

Reading a Set of Data

A Simple Payroll Program

- **Problem:**

Write a complete C program to do the following: Read in data containing information about an employee of a company. The data will contain the employee's ID, hours worked in a week, and rate of pay per hour. Compute the employee's pay for the week. Print the relevant information about this employee. Then repeat the same steps for the next employee until all employees have been processed. Count the number of employees processed and print the total at the end.

- **Typical Data:**

1234	35.0	14.20
2345	11.3	5.87

- **Pseudocode for the Program:**

do the following for each employee
read id
read hours worked
read rate
*calculate pay = hours * rate*
print relevant information
count this employee
print total number of employees

The while Statement

- The **while statement** (or **while loop**) repeats a group of statements while a given condition is true.
- **The while Statement:**
while (*condition*)
 body (statement) of the loop
- CPU action when reaching while statement:
 - It evaluates *condition* which can be TRUE or FALSE.
 - If *condition* is TRUE, then:
 - a) the CPU executes the body of the loop.
 - b) when the CPU completes the body of the loop, it jumps back to the while statement and starts the procedure all over again.
 - If *condition* is FALSE, the CPU jumps to the statement following the loop statement.

- Example:


```
n = 1;
while (n < 10)
    n * = 2;
printf("The smallest power of 2 greater than 10 is %d\n",n)
```
- Example:


```
n = 0;
while (n != 20) {
    printf("Learning C is fun!\n");
    n+ + ;
}
```
- Example:


```
n = 1;
while (n < = 10) {
    printf("Enter a value for x:\n");
    scanf("%d",&x);
    y = x * 2;
    printf("y = %d",y);
    n+ + ;
}
```
- Example (comparison using a **for** statement):


```
for (n = 1; n < = 10; n+ + ) {
    printf("Enter a value for x:\n");
    scanf("%d",&x);
    y = x * 2;
    printf("y = %d",y);
}
```

- Example:
Trace the values of the variables n and x.

```

/* program loop */
#include <stdio.h>
void main()
{
    int n,x;

    x = 1;
    n = 0;
    while (n != 3) {
        x *= 2;
        n++ ;
    }
    printf("%d",x);
}

```

	n	x
	?	?
init.	0	1
n != 3 - True	1	2
n != 3 - True	2	4
n != 3 - True	3	8
n != 3 - False		
screen		8

Using a while Loop

- To find the last employee, we can make up a fake ID to represent the last employee and test for that in a while loop. (Let us assign the ID for the fake employee is -1111.)

- **Pseudocode for the Program:**

```
int id;

while (id != -1111) {
    read id
    read hours worked
    read rate
    calculate pay = hours * rate
    print relevant information
    count this employee
}
print total number of employees
```

Interactive Data Entry

- **The Function scanf()**

```
scanf(format string,variable list);
```

where,

format string = one or more format specifications

variable list = one or more variable names, each preceded by an & (this represents the address of the variable)

- **Conversion Specifications for scanf and printf:**

<u>Data Type</u>	<u>scanf</u>	<u>printf</u>
int	%d	%d
float	%f or %e	%f or %e
double	%lf or %le	%f or %e
char	%c	%c

- **Question:**

How does the operator know what to enter?

- **Prompts & Interactive Programming:**

```
int id;
```

```
double hours,rate;
```

```
printf("Please enter an ID:\n");
```

```
scanf("%d",&id);
```

```
printf("Please enter the hours worked:\n");
```

```
scanf("%lf",&hours);
```

```
printf("Please enter the rate of pay:");
```

```
scanf("%lf",&rate);
```

Reading Multiple Data Values

- Example

```
printf("Please enter an ID, hours, and rate:\n");  
scanf("%d %f %f", &id, &hours, &rate);
```

- The blanks between the conversion specifications are not important. scanf skips them.
- To enter data for this call, the user types in three separate values followed by the ENTER key.

e.g., 1234 10.5 4.65 (ENTER)

