CME 112- Programming Languages II Lecture 7: File Operations (Part-1)

Assist. Prof.Dr. Ümit ATİLA

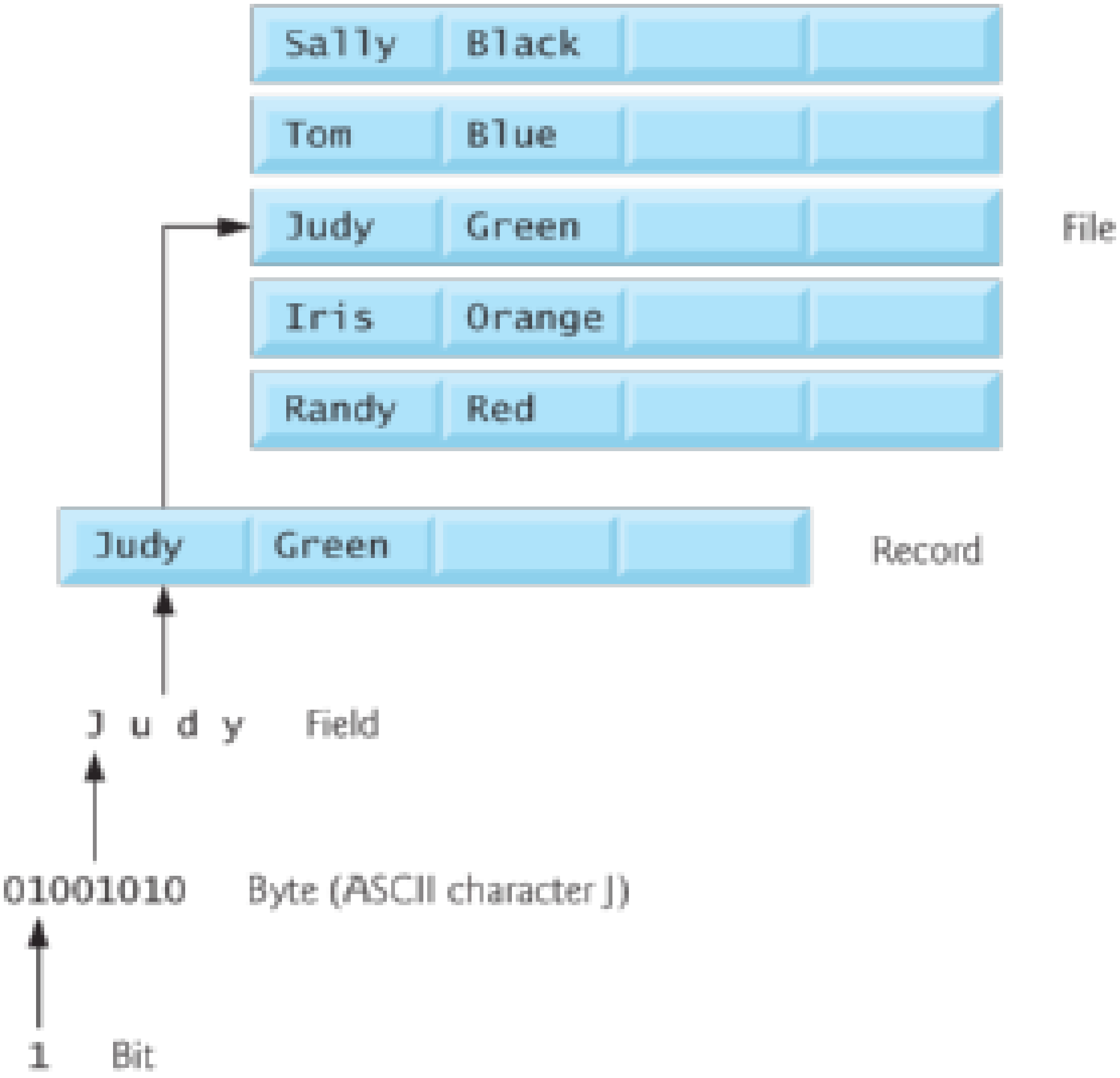
INTRODUCTION

- Storage of data in variables and arrays is temporary—such data is lost when a program terminates.
- Files are used for permanent retention of data.
- Computers store files on secondary storage devices, especially disk storage devices.

Data Hierarchy

- All data items processed by a computer are reduced to combinations of zeros and ones.
  - **Bit**: The smallest data item in a computer can assume the value 0 or the value 1.
  - **Byte**: Digits, letters, and special symbols are referred to as characters. Since computers can process only 1s and 0s, every character in a computer’s character set is represented as a pattern of 1s and 0s (called a byte). 1 byte = 8 bits
  - **Field**: Composed of characters. Field is a group of character that conveys meaning.
    - Ex: person name
  - **Record**: A group of related fields.
    - Represented by a struct or a class
    - Ex: In a payroll system, a record for a particular employee that contained his/her identification number, name, address, etc.
  - **File**: A group of related records.
    - Ex: Payroll file.
  - **Database**: A group of related files.

