# CSE102
# Computer Programming with C

2016-2017 Spring Semester

## Files

© 2015-2016 Yakup Genç & Shahid Alam

Largely adapted from J.R. Hanly, E.B. Koffman, F.E. Sevilgen, and others…

## File Processing

- Files: used for permanent storage of information

## File Processing

- Files: used for permanent storage of information

- Two types of files:
  - Text files
  - Binary files

## File Processing

- Files: used for permanent storage of information

- Two types of files:
  - Text files
  - Binary files

But all the files stored in a computer are binaries???

## Text Files

- Text file: collection of characters
  - Can be considered as stream of characters
    - Input stream
      - EX: keyboard : stdin
    - Output stream
      - EX: Screen : stdout
      - stderr
  - Can be created by using editors
    - Readable by human
  - Special characters
    - New line character (Windows: CRLF – UNIX/Mac: LF)
    - End of file character
      - EOF is returned when read
    - Other escape sequences

# Escape sequences

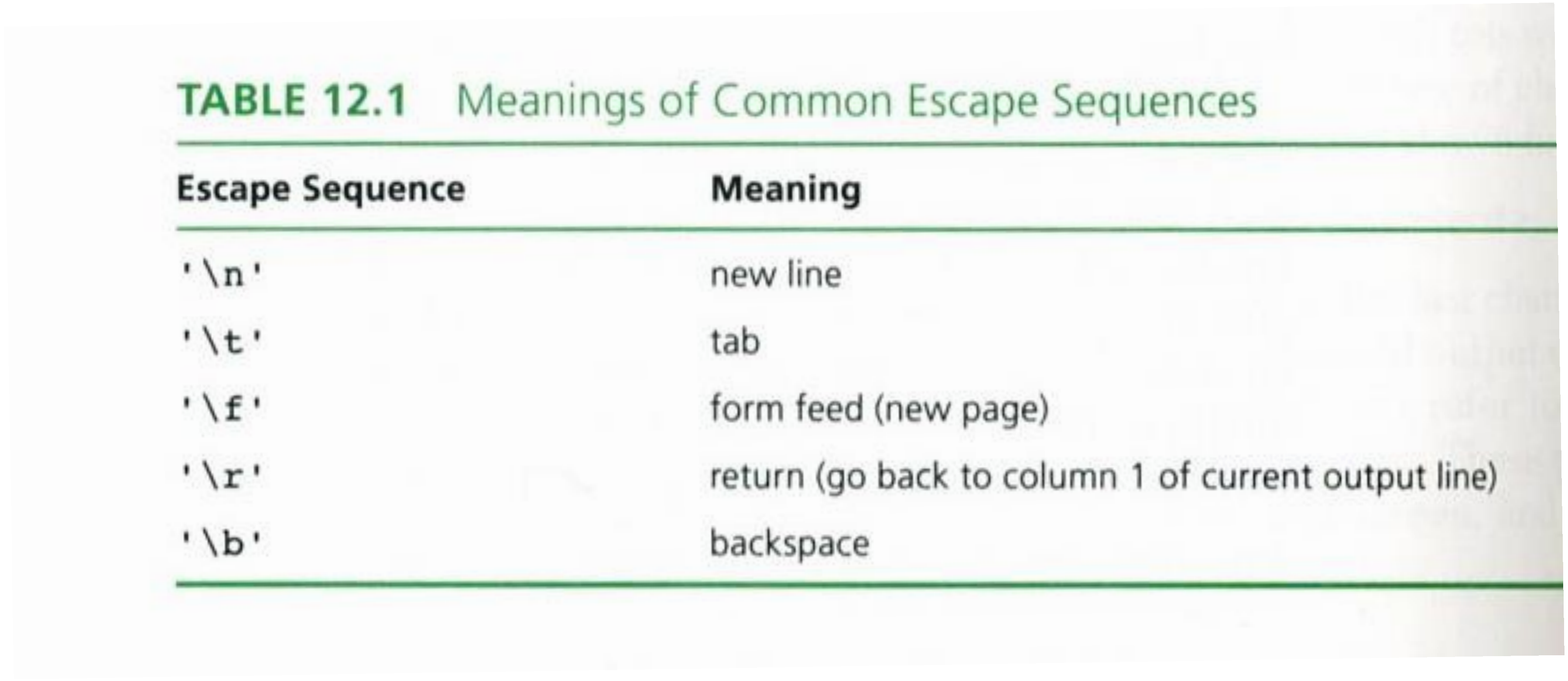


TABLE 12.1 Meanings of Common Escape Sequences

| Escape Sequence | Meaning |
|---|---|
| `'\n'` | new line |
| `'\t'` | tab |
| `'\f'` | form feed (new page) |
| `'\r'` | return (go back to column 1 of current output line) |
| `'\b'` | backspace |

# Formatting output with printf

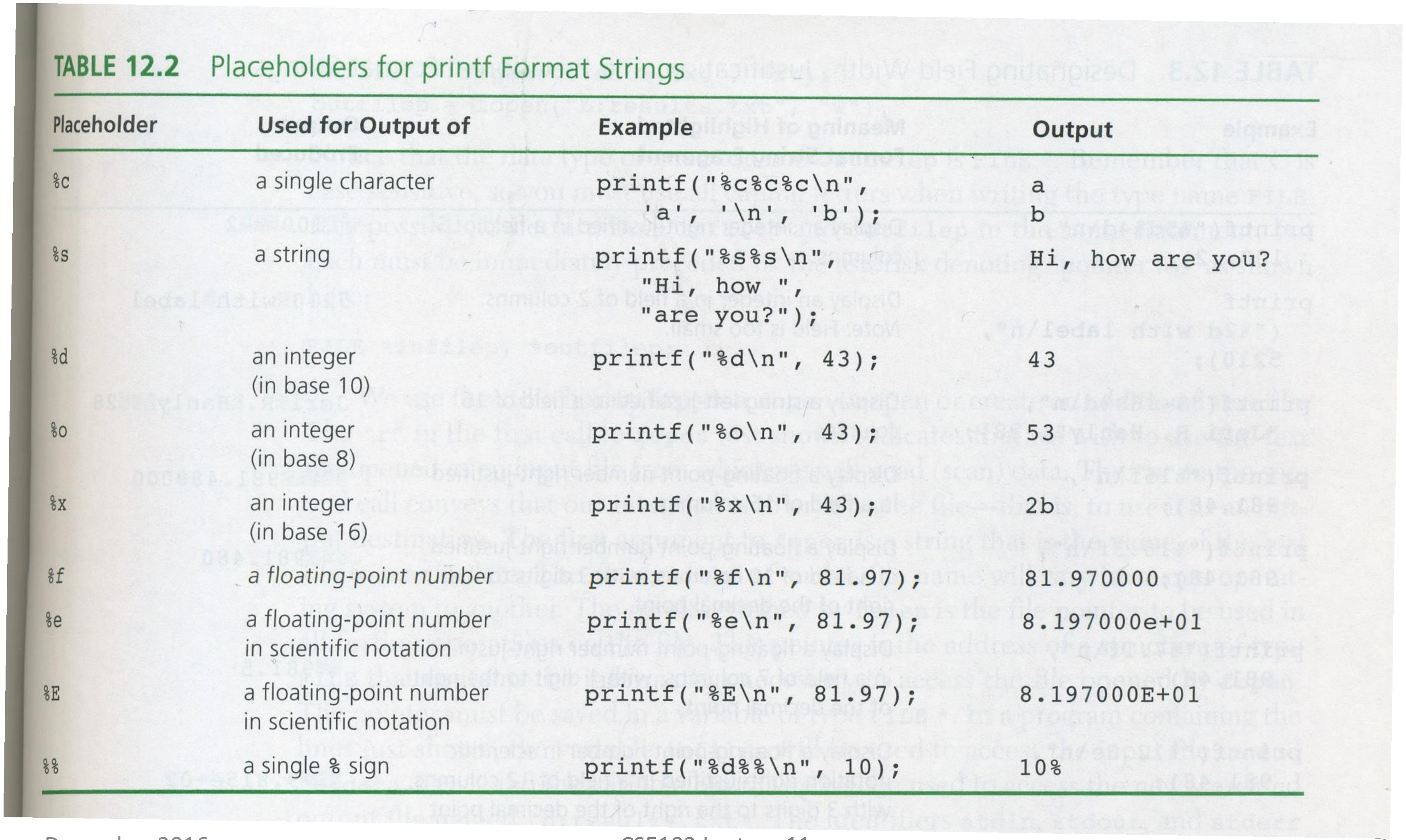


**TABLE 12.2** Placeholders for printf Format Strings

| Placeholder | Used for Output of | Example | Output |
|---|---|---|---|
| %c | a single character | `printf("%c%c%c\n", 'a', '\n', 'b');` | a<br>b |
| %s | a string | `printf("%s%s\n", "Hi, how ", "are you?");` | Hi, how are you? |
| %d | an integer (in base 10) | `printf("%d\n", 43);` | 43 |
| %o | an integer (in base 8) | `printf("%o\n", 43);` | 53 |
| %x | an integer (in base 16) | `printf("%x\n", 43);` | 2b |
| %f | a floating-point number | `printf("%f\n", 81.97);` | 81.970000 |
| %e | a floating-point number in scientific notation | `printf("%e\n", 81.97);` | 8.197000e+01 |
| %E | a floating-point number in scientific notation | `printf("%E\n", 81.97);` | 8.197000E+01 |
| %% | a single % sign | `printf("%d%%\n", 10);` | 10% |

