**CS241 Lecture 3 August 28, 2015 - Ed Karrels – Pointers and structures**

How do you declare a pointer? Give an example

What are the address-of and dereference operators?

How do you use a pointer to read/write some memory? Give an example

How can you find the size of a variable in memory?

How big are the basic data types? Are they the same on every computer?

How do you print out pointer values?

What is pointer arithmetic?

Why is it useful to have a pointer as a function argument?

What happens when you cast a pointer?

How is an integer stored in memory? What does little-endian mean?

Bonus question: how are floats and doubles stored in memory?

What is a void pointer?

What's the difference between a and b? Give an example of something you can do with a but not b

char a[] = "Hello";

char\* b = "Hello";

sizeof() returns the number of bytes. So using above code, what is sizeof(a) and sizeof(b) ?

What is sizeof(sizeof(a))?

Which of the following code is incorrect?

|  |  |
| --- | --- |
| int\* f1(int \*p) {  \*p = 42;  return p;  } | char\* f2() {  char p[] = "Hello";  return p;  } |
| char\* f3() {  char\* p = "Hello";  return p;  } | char\* f4() {  static char p[] = "Hello";  return p;  } |

Fix & Complete this function!

void mystrcpy(char\*dest, char\* src) { // void means no return value

while( \*src ) { dest = src; src ++; dest++; }

}

How do you allocate memory on the heap? How to you deallocate it?

Fix & Complete this function!

char\* mystrdup(char\*source) {

char\* p = \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ ( sizeof(source) );

strcpy(source, p);

return p;

}

Give an example of buffer overflow:

What is double free? How can you avoid?

What is a dangling pointer? How do you avoid?

How do you declare a stuct in C?

How is a C struct different from a C++ class?

What is 'typedef' and how do you use it?

**Demo code A**

#include <stdio.h>

int main() {

char str[] = "hello";

char \*p = &str;

printf("p = %s\n", p);

return 0;

}

**Demo code B**

#include <stdio.h>

int main() {

printf("sizeof(char) = %ld\n", sizeof(char));

printf("sizeof(int) = %d\n", (int)sizeof(int));

printf("sizeof(long) = %d\n", (int)sizeof(long));

printf("sizeof(float) = %d\n", (int)sizeof(float));

printf("sizeof(double) = %d\n", (int)sizeof(double));

printf("sizeof(char\*) = %d\n", (int)sizeof(char\*));

printf("sizeof(void\*) = %d\n", (int)sizeof(void\*));

printf("sizeof(double\*) = %d\n", (int)sizeof(double\*));

/\*

int i;

char \*cp;

int \*ip;

ip = &i;

cp = (char\*) &i;

// printf("&p = %p, &q = %p\n", &p, &q);

printf("ip = %p, cp = %p\n", ip, cp);

ip++;

cp++;

printf("ip = %p, cp = %p\n", ip, cp);

\*/

return 0;

}

**Demo code C**

#include <stdio.h>

#include <stdlib.h>

int main() {

int array[3];

int length = sizeof(array) / sizeof(int);

for (int i=0; i < 3; i++) {

array[i] = rand();

}

int \*p = array;

int \*end = array + length;

while (p < end) {

printf("array[%d] = %d\n", (int)(p-array), \*p );

p++;

}

return 0;

}

**Demo code D**

#include <stdio.h>

void splitFloat(double input, int \*integerPart, double \*fractionPart) {

\*integerPart = (int)input;

\*fractionPart = input - \*integerPart;

// note: this is not a robust implementation

}

int main() {

int i;

double d;

splitFloat(1234.5678, &i, &d);

printf("%d and %f\n", i, d);

splitFloat(2.7182, &i, &d);

printf("%d and %f\n", i, d);

return 0;

}

**Demo code E**

#include <stdio.h>

int main() {

char \*s = "ABCD";

// char a[] = "ABCD";

// int \*p = (int\*) s;

void \*v = s;

// printf("v+1 is %p, v is %p\n", v+1, v);

printf("\*p = %d or %x\n", \*p, \*p);

return 0;

}