Data Hierarchy



Data Hierarchy

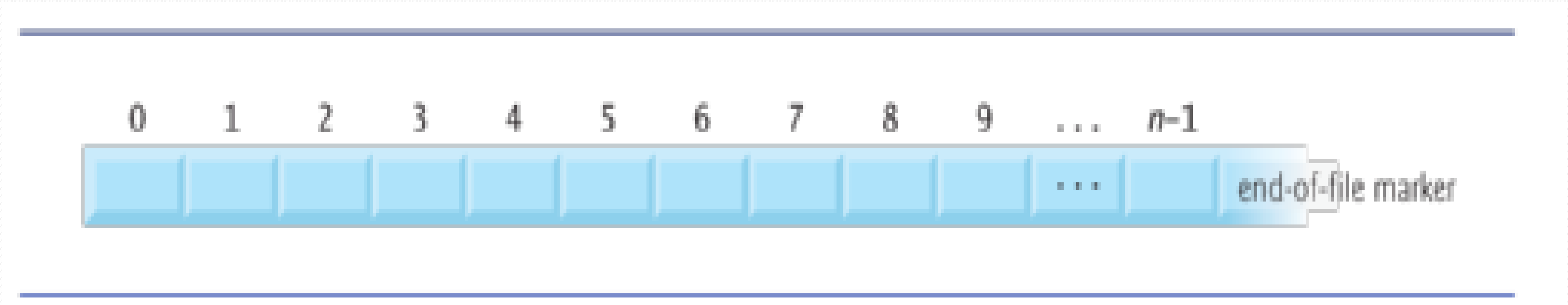
- **Record Key**: To facilitate the retrieval of specific records from a file, at least one field in each record is chosen as a record key.
  - Ex: In a school management system student id number could be chosen as a record key.
- Sequential File: Most popular way of organizing records in a file
  - Records typically sorted by record key

Files and Streams

- C views each file as a sequence of bytes
  - File ends with the **end-of-file** marker
    - Or, file ends at a specified byte
- Stream created when a file is opened. Streams provide communication channels between files and programs.
  - Provide communication channel between files and programs
  - Opening a file returns a pointer to a **FILE** structure
    - Example file pointers:
    - **stdin** - standard input (enables reading data from keyboard)
    - **stdout** - standard output (enables printing data on screen)
    - **stderr** - standard error (screen)

Files and Streams

- **FILE structure** (opening a file returns a pointer to FILE structure) that contain information used to process file
  - **File descriptor**
    - Index into operating system array called the open file table
  - **File Control Block (FCB)**
    - Found in every array element, system uses it to administer the file
- Standard input, standard output and standard error are manipulated using file pointers stdin, stdout and stderr



Files and Streams

- Read/Write functions in standard library
  - **fgetc**
    - Reads one character from a file
    - Takes a FILE pointer as an argument
    - fgetc(stdin) equivalent to getchar()
  - **fputc**
    - Writes one character to a file
    - Takes a FILE pointer and a character to write as an argument
    - fputc('a', stdout) equivalent to putchar('a')
  - **fgets**
    - Reads a line from a file
  - **fputs**
    - Writes a line to a file
  - **fscanf / fprintf**
    - File processing equivalents of scanf and printf

CREATING A SEQUENTIAL ACCESS FILE

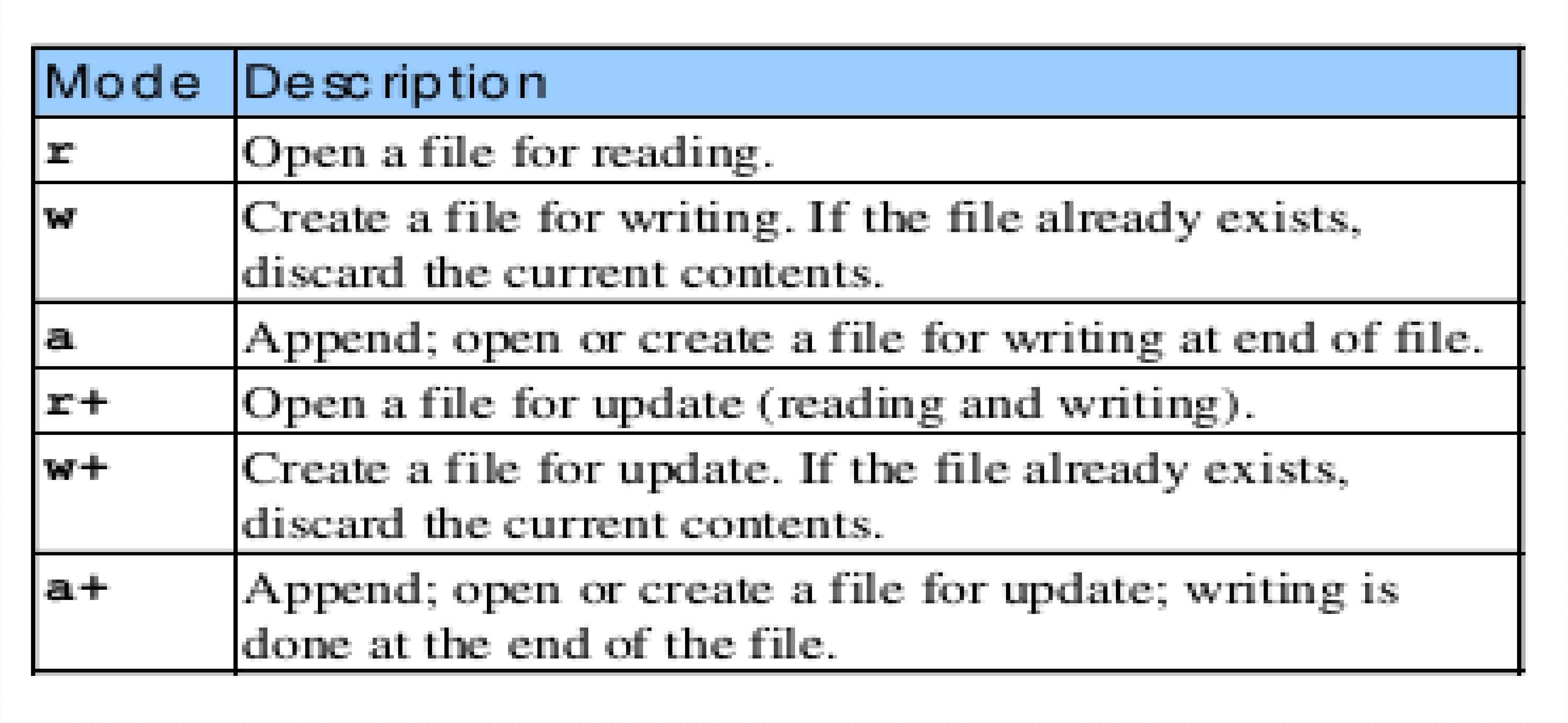
- C imposes no file structure
  - No notion of records in a file
  - Programmer must provide file structure
- Creating a File
  - **FILE *myPtr;**
    - Creates a **FILE** pointer called **myPtr**
  - **myPtr = fopen("myFile.dat", openmode);**
    - Function **fopen** returns a **FILE** pointer to file specified
    - Takes two arguments – file to open and file open mode
    - If open fails, NULL returned
  - **fprintf**
    - Used to print to a file
    - Like printf, except first argument is a FILE pointer (pointer to the file you want to print in)

