7   } emps[3]={  {"김나리", "111-0001", 150000},
8                {"까꿍이", "222-0002", 1700000},
9                {"한송이", "333-0003", 950000}  };
10
11  int main()
12  {
13      FILE *fp;
14      struct EMP temp;
15
16      if((fp = fopen("DATA4.dat", "wb")) == NULL)
17      {
18          printf("file open error. \n");
19          exit(1);
21
22      fwrite(emps, sizeof(emps), 1, fp);  <------ 파일 저장
23      //fwrite(emps, sizeof(struct EMP), 3, fp);  //파일 저장
24
25      fp = freopen("DATA4.dat", "rb", fp);
26
27      printf("\n데이터 파일 Load \n");
28      while(1)
29      {
30          if(fread(&temp, sizeof(temp), 1, fp) != 1)  <------ 파일 읽기
31              break;
32
33          printf("%s, %s, %d \n", temp.name, temp.telno, temp.salary);
34      }
35      fclose(fp);
36      printf("\n");
37
38      return 0;
39  }
```

실행결과

```
데이터 파일 Load
김나리, 111-0001, 150000
까꿍이, 222-0002, 1700000
한송이, 333-0003, 950000
```

버퍼링

- fopen()을 사용하여 파일을 열면, 버퍼가 자동으로 만들어진다.
- 버퍼는 파일로부터 읽고 쓰는 데이터의 임시 저장 장소로 이용되는 메모리의 블록
- 디스크 드라이브는 블록 단위 장치이기 때문에 블록 단위로 입출력을 해야만 가장 효율적으로 동작
- 파일과 연결된 버퍼는 파일과 물리적인 디스크 사이의 인터페이스로 사용



binary_file.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #define SIZE 3

4.  struct student {
5.          int number;             // 학번
6.          char name[20];          // 이름
7.          double gpa;             // 평점
8.  };

9.  int main(void)
10. {
11.         struct student table[SIZE] = {
12.                 { 1, "Kim", 3.99 },
13.                 { 2, "Min", 2.68 },
14.                 { 3, "Lee", 4.01 }
15.         };
16.         struct student s;
17.         FILE *fp = NULL;
18.         int i;
19.         // 이진 파일을 쓰기 모드로 연다.
20.         if( (fp = fopen("student.dat", "wb")) == NULL )
21.         {
22.                 fprintf(stderr,"출력을 위한 파일을 열 수 없습니다.₩n");
23.                 exit(1);
24.         }

25.
```

```
1.          // 배열을 파일에 저장한다.
2.          fwrite(table, sizeof(struct student), SIZE, fp);
3.          fclose(fp);

4.          // 이진 파일을 읽기 모드로 연다.
5.          if( (fp = fopen("student.dat", "rb")) == NULL )
6.          {
7.                      fprintf(stderr,"입력을 위한 파일을 열 수 없습니다.₩n");
8.                      exit(1);
9.          }

10.         for(i = 0;i < SIZE; i++)
11.         {
12.                     fread(&s, sizeof(struct student), 1, fp);
13.                     printf("학번 = %d, 이름 = %s, 평점 = %f₩n", s.number, s.name, s.gpa);
14.         }
15.         fclose(fp);

16.         return 0;
17. }
```

학번 = 1, 이름 = Kim, 평점 = 3.990000
학번 = 2, 이름 = Min, 평점 = 2.680000
학번 = 3, 이름 = Lee, 평점 = 4.010000