- The values themselves may be separated by any number of blanks, TABs, or ENTERs, **but not by commas**.
- Note: If a real value is entered, where an integer is expected, the value will be truncated.

e.g., 1234.5 10.5 4 (ENTER)

- Note: If an alphabetic character is entered when a number is expected, the variable awaiting input will keep its old value.

e.g., 1234 10.5 XYZ (ENTER)

e.g., WXYZ 10.5 4.65 (ENTER)

- **Revised Program:**

```
/* payroll program */
#include <stdio.h>
void main()
{
    int id;
    double hours,rate,pay;

    while (id != -1111) {
        printf("Please enter an ID (enter -1111 to stop):\n");
        scanf("%d",&id);
        printf("Please enter the hours worked:\n");
        scanf("%lf",&hours);
        printf("Please enter the rate of pay:");
        scanf("%lf",&rate);
        pay = hours * rate;
        printf("Employee %d worked %f hours at a rate of pay"
            " of $%f earning $%f\n",id,hours,rate,pay);
        count this employee
    }
    print total number of employees
}
```

- **FATAL FLAW IN WHILE LOOP!!!**

What happens the first time we reach the while loop?

- **Another Revision:**

```
/* payroll program */
#include <stdio.h>
void main()
{
    int id;
    double hours,rate,pay;

    printf("Please enter an ID (enter -1111 to stop):\n");
    scanf("%d",&id);
    while (id != -1111) {
        printf("Please enter the hours worked:\n");
        scanf("%lf",&hours);
        printf("Please enter the rate of pay:");
        scanf("%lf",&rate);
        pay = hours * rate;
        printf("Employee %d worked %f hours at a rate of pay"
            " of $%f earning $%f\n",id,hours,rate,pay);
        count this employee
    }
    print total number of employees
}
```

- **WE STILL HAVE A FLAW!!!**

- **Yet Another Revision:**

```
/* payroll program */
#include <stdio.h>
void main()
{
    int id;
    double hours,rate,pay;

    printf("Please enter an ID (enter -1111 to stop):\n");
    scanf("%d",&id);
    while (id != -1111) {
        printf("Please enter the hours worked:\n");
        scanf("%lf",&hours);
        printf("Please enter the rate of pay:");
        scanf("%lf",&rate);
        pay = hours * rate;
        printf("Employee %d worked %f hours at a rate of pay"
            " of $%f earning $%f\n",id,hours,rate,pay);
        count this employee
        printf("Please enter an ID (enter -1111 to stop):\n");
        scanf("%d",&id);
    }
    print total number of employees
}
```

Counting the Number of Employees

- Set up a counting variable to keep track of the number of employees processed.

- **Complete Program:**

```
/* payroll program */
#include <stdio.h>
void main()
{
    int id;                //employee id
    double hours,rate,pay;
    int numemp;          //count the employees

    numemp = 0;          //initialize the counter
    printf("Please enter an ID (enter -1111 to stop):\n");
    scanf("%d",&id);      //read ID
    while (id != -1111) { //check for fake ID
        printf("Please enter the hours worked:\n");
        scanf("%lf",&hours); //read hours worked
        printf("Please enter the rate of pay:");
        scanf("%lf",&rate); //read rate of pay
        pay = hours * rate; //calculate pay
        printf("Employee %d worked %f hours at a rate of pay"
            " of $%f earning $%f\n",id,hours,rate,pay);
        numemp+ + ;      //increment counter
        printf("Please enter an ID (enter -1111 to stop):\n");
        scanf("%d",&id); //read new ID
    }
    printf("\nWe have processed %d employees\n",numemp);
}
```