CREATING A SEQUENTIAL ACCESS FILE

- **feof(FILE pointer)**
  - Returns true if end-of-file indicator (no more data to process) is set for the specified file
- **fclose(FILE pointer)**
  - Closes specified file
  - Performed automatically when program ends
- Details
  - Programs may process no files, one file, or many files
  - Each file must have a unique name and should have its own pointer

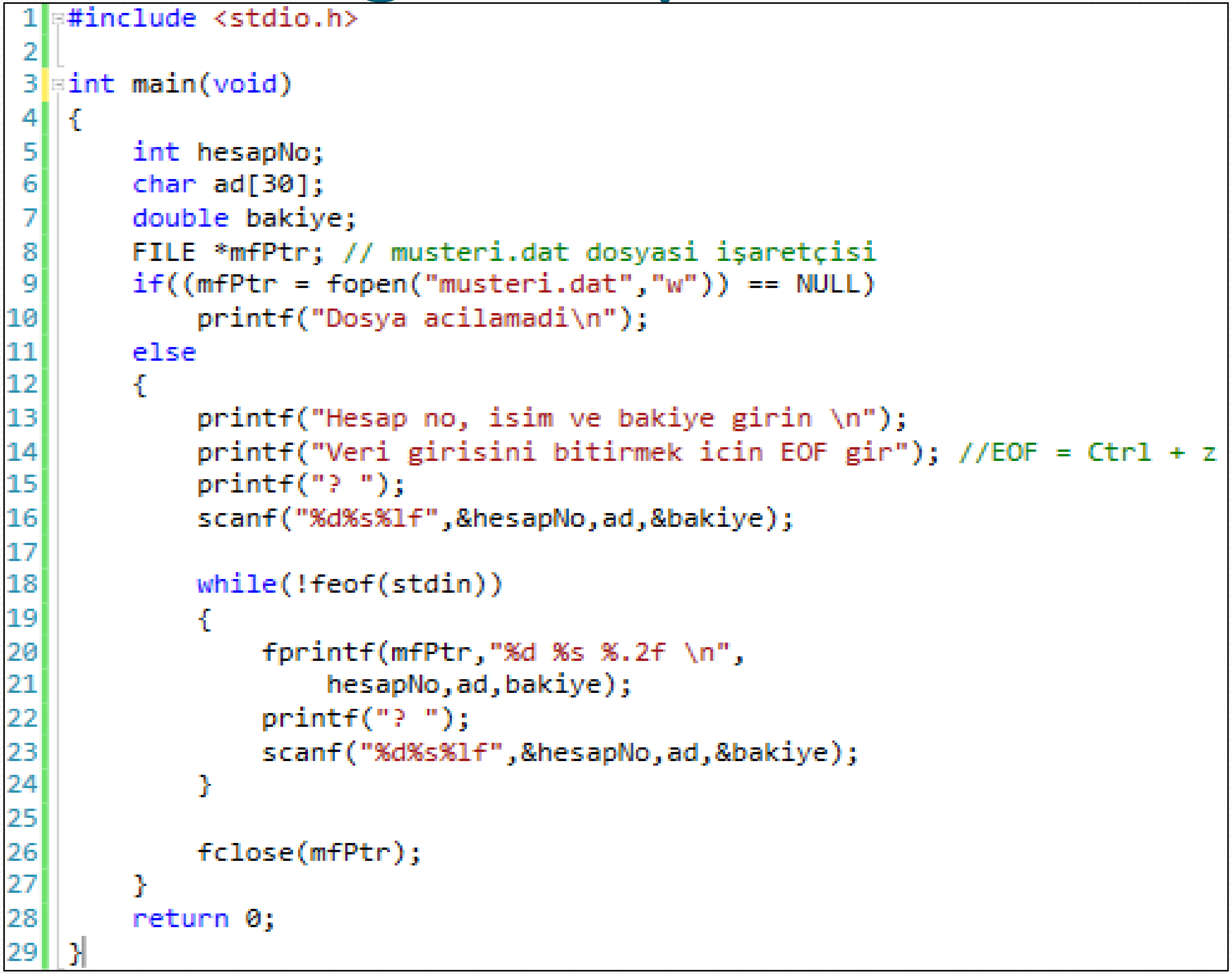
CREATING A SEQUENTIAL ACCESS FILE

- Table of file open modes



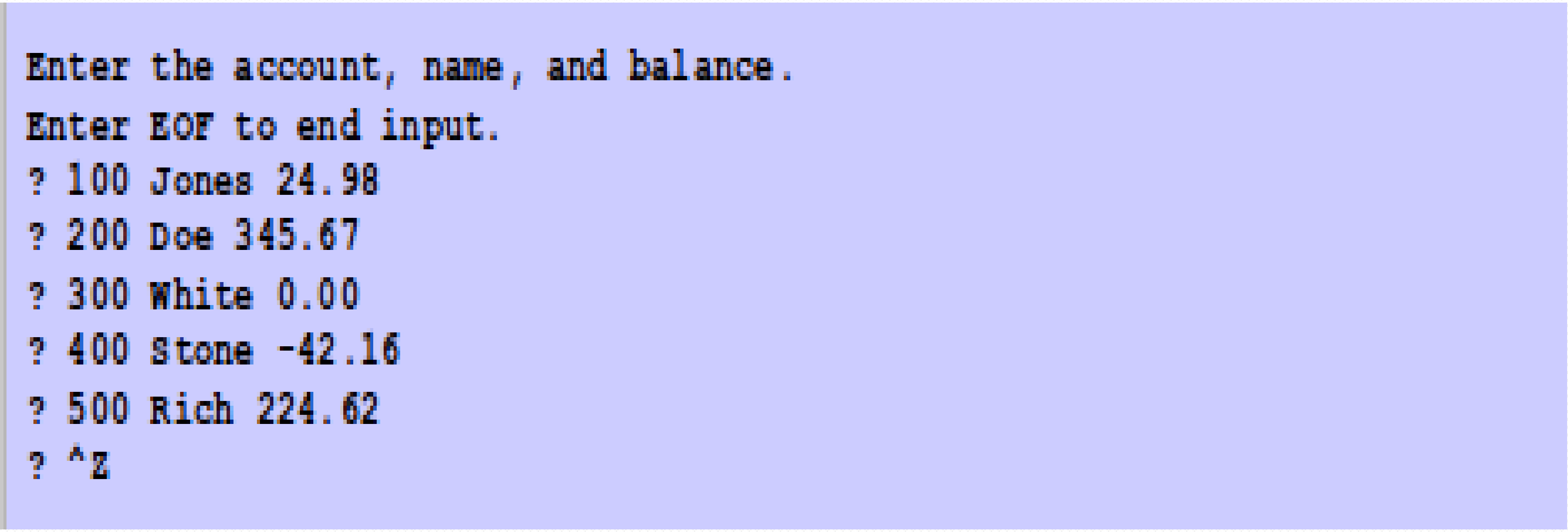
Mode	Description
`r`	Open a file for reading.
`w`	Create a file for writing. If the file already exists, discard the current contents.
`a`	Append; open or create a file for writing at end of file.
`r+`	Open a file for update (reading and writing).
`w+`	Create a file for update. If the file already exists, discard the current contents.
`a+`	Append; open or create a file for update; writing is done at the end of the file.

Creating a Sequential File



```
#include <stdio.h>

int main(void)
{
    int hesapNo;
    char ad[30];
    double bakiye;
    FILE *mfPtr; // musteri.dat dosyasi işaretçisi
    if((mfPtr = fopen("musteri.dat","w")) == NULL)
        printf("Dosya acilamadi\n");
    else
    {
        printf("Hesap no, isim ve bakiye girin \n");
        printf("Veri girisini bitirmek icin EOF gir"); //EOF = Ctrl + z
        printf("? ");
        scanf("%d%s%lf",&hesapNo,ad,&bakiye);

        while(!feof(stdin))
        {
            fprintf(mfPtr,"%d %s %.2f \n",
                hesapNo,ad,bakiye);
            printf("? ");
            scanf("%d%s%lf",&hesapNo,ad,&bakiye);
        }

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

Creating a Sequential File