# Formatting output with printf

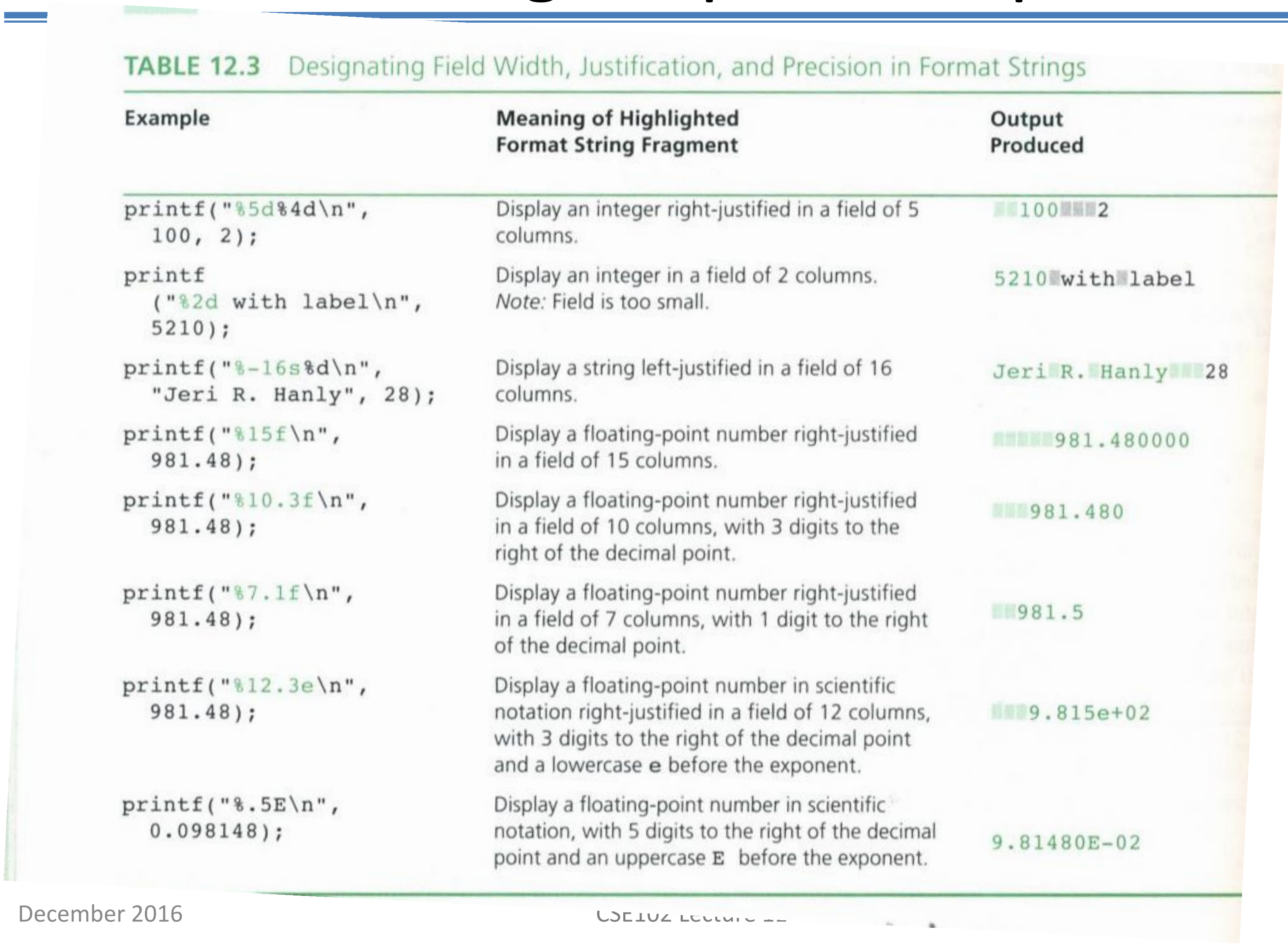


TABLE 12.3 Designating Field Width, Justification, and Precision in Format Strings

| Example | Meaning of Highlighted Format String Fragment | Output Produced |
|---|---|---|
| `printf("%5d%4d\n", 100, 2);` | Display an integer right-justified in a field of 5 columns. | `▨▨100▨▨▨2` |
| `printf ("%2d with label\n", 5210);` | Display an integer in a field of 2 columns. *Note:* Field is too small. | `5210▨with▨label` |
| `printf("%-16s%d\n", "Jeri R. Hanly", 28);` | Display a string left-justified in a field of 16 columns. | `Jeri▨R.▨Hanly▨▨▨28` |
| `printf("%15f\n", 981.48);` | Display a floating-point number right-justified in a field of 15 columns. | `▨▨▨▨▨981.480000` |
| `printf("%10.3f\n", 981.48);` | Display a floating-point number right-justified in a field of 10 columns, with 3 digits to the right of the decimal point. | `▨▨▨981.480` |
| `printf("%7.1f\n", 981.48);` | Display a floating-point number right-justified in a field of 7 columns, with 1 digit to the right of the decimal point. | `▨▨981.5` |
| `printf("%12.3e\n", 981.48);` | Display a floating-point number in scientific notation right-justified in a field of 12 columns, with 3 digits to the right of the decimal point and a lowercase **e** before the exponent. | `▨▨▨9.815e+02` |
| `printf("%.5E\n", 0.098148);` | Display a floating-point number in scientific notation, with 5 digits to the right of the decimal point and an uppercase `E` before the exponent. | `9.81480E-02` |

