CS 35L

LAB 8,

TA: Sucharitha Prabhakar

EMAIL ID: prabhakarsucharitha@gmail.com

Outline

Introduction to C

Basic data types

- int
    - Holds integer numbers
    - Usually 4 bytes
- float
    - Holds floating point numbers
    - Usually 4 bytes
- double
    - Holds higher-precision floating point numbers
    - Usually 8 bytes (double the size of a float)
- char
    - Holds a byte of data, characters
- Void

Pretty much like C++ basic data types, but NO bool before C99

Pointers

Variables that store memory addresses

Declaration

<variable_type> *<name>;

```
int *ptr;       //declare ptr as a pointer to int

int var = 77;          // define an int variable

ptr = &var;     // let ptr point to the variable var
```

Dereferencing Pointers

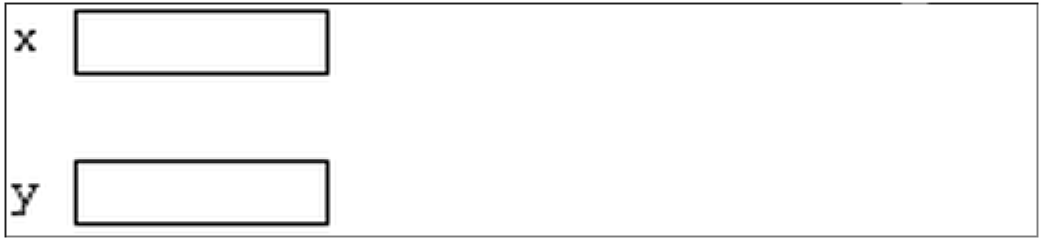
Accessing the value that the pointer points to

Example:

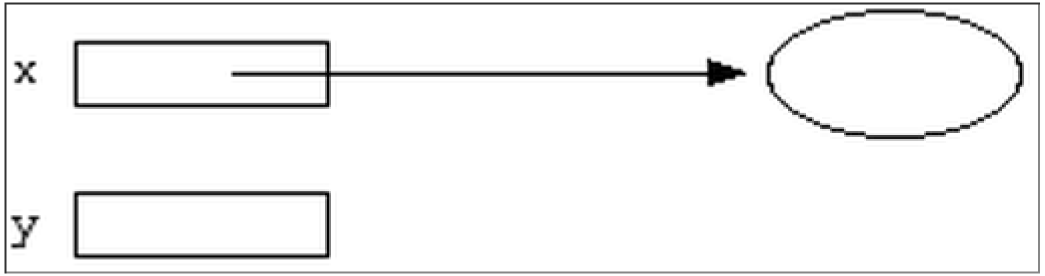
```
double x, *ptr;
 ptr = &x;          // let ptr point to x
*ptr = 7.8;         // assign the value 7.8 to x
```

Pointer Example

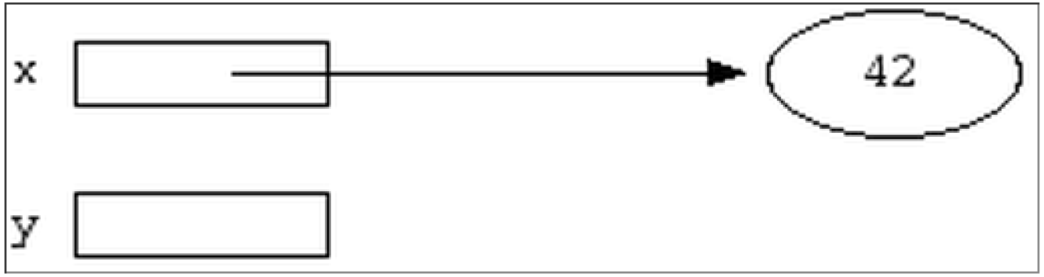
int *x; int *y;



int var; x = &var;

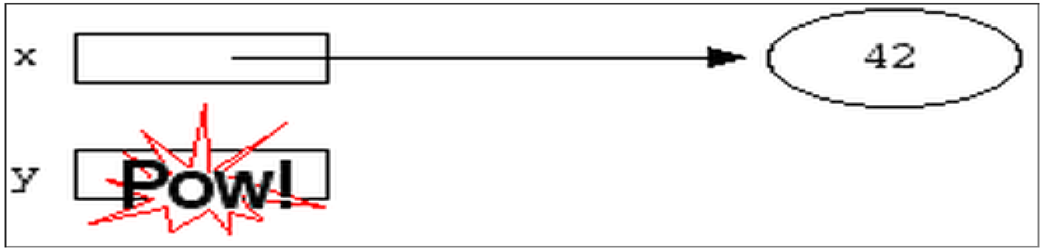


*x = 42;

