CS35L-5

Week 4 Lec1

Data Types

- Integers: char, int, short, long, long long
- Unsigned Integers: unsigned char, unsigned int ..
- Floating point numbers: float, double
- Structures
- ~~Boolean~~

```
#define BOOL char
#define FALSE 0
#define TRUE 1
```

Pointers

- Variables that store memory address
- Declaration: <base_type> * pointerName

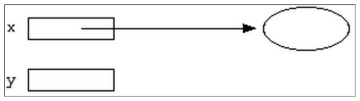
```
int *ptr;      //declare ptr as a pointer to int
int var = 77;    // define an int variable
ptr = &var;  // let ptr point to the variable
printf("%d", *ptr);  //Dereferencing
*ptr = 88;
```

Pointer Example

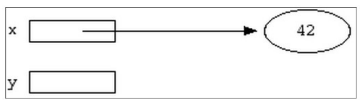
int *x;

int *y;

int var;
x = &var;

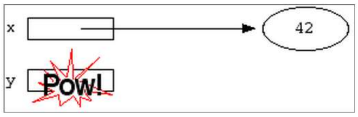


*x = 42;

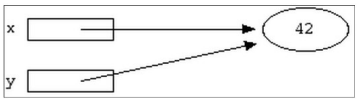


Pointer Example

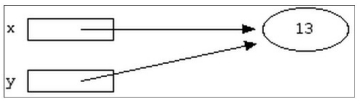
*y = 13;



y = x;



*x = 13; or
*y = 13;



Pointers to Pointers

char c = 'A' *cPtr = &c **cPtrPtr = &cPtr

More Data

- Strings - Character array/ Pointer to char
  char name[] = "John Smith";
  char name[11] = "John Smith";
  char *name = "John Smith";
  printf("%s",name);
- Arrays - Contiguous memory locations
  double *p;
  double balance[10];
  p=balance;
  *p, *(p+2), balance[0], balance[2], *(balance+2)
  //All dereferencing

Pointer Arithmetic

increase/decrease a pointer's value by the number of bytes in its data size

```
int intarray[5] = {10,20,30,40,50};
int * intPtr = intarray;

for(int i=0;i<5;i++){
   printf("%d",intarray[i]);
   printf("%d",*(intarray+i));
   printf("%d\n",*intPtr);
   intPtr++;
}
```

Pointers to Functions

- Also known as: function pointers or functors
- Goal: write a sorting function(s)
  - Has to work for ascending and descending sorting order + other
- How?
  - Write multiple functions
  - Provide a flag as an argument to the function
  - Polymorphism and virtual functions
  - Use function pointers!!

Function Pointer syntax

```
int addInt(int n, int m) {
   return n+m;
}

int (*functionPtr)(int,int);

functionPtr = &addInt;

int sum = (*functionPtr)(2, 3);

int sum = functionPtr(2, 3);
```

qsort Example

```
void qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const void*))
```

```
#include <stdio.h>
#include <stdlib.h>

int compare (const void * a, const void * b)
{
        return ( *(int*)a - *(int*)b );
}

int main ()
{
        int values[] = { 40, 10, 100, 90, 20, 25 };
        qsort (values, 6, sizeof(int), compare);
        int n;
        for (n = 0; n < 6; n++)
                printf ("%d ",values[n]);
        return 0;
}
```

Structs

- No classes in C
- Used to package related data (variables of different types) together
- Single name is convenient

```
struct Student {
        char name[64];
        char UID[10];
        int age;
        int year;
};

struct Student s;
```

```
typedef struct {
        char name[64];
        char UID[10];
        int age;
        int year;
} Student;

Student s;
```

*Student * ptr = &s; ptr -> age; s.age;*

C structs vs. C++ classes

- C structs cannot have member functions
- There's no such thing as access specifiers in C
- C structs don't have constructors defined for them
- C++ classes can have member functions
- C++ class members have access specifiers and are **private** by default
- C++ classes must have at least a default constructor

Dynamic Memory

- Memory that is allocated at runtime
- Allocated on the heap

void *malloc (size_t size);
- Allocates *size* bytes and returns a pointer to the allocated memory

void *realloc (void *ptr, size_t size);
- Changes the size of the memory block pointed to by *ptr* to *size* bytes

void free (void *ptr);
- Frees the block of memory pointed to by *ptr*

Reading/Writing Characters

- **int getchar();**
  - Returns the next character from stdin
- **int putchar(int character);**
  - Writes a character to the current position in stdout

Formatted I/O

- int fprintf(FILE * fp, const char * format, …);
- int fscanf(FILE * fp, const char * format, …);
  - FILE *fp can be either:
    - A file pointer
    - stdin, stdout, or stderr
  - The format string
    - int score = 120; char player[] = "Mary";
    - printf(**"%s has %d points.\n", player, score**);