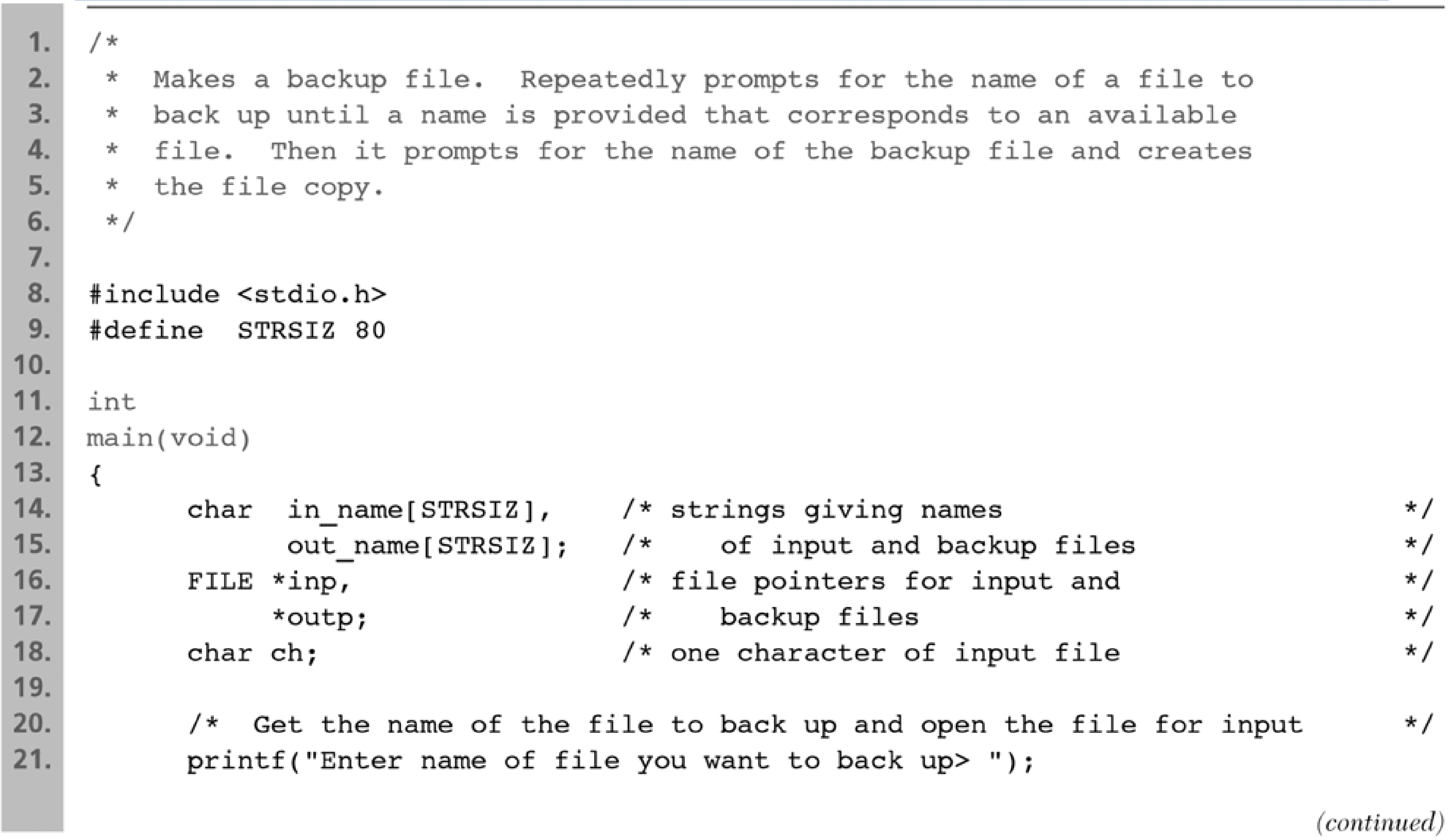
## File Pointer

- Allows to access a file

```
FILE *fileptr;
fileptr = fopen(“filename”, “access mode”);
if (fileptr == NULL)
    printf(“File open error”);
else
    …. process file ….
fclose(fileptr);
```

- Processing with getc, putc, fscanf and fprintf
  - What if stdin or stdout is used as FILE *

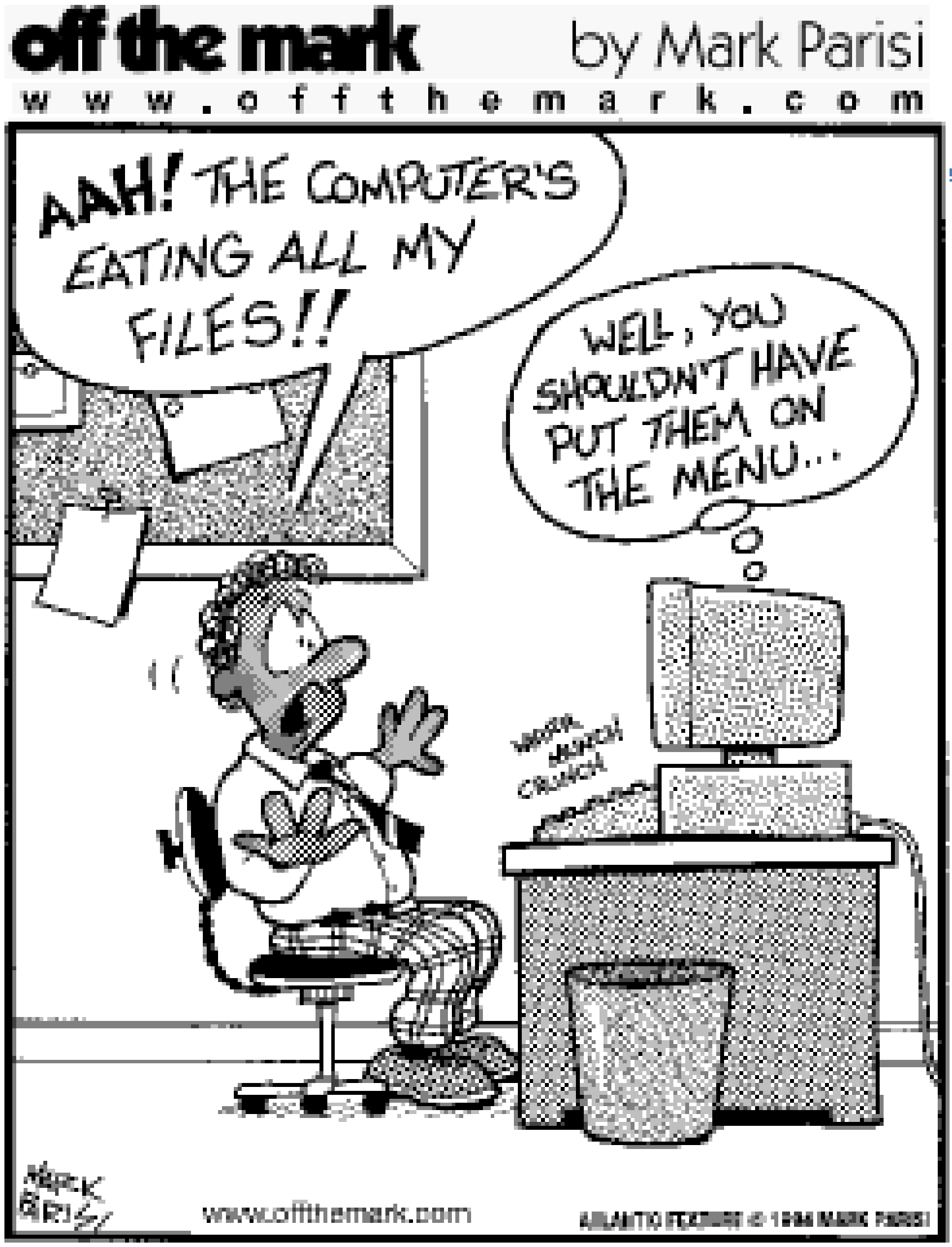
# Copying a Text File



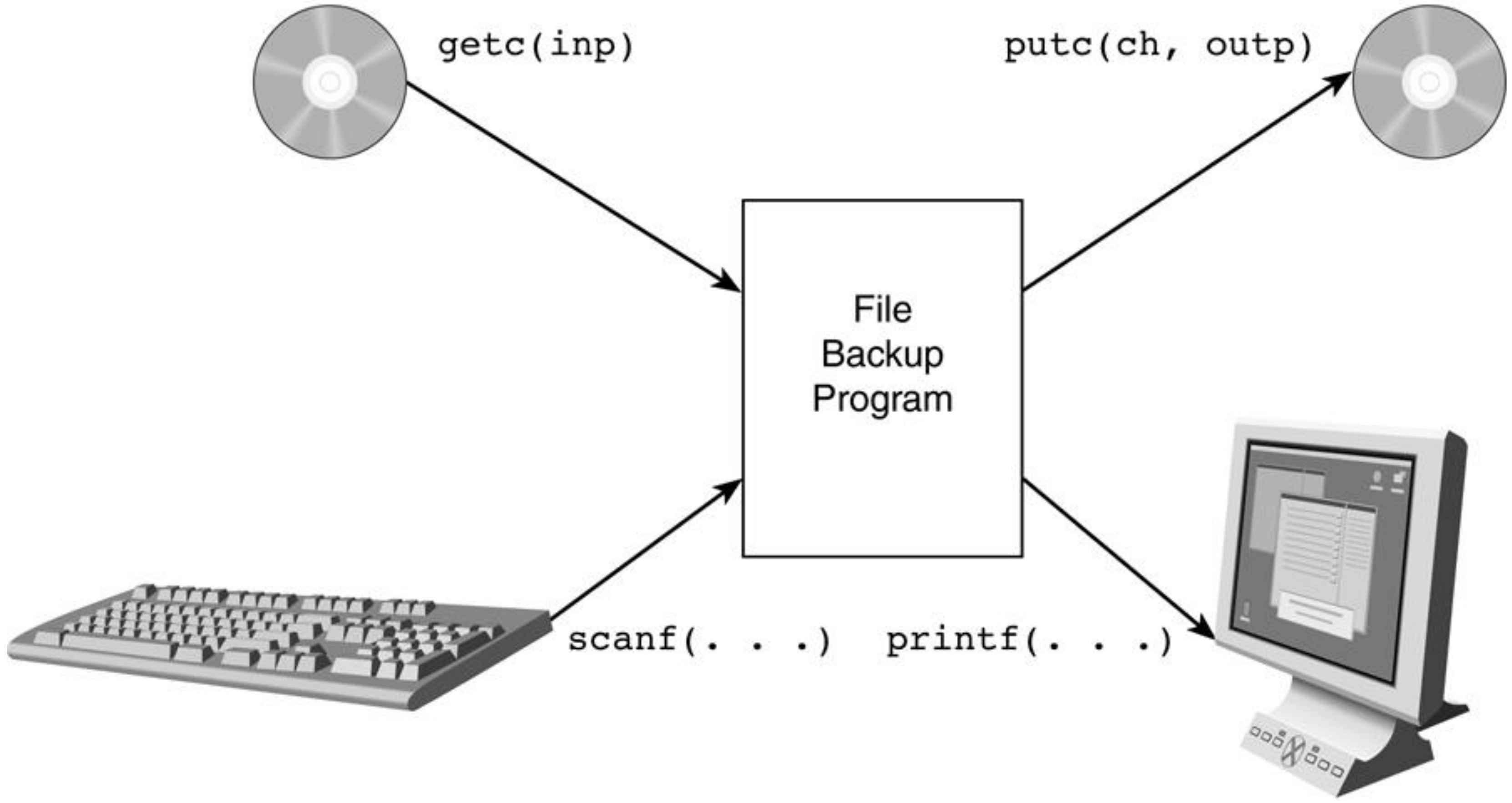
```c
/*
 *  Makes a backup file.  Repeatedly prompts for the name of a file to
 *  back up until a name is provided that corresponds to an available
 *  file.  Then it prompts for the name of the backup file and creates
 *  the file copy.
 */

#include <stdio.h>
#define  STRSIZ 80

int
main(void)
{
      char  in_name[STRSIZ],    /* strings giving names                              */
            out_name[STRSIZ];   /*    of input and backup files                      */
      FILE *inp,                /* file pointers for input and                       */
           *outp;               /*    backup files                                   */
      char ch;                  /* one character of input file                       */

      /*  Get the name of the file to back up and open the file for input       */
      printf("Enter name of file you want to back up> ");
```