Initializing Variables within the Declaration

- **Complete Program:**

```
/* payroll program */
#include <stdio.h>
void main()
{
    int id;                //employee id
    double hours,rate,pay;
    int numemp = 0;      //count the employees

    printf(" Please enter an ID (enter -1111 to stop):\n");
    scanf("%d",&id);      //read ID
    while (id != -1111) { //check for fake ID
        printf(" Please enter the hours worked:\n");
        scanf("%lf",&hours); //read hours worked
        printf(" Please enter the rate of pay:");
        scanf("%lf",&rate); //read rate of pay
        pay = hours * rate; //calculate pay
        printf(" Employee %d worked %f hours at a rate of pay"
            " of $%f earning $%f\n",id,hours,rate,pay);
        numemp+ + ;      //increment counter
        printf(" Please enter an ID (enter -1111 to stop):\n");
        scanf("%d",&id); //read new ID
    }
    printf("\nWe have processed %d employees\n",numemp);
}
```

Printing Numbers Neatly Field Width & Decimal Precision

- A real number with the specifier %f prints in **fixed point notation**. The decimal points is fixed in its normal position and the computer prints 6 decimal digits to its right.
- To print a value occupying a specific number of print positions, we define its **field width**. This is possible both for real numbers and for integers.
- For real numbers, we can also specify how many positions are to be printed to the right of the decimal point. This is called the **decimal precision**.
- The format for integers is:
%w d - where w is an integer indicating the field width.
- The format for real numbers is:
%w .df or %w .de - where w is the field width (including the decimal point) and d is the decimal precision.

- Examples:

```
double num = 32.678
```

```
int inum = 12345
```

<u>Format</u>	<u>Output</u>	<u>Comment</u>
printf("%e",num);	3.267800e+ 01	
printf("%f",num);	32.678000	
printf("%E",num);	3.267800E+ 01	
printf("%7.3f",num);	32.678	leading space
printf("%4.1f",num);	32.7	rounding
printf("%4.0f",num);	33	rounds to an integer
printf("%.2f",num);	32.68	field width omitted
printf("%5.3f",num);	32.678	adjusts width
printf("%1.1f",num);	32.7	adjusts width
printf("%7d",inum);	12345	leading spaces

Left and Right Alignment (Justification)

- Normally, numbers are printed **right justified** within their field width.
- To left justify a number, insert a - between the % and the field width.

- Example:

```
for (i = 99; i < 102; i+ + )  
    printf("%5d %c",i,'a');
```

```
    99 a  
   100 a  
   101 a
```

```
for (i = 99; i < 102; i+ + )  
    printf("%-5d %c",i,'a');
```

```
99      a  
100     a  
101     a
```

Final Version of Program

```
/* payroll program */
#include <stdio.h>
void main()
{
    int id;                                //employee id
    double hours,rate,pay;
    int numemp = 0;                        //count the employees

    printf("Please enter an ID (enter -1111 to stop):\n");
    scanf("%d",&id);                        //read ID
    while (id != -1111) {                  //check for fake ID
        printf("Please enter the hours worked:\n");
        scanf("%lf",&hours);                //read hours worked
        printf("Please enter the rate of pay:");
        scanf("%lf",&rate);                //read rate of pay
        pay = hours * rate;                //calculate pay
        printf("Employee %d worked %4.1f hours at a rate of"
            " pay of $%5.2f earning $%7.2f\n", id, hours, rate,
            pay);
        numemp+ + ;                        //increment counter
        printf("Please enter an ID (enter -1111 to stop):\n");
        scanf("%d",&id);                    //read new ID
    }
    printf("\nWe have processed %d employees\n",numemp);
}
```

The if-else (Conditional) Statement

- **The if-else (Conditional) Statement:**

```
if (condition)
    statement_1
else
    statement_2
```

- **Alternate Forms:**

```
if (condition) {
    compound_statement_1
}
else {
    compound_statement_2
}
```

- **Example:**

Find the maximum of x and y.

