```
/*
 -----------------------------------------------------------------
 PROGRAM NAME: Lab 2
 PROGRAMMER:   Samuel Jentsch
 CLASS:        CSC 214, Spring 2014
 INSTRUCTOR:   Dr. Strader
 DATE STARTED: January 30, 2014
 DUE DATE:     February 3, 2014
 REFERENCES:   Computer Organization and Architecture by Null and Lobor
 Beginning C by Ivor Horton
 Dr. Strader: assignment information sheet

 PROGRAM PURPOSE:
 Read in characters from a file and display the characters with their
 corresponding hex values. If a special character is encountered a
 readable text representation of the special character is displayed.

 VARIABLES/CONSTANTS:
 kDUMP_SIZE = 16. Size of the array used to hold characters for dump.

 METHODS:
 Prints the character and hex representation of the characters
 present in dumpArray.
 void print_group(char dumpArray[], int dumpSize, int offset);

 void print_space(char specialChar);
 int decToHex(int decimal);

 FILES USED:
 Lab2.dat
 -----------------------------------------------------------------
 */

#include <stdio.h>

#define kDUMP_SIZE 16

//Method headers
void print_group(char dumpArray[], int dumpSize, int offset);
void print_space(char specialChar);
int decToHex(int decimal);

int main(int argc, const char * argv[])
{
    FILE *dataFile;

    dataFile = fopen("../instr/lab2.dat", "r");

    //Holds the characters to be converted to hex and dumped.
    char dumpArray[kDUMP_SIZE];


    if(dataFile == NULL) {
        printf("Error, file not found.");
        return -1;
    }

    //Character read from the file.
    char inputChar = 0;

    int charactersRead = 0;
    int offsetCount = 0;

    while((inputChar = fgetc(dataFile)) != EOF) {
        dumpArray[charactersRead] = inputChar;
        charactersRead++;
        if (charactersRead == kDUMP_SIZE) {
            print_group(dumpArray, kDUMP_SIZE, (16 * offsetCount));
            charactersRead = 0;
            offsetCount++;
        }
    }

    if (charactersRead != 0) {
        print_group(dumpArray, charactersRead, (16 * offsetCount));
    }

    fclose(dataFile);

    return 0;
}

void print_group(char dumpArray[], int dumpSize, int offset) {
    //----------------------------------------------------//
    //Prints the hex and character value for the character
    //array passed (dumpArray). If a special character is
    //encountered it is printed using the print_space method.
    //Preconditions: dumpArray[] passed as parameter that method
    //prints as hex and as characters. dumpSize passed as int
    //specifying the number of characters to print. Offset
    //passed as int specifying how many characters have been
    //encountered (increments of 16).
    //Postconditions: the hex and character value for each
    //character in dumpArray is printed out. Special characters
    //are passed to print_space.
    //----------------------------------------------------//

    printf("%06d ", offset);
```

```c

    int i;

    for (i = 0; i < dumpSize; i++) {
        printf("%02x ", dumpArray[i]);
    }
    printf("\n       ");
    for (i = 0; i < dumpSize; i++) {
        int decimalValue = dumpArray[i];

        if (decimalValue < 32)
            print_space(dumpArray[i]);
        else
            printf("%c ", dumpArray[i]);
    }
    printf("\n");
}//print_group

void print_space(char specialChar) {
    //------------------------------------------------------//
    //Prints the character representation of special characters
    //based on the specialChar's decimal value.
    //Preconditions: specialChar is passed as parameter.
    //Postconditions: The character representation of the
    //special character is printed out based on the decimal
    //value of specialChar.
    //------------------------------------------------------//
    switch (specialChar) {
        case 9:
            printf("\\t ");
            break;
        case 10:
            printf("\\n ");
            break;
        case 11:
            printf("\\v ");
            break;
        case 8:
            printf("\\b ");
            break;
        case 12:
            printf("\\f ");
            break;
        default:
            printf("\\? ");
            break;
    }//end switch
}//end print_space
```