*(continued)*

```
      for (scanf("%s", in_name);
           (inp = fopen(in_name, "r")) == NULL;
           scanf("%s", in_name)) {
         printf("Cannot open %s for input\n", in_name);
         printf("Re-enter file name> ");
      }

      /*  Get name to use for backup file and open file for output                 */
      printf("Enter name for backup copy> ");
      for (scanf("%s", out_name);
           (outp = fopen(out_name, "w")) == NULL;
           scanf("%s", out_name)) {
         printf("Cannot open %s for output\n", out_name);
         printf("Re-enter file name> ");
      }

      /*  Make backup copy one character at a time                                 */
      for  (ch = getc(inp);  ch != EOF;  ch = getc(inp))
          putc(ch, outp);

      /*  Close files and notify user of backup completion                         */
      fclose(inp);
      fclose(outp);
      printf("Copied %s to %s.\n", in_name, out_name);

      return(0);
}
```


off the mark
by Mark Parisi
www.offthemark.com
AAH! THE COMPUTER'S EATING ALL MY FILES!!
WELL, YOU SHOULDN'T HAVE PUT THEM ON THE MENU...
www.offthemark.com

## Input and Output Streams



# Binary Files

- Binary Files stores the data in their internal representation
  - Note that Text files stores the data as character sequence
    - requires conversion between data types and stream of characters
  - No conversion in binary files
    - Higher performance
    - Less storage
    - Higher precision for doubles
  - System dependent
    - Not portable
  - Not human readable

# Binary Files

```
FILE *fileptr;
fileptr = fopen(“filename”, “access mode”);
if (fileptr == NULL)
    printf(“File open error”);
else
    …. process file ….
fclose(fileptr);
```

- Access mode is “rb” or “wb”
- Processing with fwrite or fread
  - Ex: creating a binary file of integer

# Creating a Binary File of Integers

```c
FILE *binaryp;
int   i;

binaryp = fopen("nums.bin", "wb");

for  (i = 2;   i <= 500;   i += 2)
    fwrite(&i, sizeof (int), 1, binaryp);

fclose(binaryp);
```

## fwrite

```
int fwrite(buffer,
           size_of_each_component,
           num_of_components,
           fileptr)



int a[20];
num = fwrite(a,
             sizeof(int),
             20,
             fptr);
```

## fread

```
int fread (buffer,
           size_of_each_component,
           num_of_components,
           fileptr)


int a[20];
num = fread (a,
             sizeof(int),
             20,
             fptr);
```

## Text file vs Binary file

- Assume following declarations

```
#define STRSIZ 10
#define MAX 40

typedef struct {
            char        name[20];
            double      diameter;
            int         moons;
            double      orbit_time,
                        rotation_time;
} planet_t;

double nums[MAX], data;
planet_t a_planet;
int i, n, status;
FILE *plan_bin_inp, *plan_bin_outp, *plan_txt_inp, *plan_txt_outp;
FILE *doub_bin_inp, *doub_bin_outp, *doub_txt_inp, *doub_txt_outp;
```

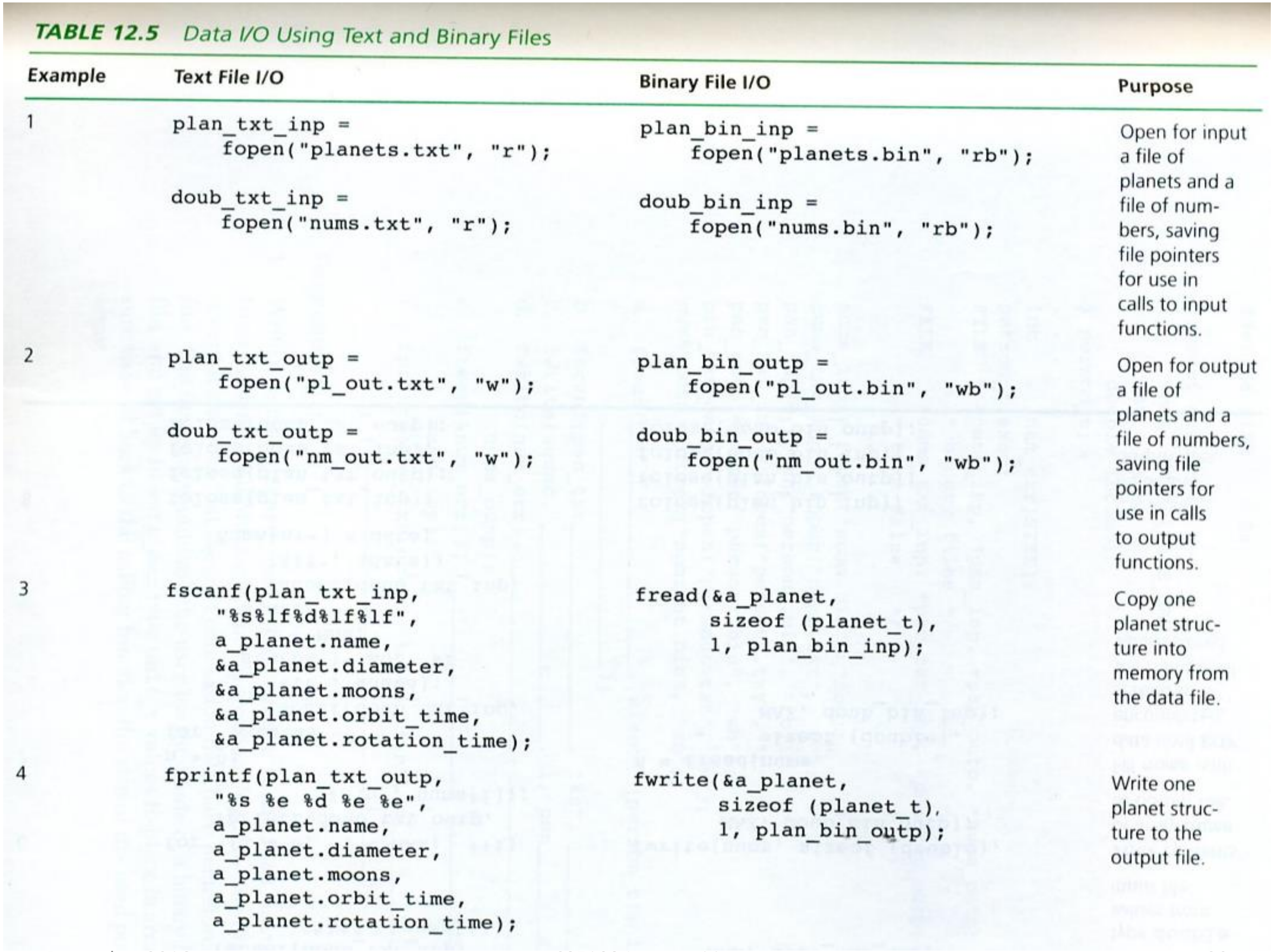


**TABLE 12.5** *Data I/O Using Text and Binary Files*

| Example | Text File I/O | Binary File I/O | Purpose |
|---|---|---|---|
| 1 | `plan_txt_inp = fopen("planets.txt", "r");` `doub_txt_inp = fopen("nums.txt", "r");` | `plan_bin_inp = fopen("planets.bin", "rb");` `doub_bin_inp = fopen("nums.bin", "rb");` | Open for input a file of planets and a file of numbers, saving file pointers for use in calls to input functions. |
| 2 | `plan_txt_outp = fopen("pl_out.txt", "w");` `doub_txt_outp = fopen("nm_out.txt", "w");` | `plan_bin_outp = fopen("pl_out.bin", "wb");` `doub_bin_outp = fopen("nm_out.bin", "wb");` | Open for output a file of planets and a file of numbers, saving file pointers for use in calls to output functions. |
| 3 | `fscanf(plan_txt_inp, "%s%lf%d%lf%lf", a_planet.name, &a_planet.diameter, &a_planet.moons, &a_planet.orbit_time, &a_planet.rotation_time);` | `fread(&a_planet, sizeof (planet_t), 1, plan_bin_inp);` | Copy one planet structure into memory from the data file. |
| 4 | `fprintf(plan_txt_outp, "%s %e %d %e %e", a_planet.name, a_planet.diameter, a_planet.moons, a_planet.orbit_time, a_planet.rotation_time);` | `fwrite(&a_planet, sizeof (planet_t), 1, plan_bin_outp);` | Write one planet structure to the output file. |