a)

```
max = y;
if (x > y) max = x;
printf("%d",max);
```

b)

```
if (x > y) max = x;
if (x <= y) max = y;
printf("%d",max);
```

c)

```
if (x > y)
    max = x;
else
    max = y;
printf("%d",max);
```

More Complex Payroll Program

Assume that tax is to be deducted from employee's gross pay. For employees that earn less than \$300 a week, the tax rate is 15% of gross. For all other employees, the tax rate is 28%. Modify the payroll program to compute each employee's net pay (which is the gross pay less the tax).

```
/* payroll program with tax deductions */
#include <stdio.h>
void main()
{
    int id;                                //employee id
    double hours,rate,pay,tax,netpay;
    int numemp = 0;                        //count the employees

    printf("Please enter an ID (enter -1111 to stop):\n");
    scanf("%d",&id);                        //read ID
    while (id != -1111) {                  //check for fake ID
        printf("Please enter the hours worked:\n");
        scanf("%lf",&hours);                //read hours worked
        printf("Please enter the rate of pay:");
        scanf("%lf",&rate);                //read rate of pay
        pay = hours * rate;                //calculate pay
        if (pay < 300)
            tax = 0.15 * pay;
        else
            tax = 0.28 * pay;
        netpay = pay - tax;
        printf("Employee %d worked %4.1f hours at a rate of pay of "
            "$%5.2f earning $%7.2f\n", id, hours, rate, pay);
        printf("tax withheld was $%7.2f, leaving net pay of $%7.2f",
            tax, netpay);
        numemp+ + ;                        //increment counter
        printf("Please enter an ID (enter -1111 to stop):\n");
        scanf("%d",&id);                    //read new ID
    }
    printf("\nWe have processed %d employees\n",numemp);
}
```

The Conditional (?:) Operator

- C allows statements of the following form:

variable = conditional expression;

where a conditional expression is of the form:

expr1 ? expr2 : expr3

where

expr1 is a condition that evaluates to true or false,

if expr1 is true

expr2 is the value of the conditional expression

else

expr3 is the value of the conditional expression

- Example:

```
if (x > y)
```

```
    max = x;
```

```
else
```

```
    max = y;
```

can be written as:

```
max = (x > y) ? x : y;
```

Payroll Program (one last time)

```
/* payroll program with tax deductions */
#include <stdio.h>
void main()
{
    int id;                                //employee id
    double hours,rate,pay,tax,netpay;
    int numemp = 0;                        //count the employees

    printf("Please enter an ID (enter -1111 to stop):\n");
    scanf("%d",&id);                        //read ID
    while (id != -1111) {                  //check for fake ID
        printf("Please enter the hours worked:\n");
        scanf("%lf",&hours);                //read hours worked
        printf("Please enter the rate of pay:");
        scanf("%lf",&rate);                //read rate of pay
        pay = hours * rate;                //calculate pay
        tax = (pay < 300) ? 0.15 * pay : 0.28 * pay;
        netpay = pay - tax;
        printf("Employee %d worked %4.1f hours at a rate of"
            " pay of $%5.2f earning $%7.2f\n", id, hours, rate,
            pay);
        printf("tax withheld was $%7.2f, leaving net pay of "
            "$%7.2f", tax, netpay);
        numemp+ + ;                        //increment counter
        printf("Please enter an ID (enter -1111 to stop):\n");
        scanf("%d",&id);                    //read new ID
    }
    printf("\nWe have processed %d employees\n",numemp);
}
```

Input Using fscanf()

- **fscanf() Syntax:**

```
fscanf(sourcefile, "format_string", addresses_of_variables);
```

- **Examples:**

```
FILE * infile;  
FILE * changes;  
int num;  
int num1, num2;
```

```
infile = fopen("letter.dat", "r");  
changes = fopen("c:\\hw\\changes.dat", "r");
```

```
fscanf(infile, "%d", &num);  
fscanf(changes, "%d %d", &num1, &num2);  
fscanf(stdin, "%d", &num);
```

```
fclose(infile);  
fclose(changes);
```

NOTE: DO NOT OPEN OR CLOSE stdin

- **Checking for File Existence:**

```
#include <stdlib.h> //must include stdlib.h  
FILE * infile;  
int num;  
  
infile = fopen("numbers.dat", "r");  
if (infile == NULL) {  
    fprintf(stderr, "error in opening file numbers.dat\n");  
    exit(1);  
}  
while (fscanf(infile, "%d", &num) != EOF)  
    fprintf(stdout, "The number is %d\n", num);  
fclose(infile);
```