fappend.c

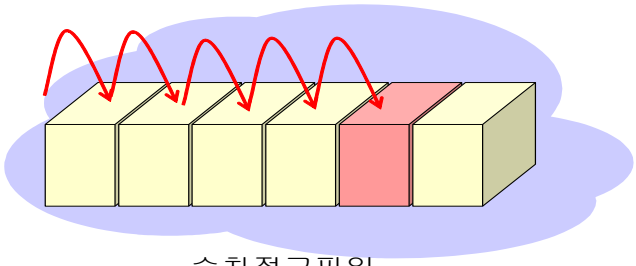
```
1.  #include <stdio.h>
2.  #include <stdlib.h>

3.  int main(void)
4.  {
5.          FILE *fp1, *fp2;
6.          char file1[100], file2[100];
7.          char buffer[1024];
8.          int count;

9.          printf("첫번째 파일 이름: ");
10.         scanf("%s", file1);

11.         printf("두번째 파일 이름:  ");
12.         scanf("%s", file2);

13.         // 첫번째 파일을 이진파일 읽기 모드로 연다.
14.         if( (fp1 = fopen(file1, "rb")) == NULL )
15.         {
16.                 fprintf(stderr,"파일을 열 수 없습니다.₩n");
17.                 exit(1);
18.         }
```

```
1.              // 두번째 파일을 추가 모드로 연다.
2.              if( (fp2 = fopen(file2, "ab")) == NULL )
3.              {
4.                              fprintf(stderr,"추가를 위한 파일을 열 수 없습니다.₩n");
5.                              exit(1);
6.              }

7.              // 첫번째 파일을 두번째 파일 끝에 추가한다.
8.              while((count = fread(buffer, sizeof(char), 1024, fp1)) > 0)
9.              {
10.                             fwrite(buffer, sizeof(char), count, fp2);
11.             }

12.             fclose(fp1);
13.             fclose(fp2);

14.             return 0;
15.   }
```

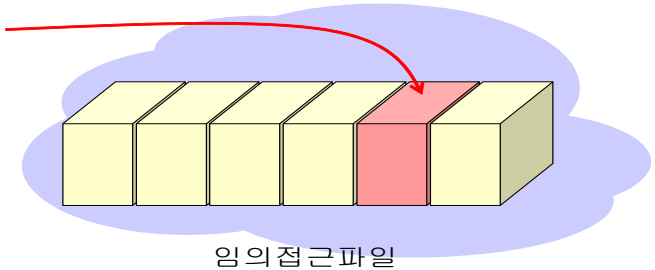
첫번째 파일 이름: a.dat
두번째 파일 이름: b.dat

파일의 임의 접근(random access)

- **순차 접근(sequential access)**: 데이터를 파일의 처음부터 순차적으로 읽거나 기록하는 방법
- **임의 접근(random access)**: 파일의 어느 위치에서든 읽기와 쓰기가 가능한 방법



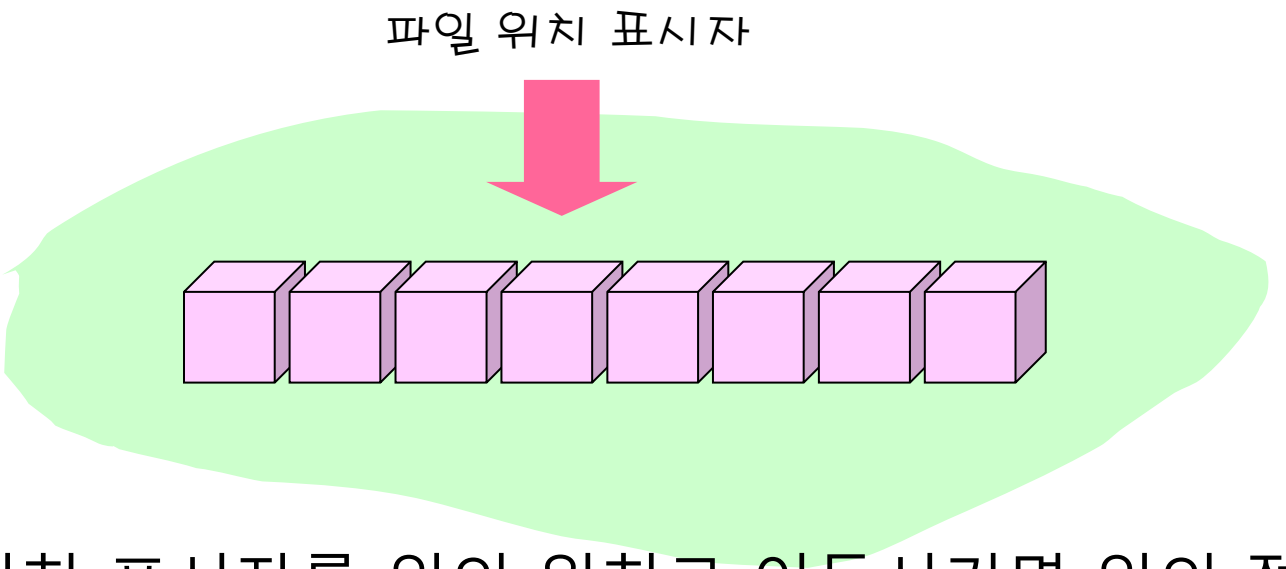
순차접근파일



임의접근파일

임의 접근 원리

- 파일 내부의 위치 표시자(location pointer): 읽기와 쓰기 동작이 현재 파일 내부의 어떤 위치에서 이루어지는 지를 나타낸다.