**TABLE 12.5** (continued)

| Example | Text File I/O | Binary File I/O | Purpose |
|---|---|---|---|
| 5 | `for (i = 0; i < MAX; ++i) fscanf(doub_txt_inp, "%lf", &nums[i]);` | `fread(nums, sizeof (double), MAX, doub_bin_inp);` | Fill array **nums** with type **double** values from input file. |
| 6 | `for (i = 0; i < MAX; ++i) fprintf(doub_txt_outp, "%e\n", nums[i]);` | `fwrite(nums, sizeof (double), MAX, doub_bin_outp);` | Write contents of array **nums** to output file. |
| 7 | `n = 0; for (status = fscanf(doub_txt_inp, "%lf", &data); status != EOF && n < MAX; status = fscanf(doub_txt_inp, "%lf", &data)) nums[n++] = data;` | `n = fread(nums, sizeof (double), MAX, doub_bin_inp);` | Fill **nums** with data until **EOF** encountered, setting **n** to the number of values stored. |
| 8 | `fclose(plan_txt_inp); fclose(plan_txt_outp); fclose(doub_txt_inp); fclose(doub_txt_outp);` | `fclose(plan_bin_inp); fclose(plan_bin_outp); fclose(doub_bin_inp); fclose(doub_bin_outp);` | Close all input and output files. |

## Case Study: Steganography

**Steganography*** is the practice of concealing a file, message, image, or video within another file, message, image, or video.

* https://en.wikipedia.org/wiki/Steganography

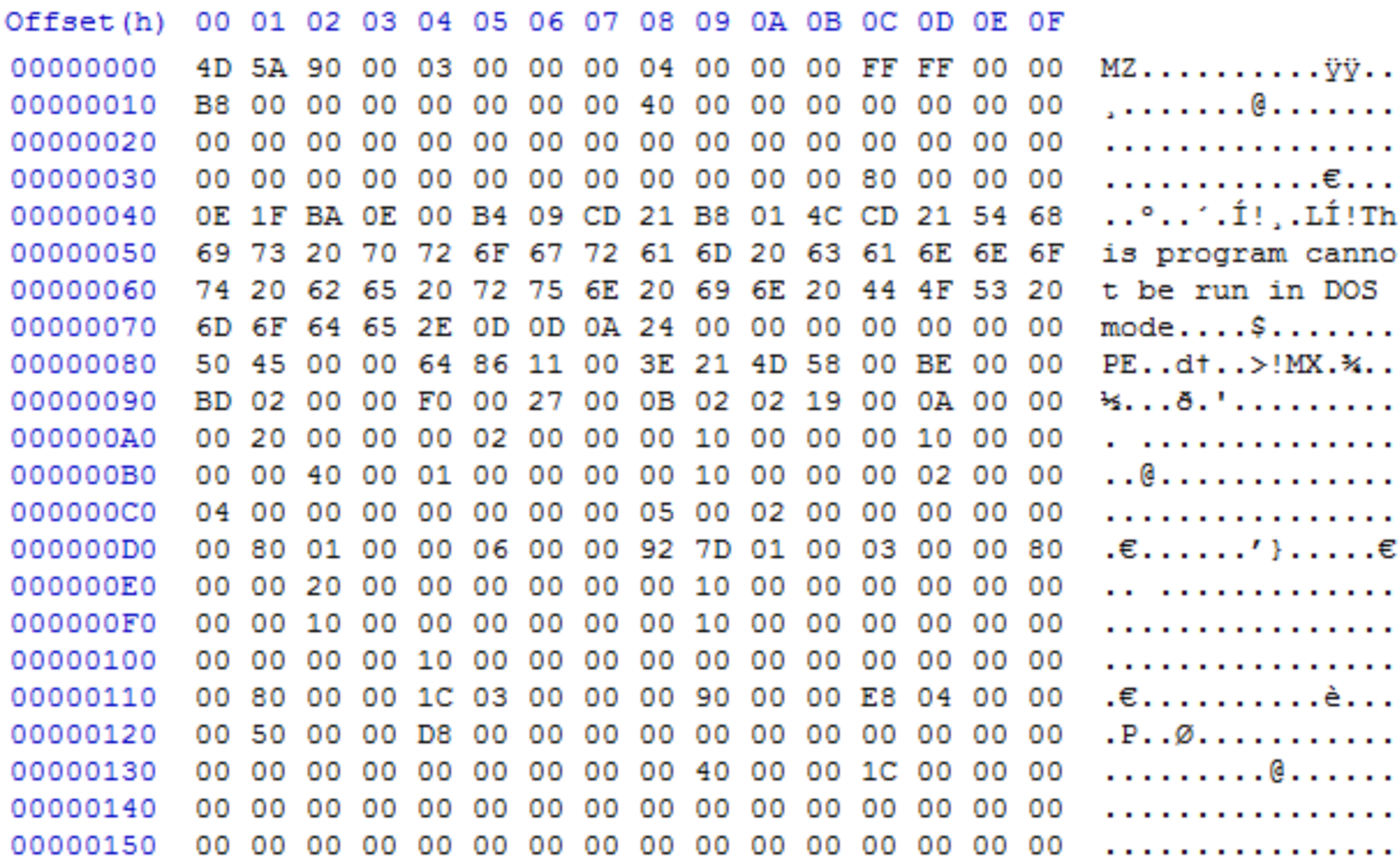
## Case Study: Steganography

**Steganography**[*] is the practice of concealing a file, message, image, or video within another file, message, image, or video.

Greek words **steganos** (στεγανός), meaning "covered, concealed, or protected", and **graphein** (γράφειν) meaning "writing".

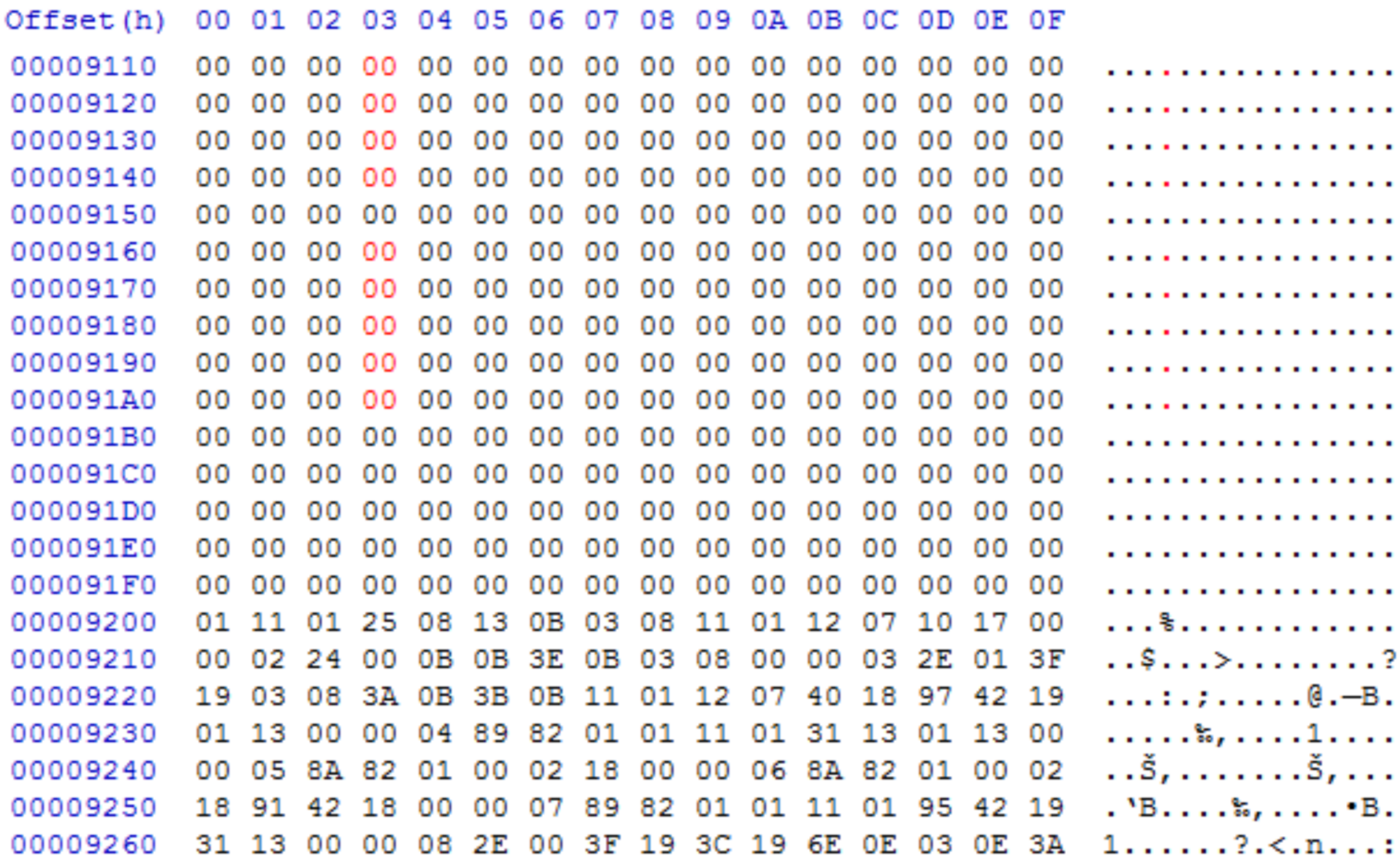
[*] https://en.wikipedia.org/wiki/Steganography