```
Enter the account, name, and balance.
Enter EOF to end input.
? 100 Jones 24.98
? 200 Doe 345.67
? 300 White 0.00
? 400 Stone -42.16
? 500 Rich 224.62
? ^Z
```

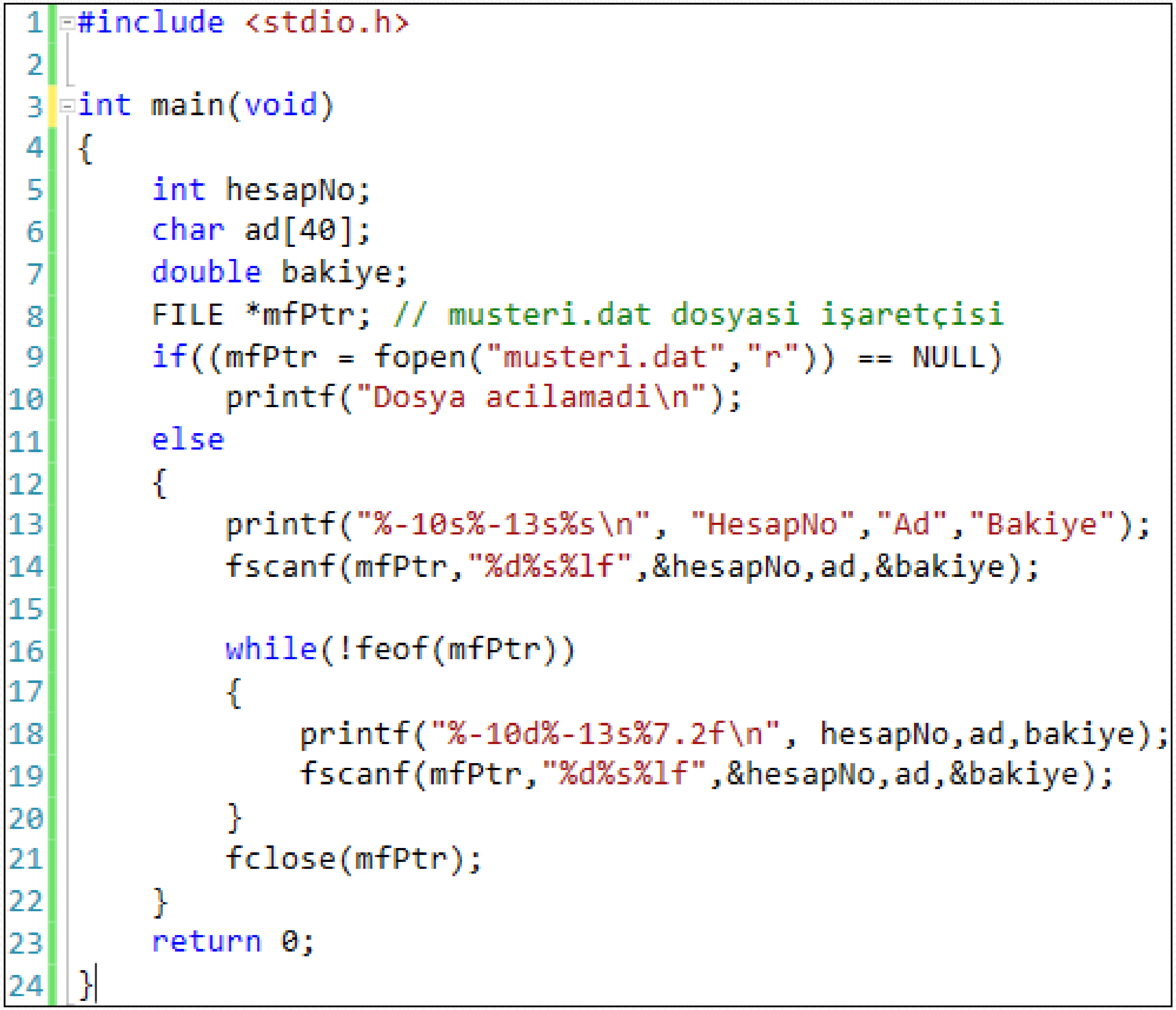
READING DATA FROM A SEQUENTIAL ACCESS FILE

- Reading a sequential access file
  - Create a **FILE** pointer, link it to the file to read

    myPtr = fopen("myFile.dat", "r");
  - Use **fscanf** to read from the file
    - Like scanf, except first argument is a FILE pointer