- 위치 표시자를 임의 위치로 이동시키면 임의 접근이 가능

임의 접근 관련 함수

```
int fseek(FILE *fp, long offset, int origin);
```

상수	값	설명
SEEK_SET	0	파일의 시작
SEEK_CUR	1	현재 위치
SEEK_END	2	파일의 끝

```
fseek(fp, 0L, SEEK_SET);                    // 파일의 처음으로 이동
fseek(fp, 0L, SEEK_END);                    // 파일의 끝으로 이동
fseek(fp, 100L, SEEK_SET);                  // 파일의 처음에서 100바이트 이동
fseek(fp, 50L, SEEK_CUR);                   // 현재 위치에서 50바이트 이동
fseek(fp, -20L, SEEK_END);                  // 파일의 끝에서 20바이트 앞으로 이동
fseek(fp, sizeof(struct element), SEEK_SET);        // 구조체만큼 앞으로 이동
```

임의 접근 관련 함수

```
void rewind(FILE *fp);
```

파일 위치 표시자를 0으로 초기화

```
long ftell(FILE *fp);
```

파일 위치 표시자의 현재 위치를 반환

```
int feof( FILE *stream);
int ferror( FILE *stream);
int remove( char const *filename);
int rename( char const *oldname , char const *newname);
int fflush(FILE *stream);
```

fseek.c

```
1.  #include <stdio.h>
2.  #include <stdlib.h>

3.  #define SIZE 1000
4.  void init_table(int table[], int size);

5.  int main(void)
6.  {
7.          int table[SIZE];
8.          int n, data;
9.          long pos;
10.         FILE *fp = NULL;
11.
12.         // 배열을 초기화한다.
13.         init_table(table, SIZE);

14.         // 이진 파일을 쓰기 모드로 연다.
15.         if( (fp = fopen("sample.dat", "wb")) == NULL )
16.         {
17.                 fprintf(stderr,"출력을 위한 파일을 열 수 없습니다.₩n");
18.                 exit(1);
19.         }

20.         // 배열을 이진 모드로 파일에 저장한다.
21.         fwrite(table, sizeof(int), SIZE, fp);
22.         fclose(fp);
```

```
1.              // 이진 파일을 읽기 모드로 연다.
2.              if( (fp = fopen("sample.dat", "rb")) == NULL )
3.              {
4.                              fprintf(stderr,"입력을 위한 파일을 열 수 없습니다.₩n");
5.                              exit(1);
6.              }
7.              // 사용자가 선택한 위치의 정수를 파일로부터 읽는다.
8.              while(1)
9.              {
10.                             printf("파일에서의 위치를 입력하십시요(0에서 %d, 종료-1): ", SIZE - 1);
11.                             scanf("%d", &n);
12.                             if( n == -1 ) break
13.                             pos = (long) n * sizeof(int);
14.                             fseek(fp, pos, SEEK_SET);
15.                             fread(&data, sizeof(int), 1, fp);
16.                             printf("%d 위치의 값은 %d입니다.₩n", n, data);
17.             }
18.             fclose(fp);
19.             return 0;
20.     }
21.     // 배열을 인덱스의 제곱으로 채운다.
22.     void init_table(int table[], int size)
23.     {
24.             int i;

25.             for(i = 0; i < size; i++)
26.                             table[i] = i * i;
27.     }
```

```
파일에서의 위치를 입력하십시요(0에서 999, 종료 -1): 3
3 위치의 값은 9입니다.
파일에서의 위치를 입력하십시요(0에서 999, 종료 -1): 9
9 위치의 값은 81입니다.
파일에서의 위치를 입력하십시요(0에서 999, 종료 -1): -1
```