## Case Study: Steganography



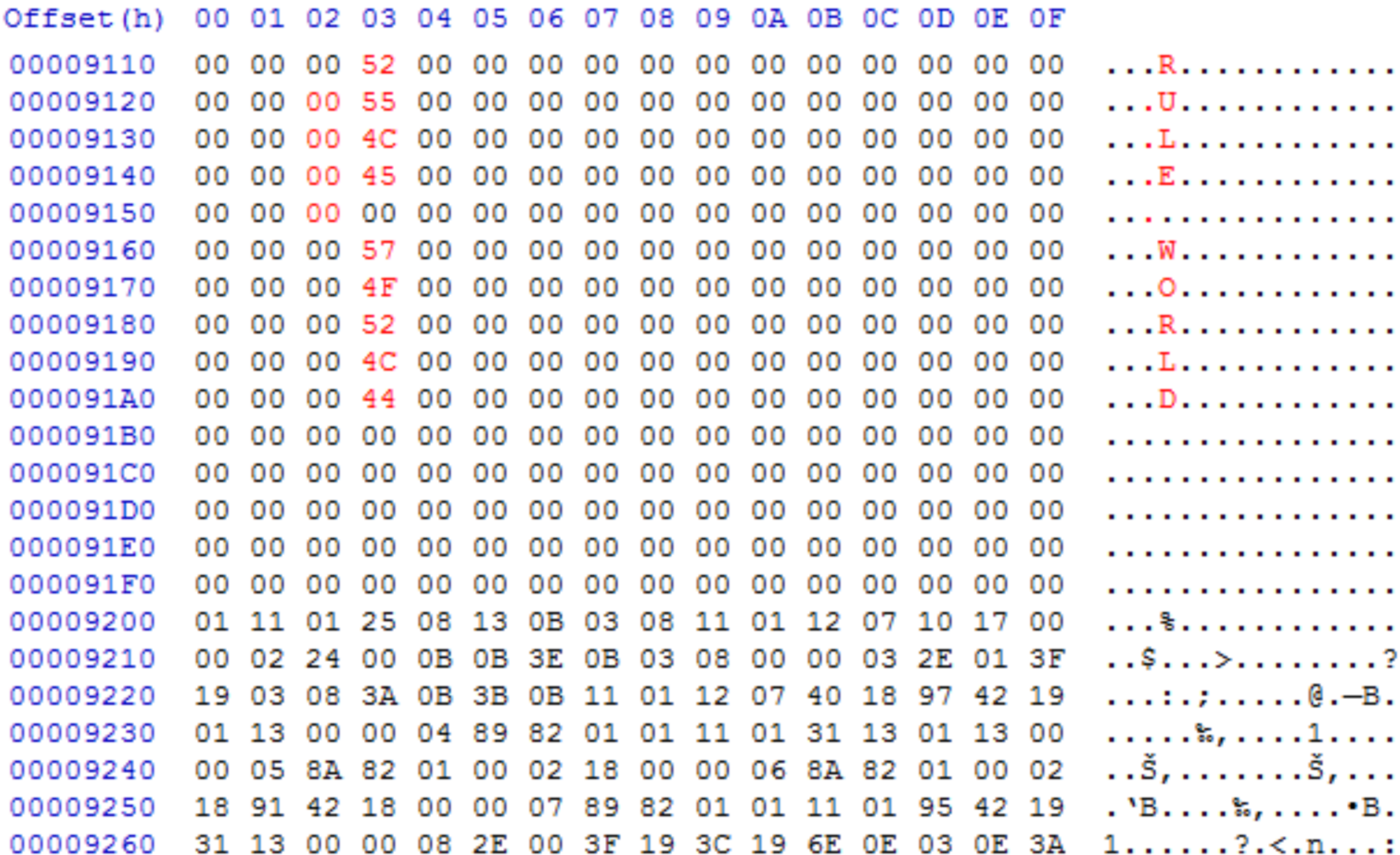
```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00   MZ..........ÿÿ..
00000010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00   ¸.......@.......
00000020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000030  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00   ............€...
00000040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68   ..º..´.Í!¸.LÍ!Th
00000050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F   is program canno
00000060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20   t be run in DOS 
00000070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00   mode....$.......
00000080  50 45 00 00 64 86 11 00 3E 21 4D 58 00 BE 00 00   PE..d†..>!MX.¾..
00000090  BD 02 00 00 F0 00 27 00 0B 02 02 19 00 0A 00 00   ½...ð.'.........
000000A0  00 20 00 00 00 02 00 00 00 10 00 00 00 10 00 00   . ..............
000000B0  00 00 40 00 01 00 00 00 00 10 00 00 00 02 00 00   ..@.............
000000C0  04 00 00 00 00 00 00 00 05 00 02 00 00 00 00 00   ................
000000D0  00 80 01 00 00 06 00 00 92 7D 01 00 03 00 00 80   .€......’}.....€
000000E0  00 00 20 00 00 00 00 00 00 10 00 00 00 00 00 00   .. .............
000000F0  00 00 10 00 00 00 00 00 00 10 00 00 00 00 00 00   ................
00000100  00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00   ................
00000110  00 80 00 00 1C 03 00 00 00 90 00 00 E8 04 00 00   .€..........è...
00000120  00 50 00 00 D8 00 00 00 00 00 00 00 00 00 00 00   .P..Ø...........
00000130  00 00 00 00 00 00 00 00 00 40 00 00 1C 00 00 00   .........@......
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

# Case Study: Steganography



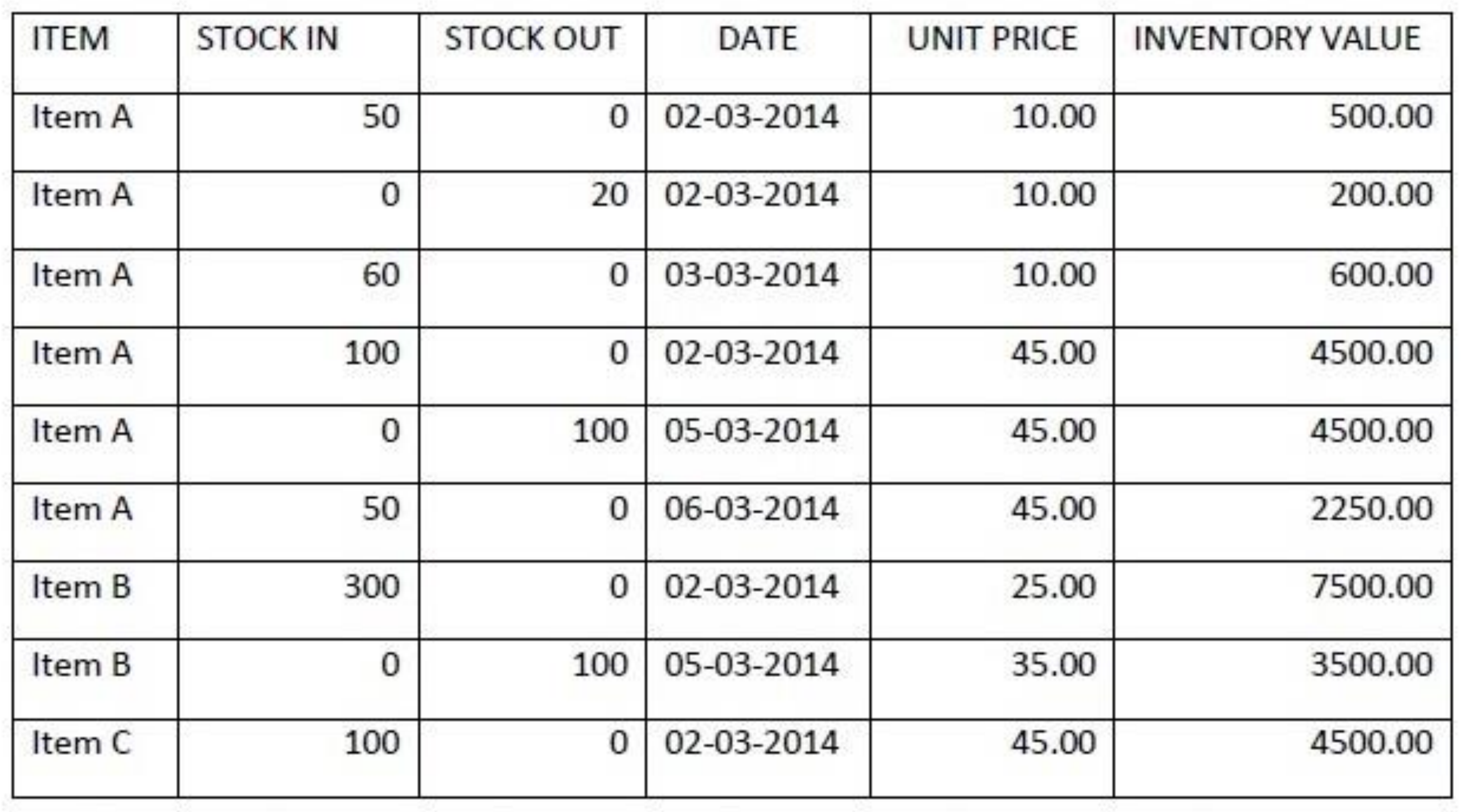
```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00009110   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009120   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009130   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009140   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009150   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009160   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009170   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009190   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091A0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009200   01 11 01 25 08 13 0B 03 08 11 01 12 07 10 17 00   ...%............
00009210   00 02 24 00 0B 0B 3E 0B 03 08 00 00 03 2E 01 3F   ..$...>........?
00009220   19 03 08 3A 0B 3B 0B 11 01 12 07 40 18 97 42 19   ...:.;.....@.—B.
00009230   01 13 00 00 04 89 82 01 01 11 01 31 13 01 13 00   .....‰‚....1....
00009240   00 05 8A 82 01 00 02 18 00 00 06 8A 82 01 00 02   ..Š‚.......Š‚...
00009250   18 91 42 18 00 00 07 89 82 01 01 11 01 95 42 19   .‘B....‰‚....•B.
00009260   31 13 00 00 08 2E 00 3F 19 3C 19 6E 0E 03 0E 3A   1......?.<.n...:
```

# Case Study: Steganography



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00009110   00 00 00 52 00 00 00 00 00 00 00 00 00 00 00 00   ...R............
00009120   00 00 00 55 00 00 00 00 00 00 00 00 00 00 00 00   ...U............
00009130   00 00 00 4C 00 00 00 00 00 00 00 00 00 00 00 00   ...L............
00009140   00 00 00 45 00 00 00 00 00 00 00 00 00 00 00 00   ...E............
00009150   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009160   00 00 00 57 00 00 00 00 00 00 00 00 00 00 00 00   ...W............
00009170   00 00 00 4F 00 00 00 00 00 00 00 00 00 00 00 00   ...O............
00009180   00 00 00 52 00 00 00 00 00 00 00 00 00 00 00 00   ...R............
00009190   00 00 00 4C 00 00 00 00 00 00 00 00 00 00 00 00   ...L............
000091A0   00 00 00 44 00 00 00 00 00 00 00 00 00 00 00 00   ...D............
000091B0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091C0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
000091F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00009200   01 11 01 25 08 13 0B 03 08 11 01 12 07 10 17 00   ...%............
00009210   00 02 24 00 0B 0B 3E 0B 03 08 00 00 03 2E 01 3F   ..$...>.........?
00009220   19 03 08 3A 0B 3B 0B 11 01 12 07 40 18 97 42 19   ...:.;.....@.—B.
00009230   01 13 00 00 04 89 82 01 01 11 01 31 13 01 13 00   .....‰‚....1....
00009240   00 05 8A 82 01 00 02 18 00 00 06 8A 82 01 00 02   ..Š‚......Š‚....
00009250   18 91 42 18 00 00 07 89 82 01 01 11 01 95 42 19   .'B....‰‚....•B.
00009260   31 13 00 00 08 2E 00 3F 19 3C 19 6E 0E 03 0E 3A   1......?.<.n...:
```