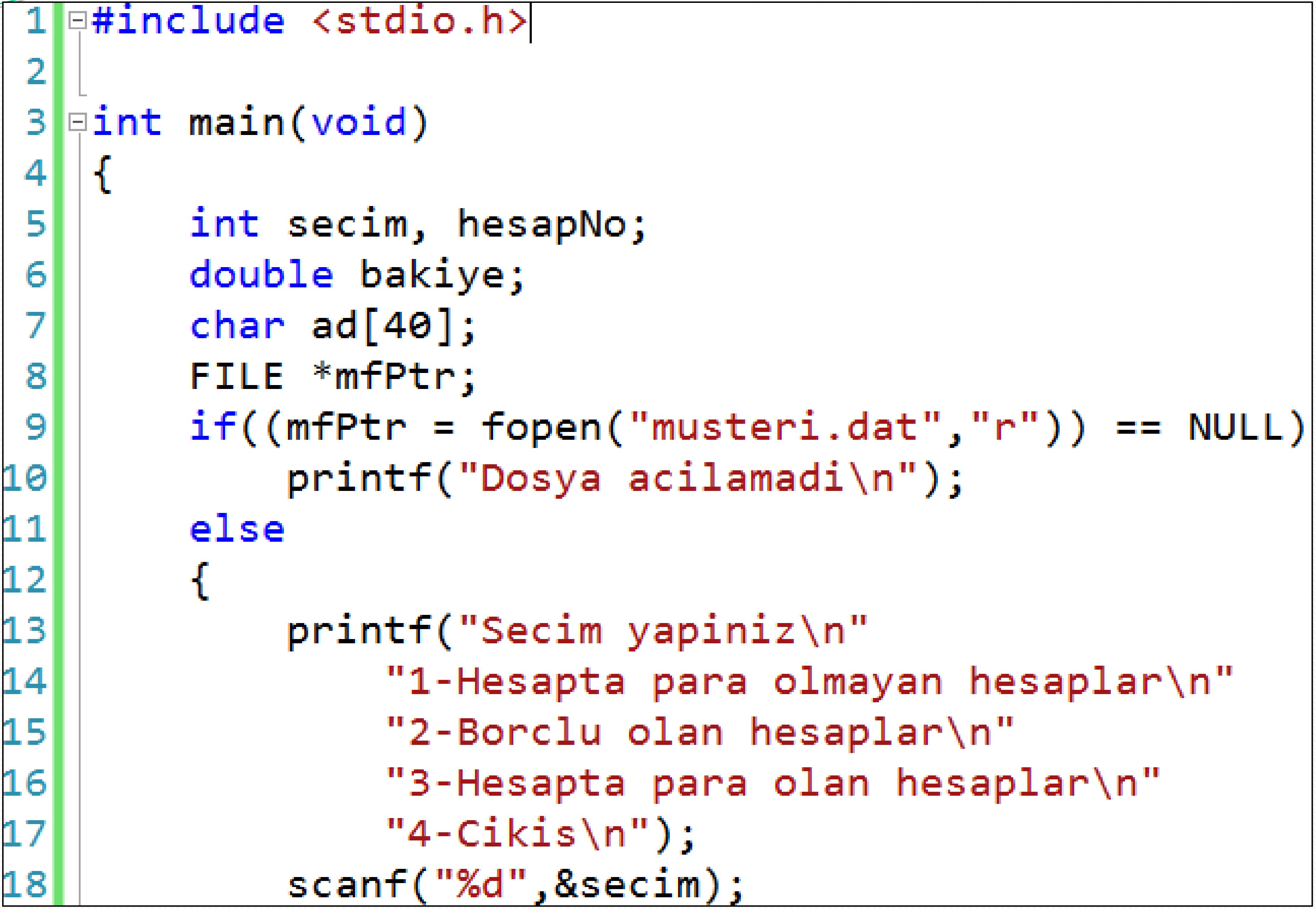
    fscanf(myPtr, "%d%s%f", &myInt, &myString, &myFloat);
  - Data read from beginning to end
  - File position pointer
    - Indicates number of next byte to be read / written
    - Not really a pointer, but an integer value (specifies byte location)
    - Also called byte offset
  - **rewind(myPtr)**
    - Repositions file position pointer to beginning of file (byte 0)

Reading & Printing a Sequential File



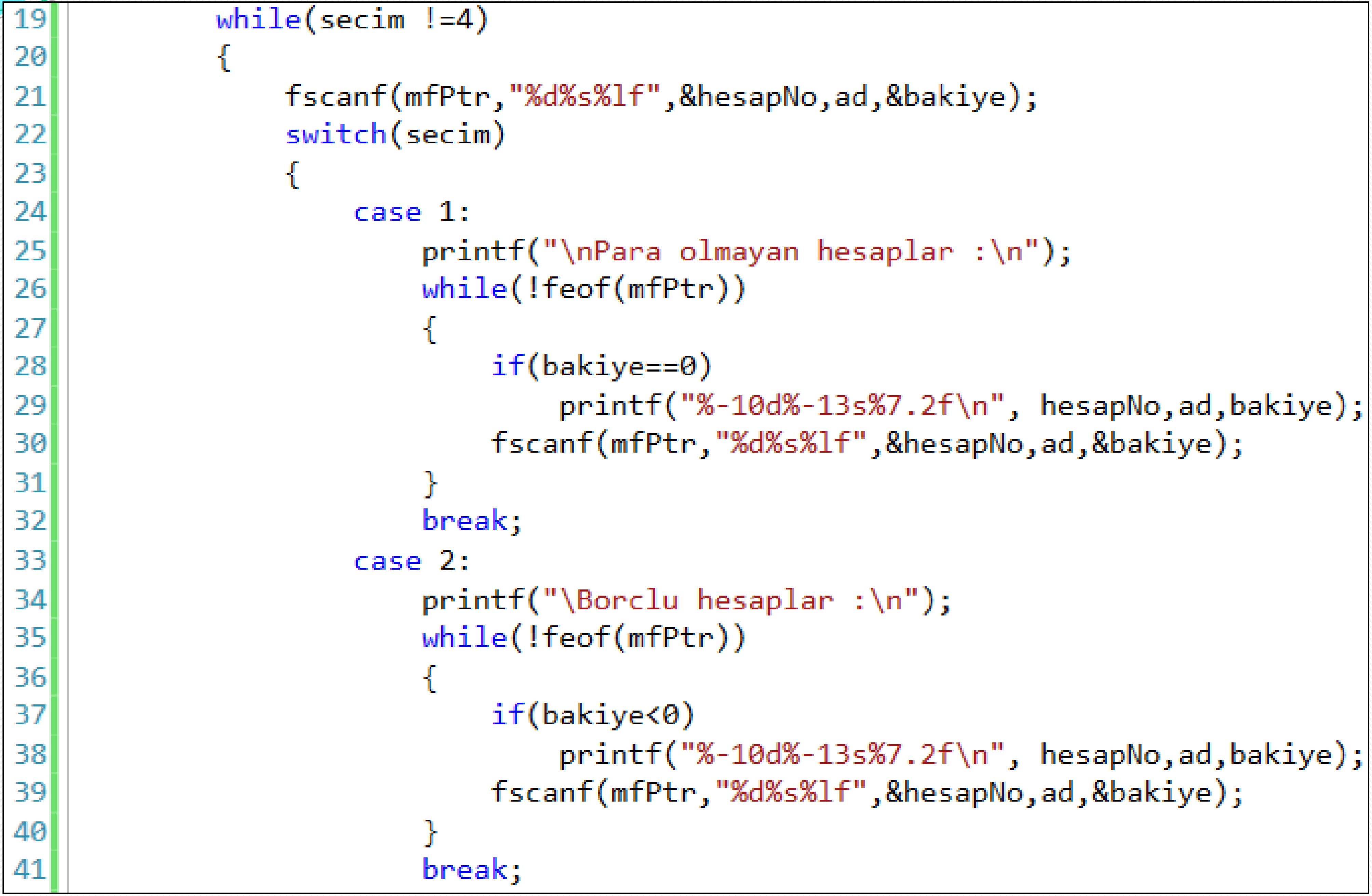
```
#include <stdio.h>