● **Example - using output file:**

```
/* payroll program with tax deductions */
#include < stdio.h>
void main()
{
    int id;                                //employee id
    double hours,rate,pay,tax,netpay;
    int numemp = 0;                        //count the employees
    FILE * outfile;

    outfile = fopen("b:payroll.out","w");
// outfile = stdout;                      //un-comment for testing pgm

    printf("Please enter an ID (enter -1111 to stop):\n");
    scanf("%d",&id);                       //read ID
    while (id != -1111) {                  //check for fake ID
        printf("Please enter the hours worked:\n");
        scanf("%lf",&hours);                //read hours worked
        printf("Please enter the rate of pay:");
        scanf("%lf",&rate);                //read rate of pay
        pay = hours * rate;                //calculate pay
        tax = (pay < 300) ? 0.15 * pay : 0.28 * pay;
        netpay = pay - tax;
        fprintf(outfile,"Employee %d worked %4.1f hours at a rate of"
            " pay of $%5.2f earning $%7.2f\n", id, hours, rate, pay);
        fprintf(outfile,"tax withheld was $%7.2f, leaving net"
            " pay of $%7.2f", tax, netpay);
        numemp+ + ;                        //increment counter
        printf("Please enter an ID (enter -1111 to stop):\n");
        scanf("%d",&id);                    //read new ID
    }
    fprintf(outfile,"\nWe have processed %d employees\n",numemp);

    fclose(outfile);
}
```

- **Example - using input & output files:**

```
/* payroll program with tax deductions */
#include <stdio.h>
void main()
{
    int id;                                //employee id
    double hours,rate,pay,tax,netpay;
    int numemp = 0;                         //count the employees
    FILE * infile, * outfile;

    infile = fopen("b:payroll.dat","r");
    outfile = fopen("b:payroll.out","w");
    // infile = stdin;                       //un-comment for testing pgm
    // outfile = stdout;                     //un-comment for testing pgm

    printf("Please enter an ID (enter -1111 to stop):\n");
    fscanf(infile,"%d",&id);                 //read ID
    while (id != -1111) {                   //check for fake ID
        printf("Please enter the hours worked:\n");
        fscanf(infile,"%lf",&hours);         //read hours worked
        printf("Please enter the rate of pay:");
        fscanf(infile,"%lf",&rate);         //read rate of pay
        pay = hours * rate;                 //calculate pay
        tax = (pay < 300) ? 0.15 * pay : 0.28 * pay;
        netpay = pay - tax;
        fprintf(outfile,"Employee %d worked %4.1f hours at a rate of"
            " pay of $%5.2f earning $%7.2f\n", id, hours, rate, pay);
        fprintf(outfile,"tax withheld was $%7.2f, leaving net"
            " pay of $%7.2f", tax, netpay);
        numemp+ + ;                         //increment counter
        printf("Please enter an ID (enter -1111 to stop):\n");
        fscanf(infile,"%d",&id);             //read new ID
    }
    fprintf(outfile,"\nWe have processed %d employees\n",numemp);

    fclose(infile);
    fclose(outfile);
}
```

Reading In File Names

- Hard-coding the names of specific files directly into a program is sometimes very limiting.
- A program should be capable of reading from and writing to many different files.
- One solution is to read in the file names.

- **Reading In File Names:**

```
FILE * infile, * outfile;
```

```
char infilename[30], outfilefilename[30];
```

```
printf("Enter the name of the input file:\n");
```

```
scanf("%s",infilename); // IT'S A POINTER! NO & NEEDED
```

```
printf("Enter the name of the output file:\n");
```

```
scanf("%s",outfilefilename);
```

```
infile = fopen(infilename,"r");
```

```
outfile = fopen(outfilefilename,"w");
```