## Case Study: Database Inquiry Problem



| ITEM | STOCK IN | STOCK OUT | DATE | UNIT PRICE | INVENTORY VALUE |
|---|---|---|---|---|---|
| Item A | 50 | 0 | 02-03-2014 | 10.00 | 500.00 |
| Item A | 0 | 20 | 02-03-2014 | 10.00 | 200.00 |
| Item A | 60 | 0 | 03-03-2014 | 10.00 | 600.00 |
| Item A | 100 | 0 | 02-03-2014 | 45.00 | 4500.00 |
| Item A | 0 | 100 | 05-03-2014 | 45.00 | 4500.00 |
| Item A | 50 | 0 | 06-03-2014 | 45.00 | 2250.00 |
| Item B | 300 | 0 | 02-03-2014 | 25.00 | 7500.00 |
| Item B | 0 | 100 | 05-03-2014 | 35.00 | 3500.00 |
| Item C | 100 | 0 | 02-03-2014 | 45.00 | 4500.00 |

## Case Study: Database Inquiry Problem

**Possible queries**

What items that cost less than $20 are available?

## Case Study: Database Inquiry Problem

**Possible queries**

What items that cost less than $20 are available?

What items are new (arrived after 04-03-2014)?

## Case Study: Database Inquiry Problem

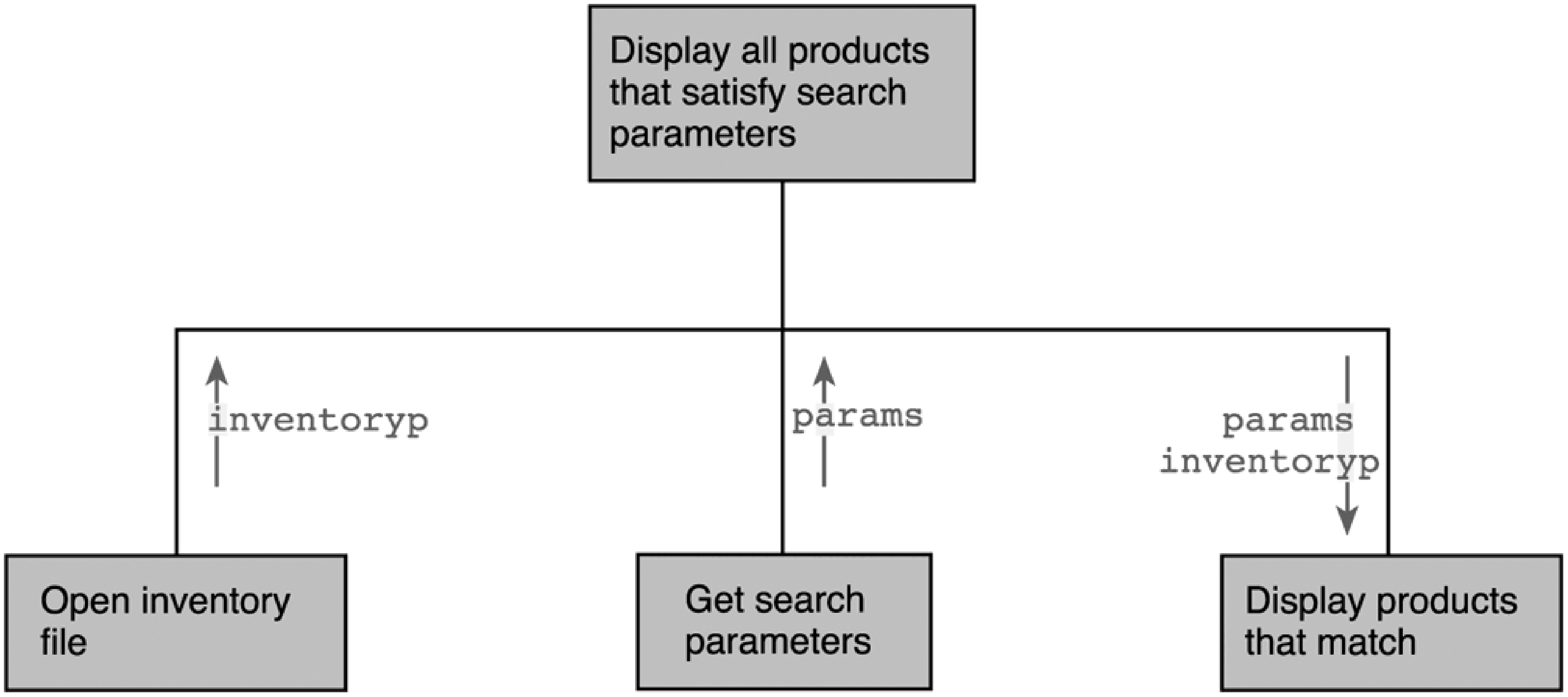
**Possible queries**

What items that cost less than $20 are available?