int main(void)
{
    int hesapNo;
    char ad[40];
    double bakiye;
    FILE *mfPtr; // musteri.dat dosyasi işaretçisi
    if((mfPtr = fopen("musteri.dat","r")) == NULL)
        printf("Dosya acilamadi\n");
    else
    {
        printf("%-10s%-13s%s\n", "HesapNo","Ad","Bakiye");
        fscanf(mfPtr,"%d%s%lf",&hesapNo,ad,&bakiye);

        while(!feof(mfPtr))
        {
            printf("%-10d%-13s%7.2f\n", hesapNo,ad,bakiye);
            fscanf(mfPtr,"%d%s%lf",&hesapNo,ad,&bakiye);
        }
        fclose(mfPtr);
    }
    return 0;
}
```

Application-1



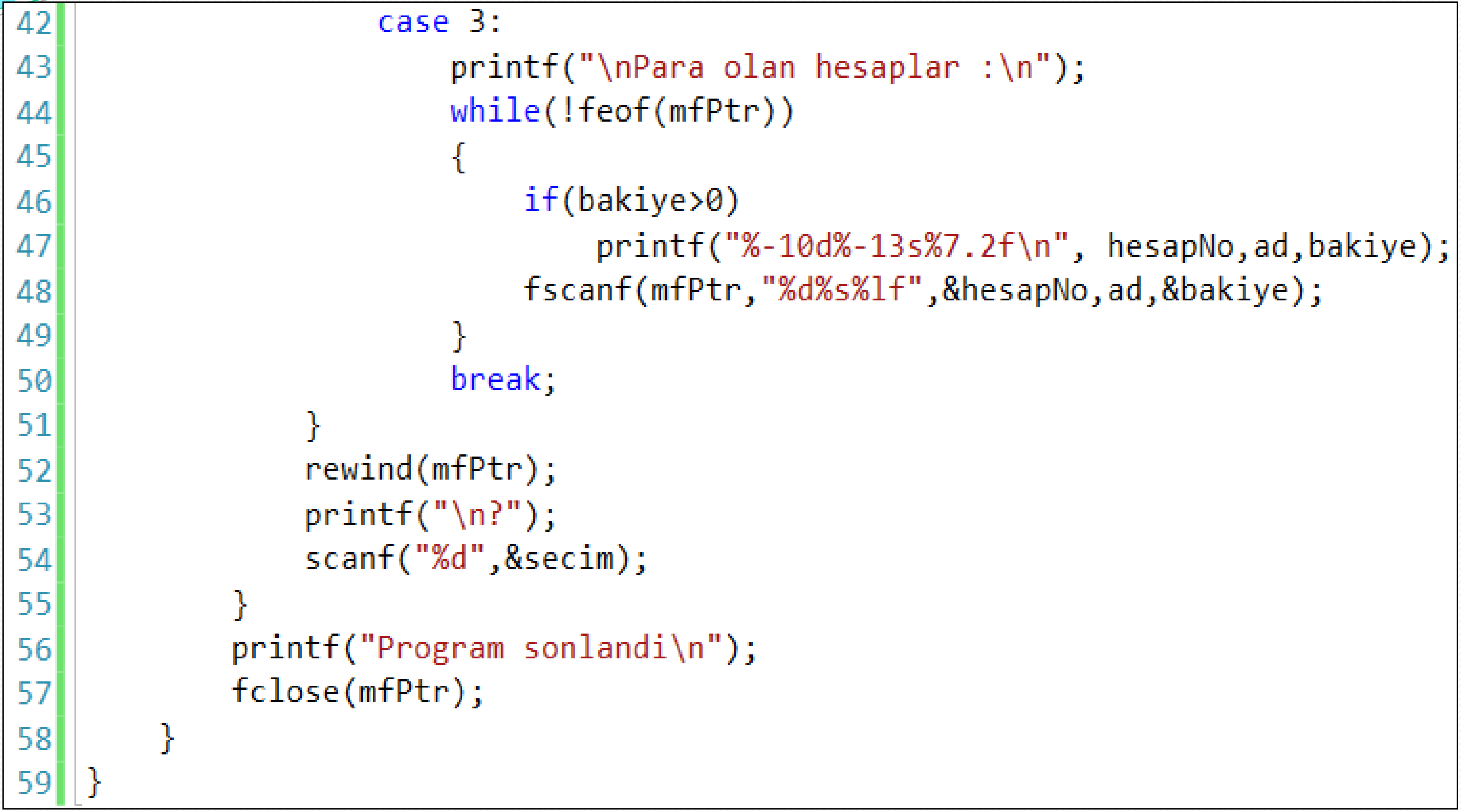
```c
#include <stdio.h>

int main(void)
{
    int secim, hesapNo;
    double bakiye;
    char ad[40];
    FILE *mfPtr;
    if((mfPtr = fopen("musteri.dat","r")) == NULL)
        printf("Dosya acilamadi\n");
    else
    {
        printf("Secim yapiniz\n"
            "1-Hesapta para olmayan hesaplar\n"
            "2-Borclu olan hesaplar\n"
            "3-Hesapta para olan hesaplar\n"