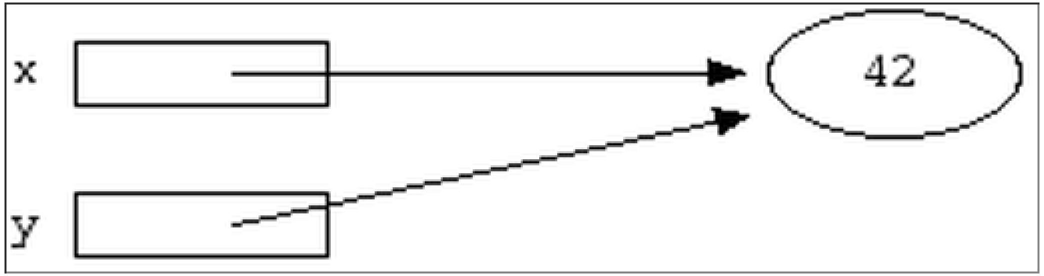


Pointer Example

*y = 13;

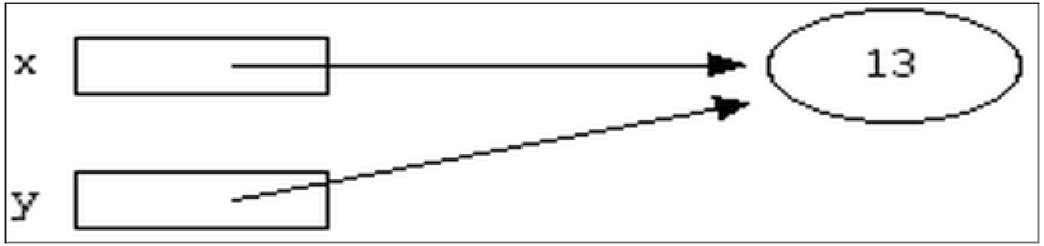


y = x;



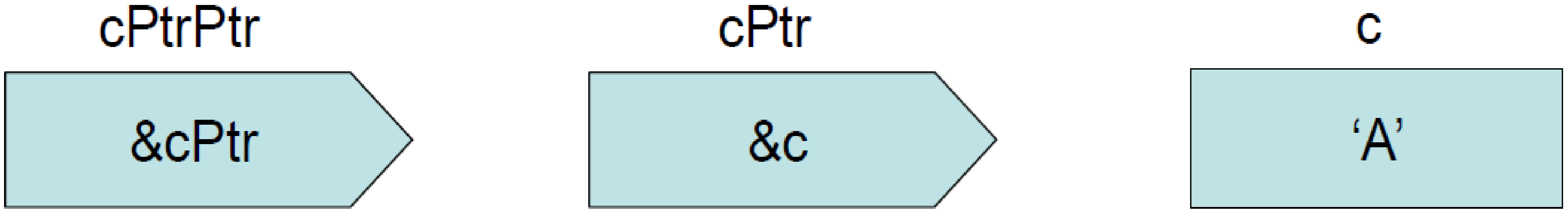
*x = 13; or

*y = 13;



Pointers to Pointers

char c = ‘A’ ;   *cPtr = &c;   **cPtrPtr = &cPtr;



Pointers to Functions

Also known as: function pointers or functors

Declaration: <function return type>(*<Pointer_name>)(function argument list)

Example: double (*p2f)(double, char, int)

Pointers to Functions - Example

Goal: write a sorting function

Has to work for ascending and descending sorting order

How?

- Write multiple functions
- Provide a flag as an argument to the function
- Use function pointers!!

Pointers to Functions - Example

User can pass in a function to the sort function

Declaration

```
double (*func_ptr) (double, double);

func_ptr = pow;  // func_ptr points to pow()
```

Usage

```
// Call the function referenced by func_ptr

double result = (*func_ptr)( 1.5, 2.0 );

// The same function call

 double result = func_ptr( 1.5, 2.0 );
```

Pointers to Functions - qsort

```
#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;
```

C structs vs. C++ classes

C structs cannot have member functions	C++ classes can have member functions
There’s no such thing as access specifiers in C	C++ class members have access specifiers and are private by default
C structs don’t have constructors defined for them	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 and Writing Chars

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);

Homework 5

Write a C program called sfrob

- Reads stdin byte-by-byte (getchar)
    - Consists of records that are space-delimited
- Each byte is frobnicated (XOR with dec 42)
    - Sort records without decoding (qsort, frobcmp)
    - Output result in frobnicated encoding to stdout (putchar)
- Error checking (fprintf)
- Dynamic memory allocation (malloc, realloc, free)

Homework 5

Input: printf 'sybjre obl’

```
$ printf 'sybjre obl ' | ./sfrob
```

Read the records: sybjre, obl

Compare records using frobcmp function

Use frobcmp as compare function in qsort

Output: obl sybjre