CS2263 Lab 3

Stephen Cole

3553803

Exercise Zero

A screenshot of a cell phone

Description automatically generated

A close up of a sign

Description automatically generated

Exercise One

arithmetic.c

#include <stdio.h>

#include <stdlib.h>

int main (int argc ,char \* \* argv)

{

int arr1[] = {7, 2, 5, 3, 1, 6, -8, 16, 4};

char arr2[] = {'m', 'q', 'k', 'z', '%', '>'};

double arr3[] = {3.14, -2.718, 6.626, 0.529};

int len1 = sizeof(arr1) / sizeof(int);

int len2 = sizeof(arr2) / sizeof(char);

int len3 = sizeof(arr3) / sizeof(double);

printf("lengths = %d, %d, %d\n", len1, len2, len3);

int \* iptr = arr1;

char \* cptr = arr2;

double \* dptr = arr3;

printf("values = %p, %p, %p\n", iptr, cptr, dptr);

iptr ++;

cptr ++;

dptr ++;

printf("values = %p, %p, %p\n", iptr, cptr, dptr);

iptr ++;

cptr ++;

dptr ++;

printf("values = %p, %p, %p\n", iptr, cptr, dptr);

iptr ++;

cptr ++;

dptr ++;

printf("values = %p, %p, %p\n", iptr, cptr, dptr);

return EXIT\_SUCCESS;

}

A screenshot of a cell phone

Description automatically generated

Are the pointer variables incremented correctly? Show your calculations based on the memory addresses printed by your program.

Yes, they are. Integer size = 4 Bytes, Character = 1 Byte and Double = 8 Byte.

iptr: 0x7ffddc41ae20 + 4 = 0x7ffddc41ae24 + 4 = 0x7ffddc41ae28

cptr: 0x7ffddc41ae10 + 1 = 0x7ffddc41ae11 + 1 = 0x7ffddc41ae12

dptr: 0x7ffddc41adf0 + 8 = 0x7ffddc41adf8 + 8 = 0x7ffddc41ae00

Are the increments for different pointers the same? Explain why.

The increments for pointers differ based on their data type. Integer size = 4 Bytes, Character = 1 Byte and Double = 8 Byte. Based on the type of the array they will increment by the size of the data type they contain.

Exercise Two

loopbyaddress.c

#include<stdlib.h>

#include<stdio.h>

int main(void)

{

int arr[] = {10, 11, 12, 13, 14, 15, 16};

int size = 7;

int\* anchor = &arr[0];

int i;

int\* next = anchor;

int loops = 0;

for(i=0; i<=size; i++)

{

if(loops == 2)

break;

else if(i == size)

{

next = anchor;

i=-1;

loops++;

continue;

}

printf("%d %d %p %d\n", i, \*next, next, \*next);

next++;

}

return 0;

}

A close up of text on a black background

Description automatically generated

Push for Exercise One and Exercise Two

A screenshot of a cell phone

Description automatically generated

Exercise Three

arrindex.c

#include<stdlib.h>

#include<stdio.h>

int main(void)

{

int arr[] = {10, 11, 12, 13, 14, 15, 16};

int i;

for (i=0; i< sizeof(arr)/sizeof(arr[0]); i++)

printf("%d %d \n", i, arrindex( &arr[0], &arr[i]));

}

int arrindex(int \* p1, int \* p2)

{

int\* address = p2;

int i;

for(i=0; address >= p1; i++)

{

if(address == p1)

return i;

address--;

}

return 0;

}

A close up of a logo

Description automatically generated

Exercise Four

wrongindex.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char \* \* argv)

{

int x = -2;

int arr[] = {0, 1, 2, 3, 4};

int y = 15;

printf("& x = %p, & y = %p\n", & x, & y);

printf("& arr[0] = %p, & arr[4] = %p\n", & arr[0],

& arr[4]);

printf("x = %d, y = %d\n", x, y);

arr[-1] = 7;

arr[5] = -23;

printf("x = %d, y = %d\n", x, y);

arr[6] = 108;

printf("x = %d, y = %d\n", x, y);

arr[7] = -353;

printf("x = %d, y = %d\n", x, y);

printf("x %p, y %p\n", &x, &y);

int i;

for(i=-1; i<7; i++)

printf("Index %d: %p\n", i, &arr[i]);

return EXIT\_SUCCESS;

}

A close up of text on a black background

Description automatically generated

The results from the textbook are different because the array is indexed in a slightly different location. In my example the location of arr[-1] == y; therefore, y is set to the value intended for arr[-1] and the same for the address of arr[6] and x.

Push for Exercise Three and Four

A screenshot of a cell phone

Description automatically generated

Exercise Five

A screenshot of a cell phone

Description automatically generated