            "4-Cikis\n");
        scanf("%d",&secim);
```

Application-1



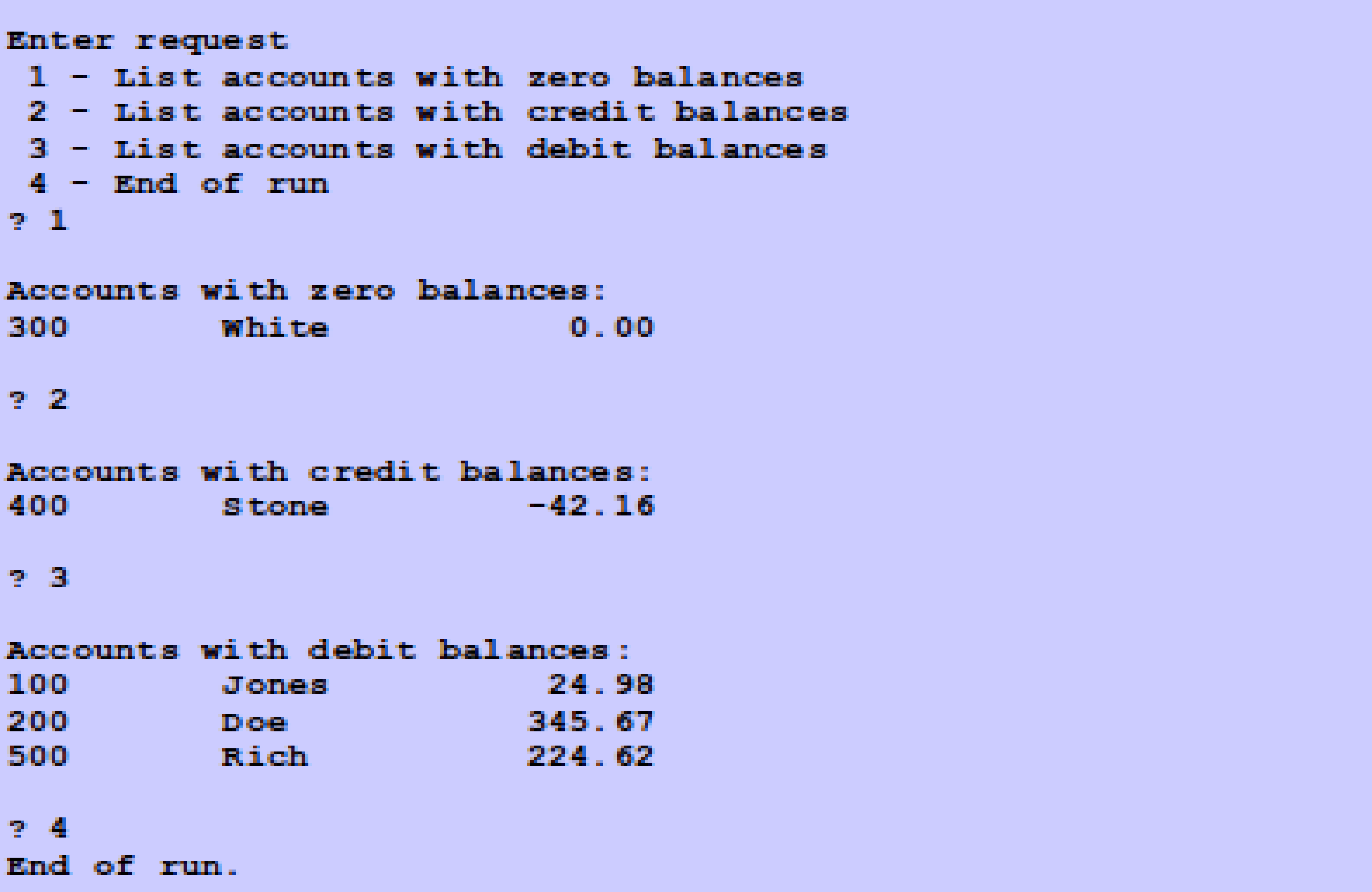
```
        while(secim !=4)
        {
            fscanf(mfPtr,"%d%s%lf",&hesapNo,ad,&bakiye);
            switch(secim)
            {
                case 1:
                    printf("\nPara olmayan hesaplar :\n");
                    while(!feof(mfPtr))
                    {
                        if(bakiye==0)
                            printf("%-10d%-13s%7.2f\n", hesapNo,ad,bakiye);
                        fscanf(mfPtr,"%d%s%lf",&hesapNo,ad,&bakiye);
                    }
                    break;
                case 2:
                    printf("\Borclu hesaplar :\n");
                    while(!feof(mfPtr))
                    {
                        if(bakiye<0)
                            printf("%-10d%-13s%7.2f\n", hesapNo,ad,bakiye);
                        fscanf(mfPtr,"%d%s%lf",&hesapNo,ad,&bakiye);
                    }
                    break;
```

Application-1