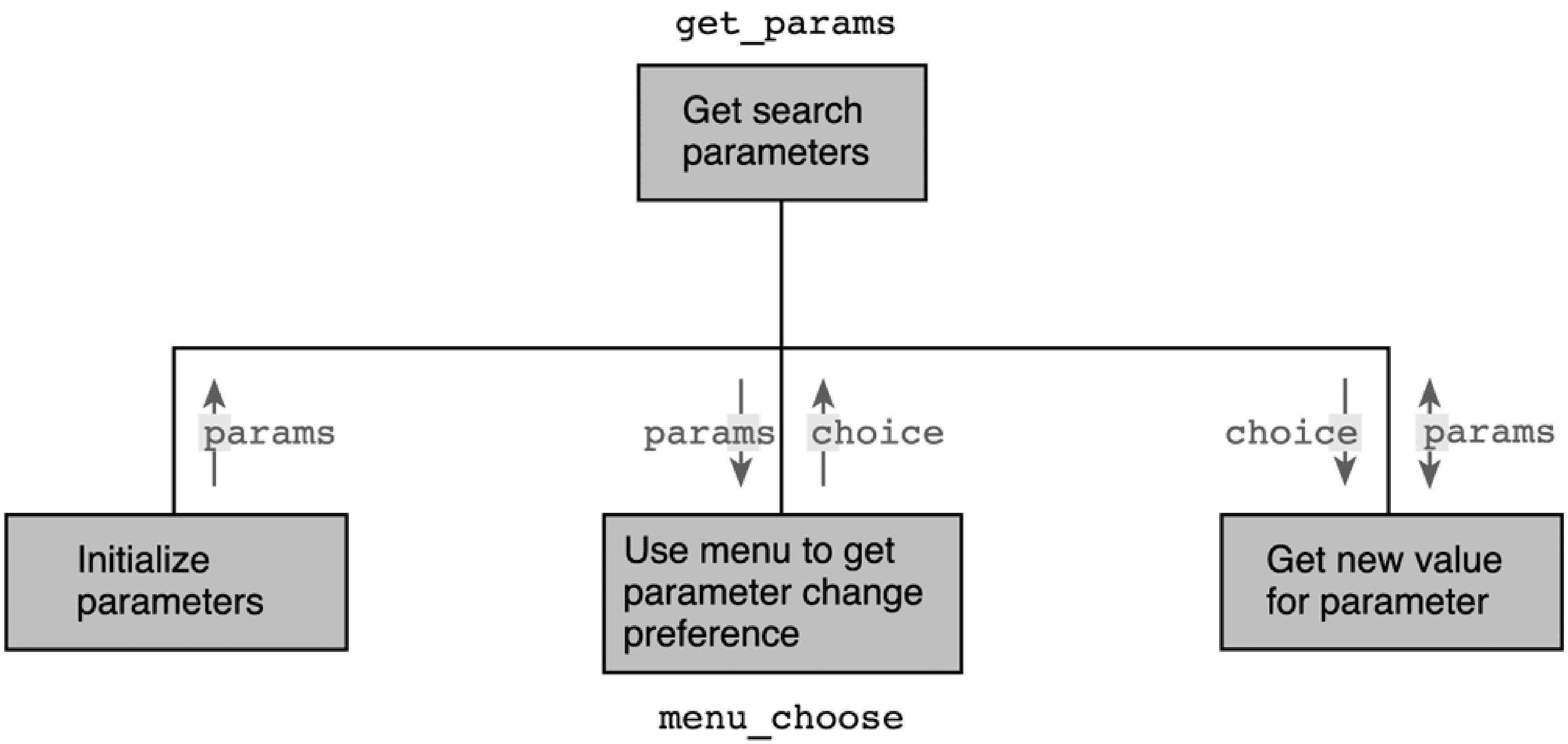
What items are new (arrived after 04-03-2014)?

What items are old (arrived before 04-03-2014)?

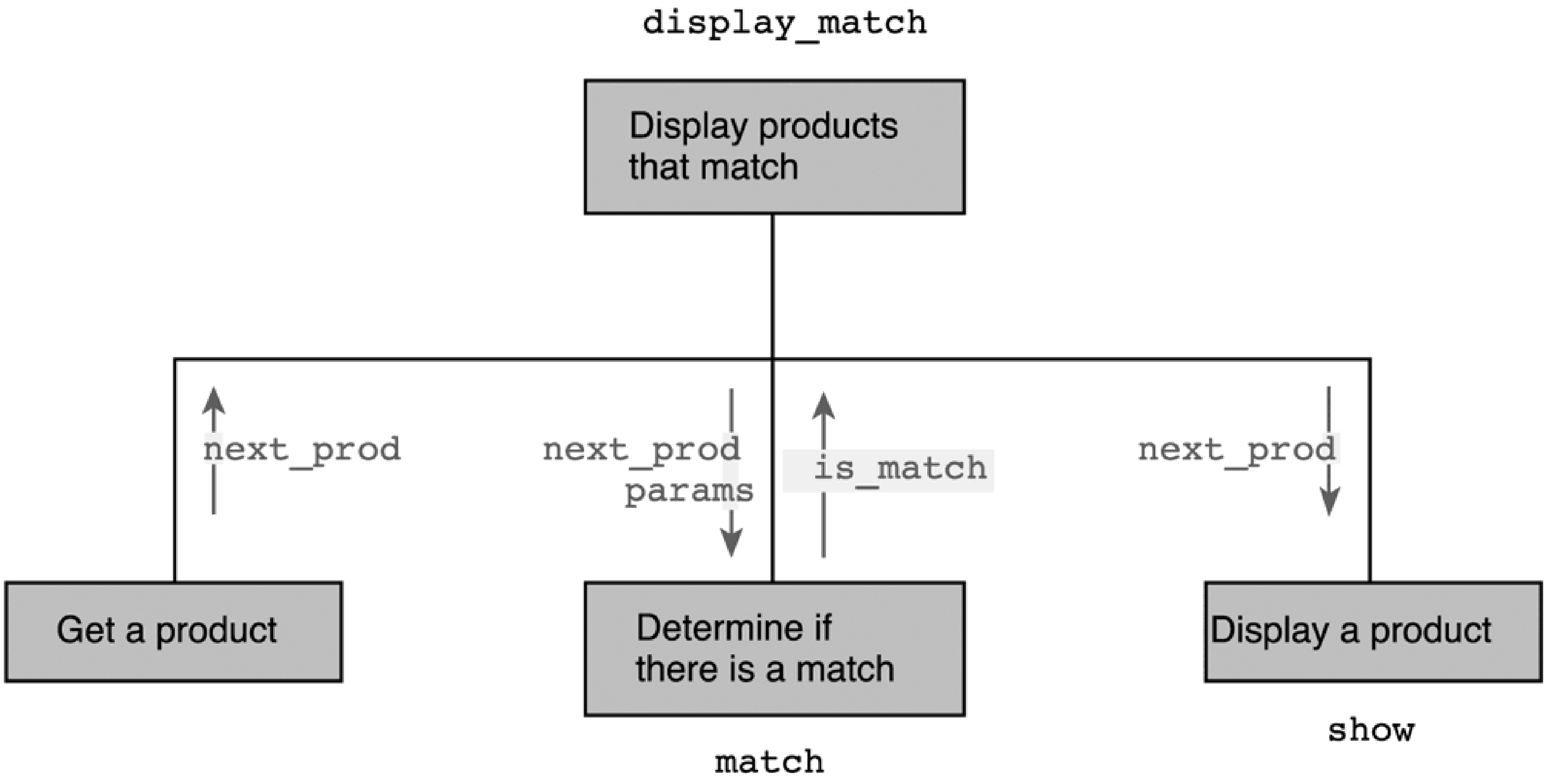
# Case Study: Database Inquiry Problem


Display all products that satisfy search parameters
inventoryp
params
params
inventoryp
Open inventory file
Get search parameters
Display products that match

## Structure Chart for get_params



# Structure Chart for display_match


display_match
Display products that match
next_prod
next_prod
params
is_match
next_prod
Get a product
Determine if there is a match
Display a product
match
show