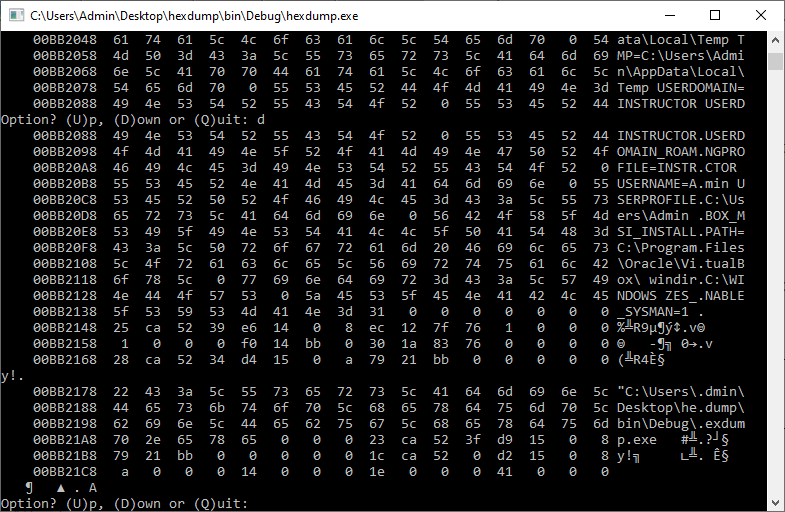
A Rust program is needed which creates a traditional hexdump display of data in the following format:



Each row has 16 bytes of data displayed in both hexadecimal and its ASCII/UTF-8 equivalent. These are displayed in blocks of 15 rows.

The data source may either be:

* RAM (stack/heap) or
* a text/binary file passed through the command line arguments by the user.

The user should be able to move up and down the data blocks or quit, through the use of a simple menu system.

C Code primer, if required:

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#define COLS 16

#define PRINTABLE 32

#define LASTUPCASE 90

#define BLOCKSIZE 320

#define DATABLOCK 120

typedef unsigned char byte;

int get\_option(void);

void display\_memory(byte \*, size\_t);

int main(void)

{

    int \* data;

    char option;

    //create sample data on the heap to browse/check endian-ness...

    data = malloc(sizeof(10)\*DATABLOCK);

    data[0] = 10;

    data[1] = 0x14; //hex, because we can

    data[2] = 036;  //octal, ditto

    data[3] = 65;

    data[4] = 513;

    //get us a pointer we can enumerate over

    byte \* starting = (byte \*)data;

    //show default "near" memory screen dump, including test data

    display\_memory(starting, BLOCKSIZE);

    do

    {

        option = get\_option();

option = option > LASTUPCASE ? option-PRINTABLE : option;

        switch(option)

        {

            case 'U': display\_memory(starting-=BLOCKSIZE, BLOCKSIZE);

                      break;

            case 'D': display\_memory(starting+=BLOCKSIZE, BLOCKSIZE);

                      break;

        }

    } while (option != 'Q');

    free(data);

    return 0;

}

//function to get process menu

int get\_option(void)

{

    printf("\rOption? (U)p, (D)own or (Q)uit: ");

    return getche();

}

//function to display memory - could be refactored, obviously...

void display\_memory(byte \* start, size\_t qty)

{

    printf("\n");

    //loop over desired block in column-based chunks

    for (byte \* loc = start; loc <= start+qty; loc+=COLS)

    {

        //LHS memory address

        printf("%12p ", loc);

        //MIDDLE hex columns

        for (size\_t i=0; i<COLS; i++)

        {

           printf(" %2x ", loc[i]);

        }

        //RHS ASCII block/row

        for (size\_t i=0; i<COLS; i++)

        {

           if (start[i] >= PRINTABLE)

           {

             printf("%c", loc[i]);

           }

           else

           {

             printf(".");

           }

        }

        printf("\n");

    }

}