```c
                case 3:
                    printf("\nPara olan hesaplar :\n");
                    while(!feof(mfPtr))
                    {
                        if(bakiye>0)
                            printf("%-10d%-13s%7.2f\n", hesapNo,ad,bakiye);
                        fscanf(mfPtr,"%d%s%lf",&hesapNo,ad,&bakiye);
                    }
                    break;
            }
            rewind(mfPtr);
            printf("\n?");
            scanf("%d",&secim);
        }
        printf("Program sonlandi\n");
        fclose(mfPtr);
    }
}
```

Application-1



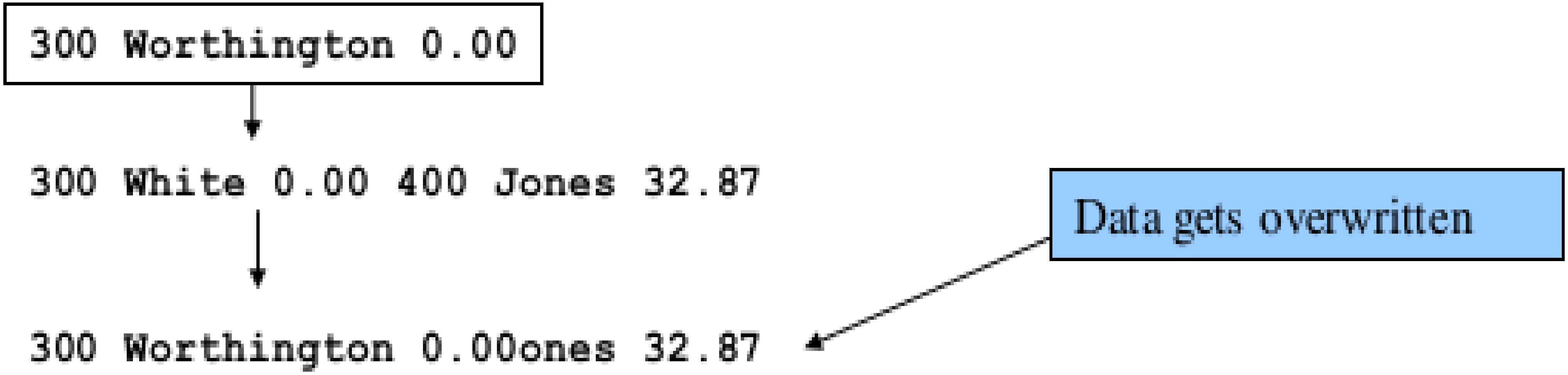
```
Enter request
 1 - List accounts with zero balances
 2 - List accounts with credit balances
 3 - List accounts with debit balances
 4 - End of run
? 1

Accounts with zero balances:
300       White             0.00

? 2

Accounts with credit balances:
400       Stone           -42.16

? 3

Accounts with debit balances:
100       Jones            24.98
200       Doe             345.67
500       Rich            224.62

? 4
End of run.
```

READING DATA FROM A SEQUENTIAL ACCESS FILE

- Sequential access file
  - Cannot be modified without the risk of destroying other data
  - Fields can vary in size
    - Different representation in files and screen than internal representation
    - 1, 34, -890 are all ints, but have different sizes on disk

300 White 0.00 400 Jones 32.87 (old data in file)

- If we want to change White's name to Worthington



```
300 Worthington 0.00

300 White 0.00 400 Jones 32.87

300 Worthington 0.00ones 32.87
```

Data gets overwritten