ECE220 Lab5

Brain Teaser - Array Initialization

What do these initialize to?

- int arr1[10];
- int arr2[10] = { 0 };
- int arr3[10] = { 1, 2, 3 };
- int arr4[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

Initialization: *type* name[len] = $\{0\}$ or $\{v_0, v_1\}$ or $\{v_0, v_1, \ldots, v_{len-1}\}$

Given int arr[3], which of these are valid?

- int x = arr[1];
- int x = arr[-1];
- int x = arr[3];
- int x = 1[arr];
- int x = *(arr + 1);

sizeof() Operator

sizeof() – returns the size of a variables in bytes

Examples

- sizeof(char) = 1
- sizeof(int) = 2 or 4 depending on system
- sizeof(long) = 4
- sizeof(float) = 4
- sizeof(double) = 8
- sizeof(char[10]) = 10

Computing size of an array

- int arr = {1, 2, 3, … }
- size_t n = sizeof(arr) / sizeof(arr[0]);

Structure of C Files

File types:

- header file (*.h) – contains function declarations to be shared between source files
- source files (*.c) – defines functions declared in header files and uses them

Example

add.h

```
#ifndef ADD_H
#define ADD_H

int add(int a, int b);

#endif /* ADD_H */
```

add.c

```
#include <stdio.h>
#include "add.h"

int add(int a, int b)
{
        int sum = a + b;
        printf("Sum of %d and %d is %d\n", a, b, sum);
        return sum;
}
```

main.c

```
#include "add.h"

int main()
{
        add(2, 3);
        return 0;
}
```

Header and Source Files

```
#ifndef ADD_H
#define ADD_H

int add(int a, int b);

#endif /* ADD_H */
```

```
#include <stdio.h>
#include "add.h"

int add(int a, int b)
{
        int sum = a + b;
        printf("Sum of %d and %d is %d\n", a, b, sum);
        return sum;
}
```

Header files
- Header guards
  - If constant ADD_H not defined, then define it
  - Prevents a header file from being included multiple times
- Function declarations
  - Start of a function but ends in with a semicolon
  - Tells source files that functions exist, in this case the ‘add’ function

Source files
- Includes the header file
- Defines function included in the header file with actual code

Corresponding names
- <filename>.c and <filename>.h convention

Compiling C Programs

Includes

- #include <filename> - searches for system wide header files
- #include “” – searches for local (same directory) header files
- #include literally copies contents of filename.h into location

Compilation

- Object compilation
  - Every *.c file is individually compiled into a separate object files *.o
  - Source files know included functions exist but don’t know where
- Linking
  - Looks at indexes of object files and tries to solve dependencies between them
  - Generates a single executable program

```
#include "add.h"

int main()
{
        add(2, 3);
        return 0;
}
```

Lab Makefile

Compilation

- Object compilation at $(OBJECTS)
- Linking at $(TARGET)

Make commands

- *all* (Default) – creates required directories and compiles code
- *dirs* – creates *obj* and *bin* directories
- *test* – runs tests to test your program
- *clean* – removes files generated by *all*

```
#Compiler
CC = gcc
CFLAGS = -g -Wall -std=c99

#Directories
SRC_DIR = src
OBJ_DIR = obj
BIN_DIR = bin
TST_DIR = test

#Files
SOURCES  := $(wildcard $(SRC_DIR)/*.c)
INCLUDES := $(wildcard $(SRC_DIR)/.*h)
OBJECTS  := $(SOURCES:$(SRC_DIR)/%.c=$(OBJ_DIR)/%.o)

#Binaries
EXEC = dup_unique
TARGET := $(BIN_DIR)/$(EXEC)

.PHONY: all
all: $(TARGET)

$(TARGET): dirs $(OBJECTS)
        $(CC) $(CFLAGS) -o $@ $(OBJECTS)

$(OBJECTS): $(OBJ_DIR)/%.o: $(SRC_DIR)/%.c
        $(CC) $(CFLAGS) -c -o $@ $<

.PHONY: dirs
dirs:
        mkdir -p $(OBJ_DIR)
        mkdir -p $(BIN_DIR)

.PHONY: test
test:
        bash ./$(TST_DIR)/runtests.sh

.PHONY: clean
clean:
        rm -rf $(BIN_DIR)
rm -rf $(OBJ_DIR)
```

Lab Testing

LAB 4

```
[potok@xps-debian: tests]$ cat runtests.sh
#!/bin/bash

score=0
out=("goldout1" "goldout2" "goldout3")
tests=("test1" "test2" "test3")

end=$((${#tests[@]}-1))
for i in $(seq 0 1 $end); do
  ./bin/dice < tests/${tests[i]} > output/${tests[i]}
  diff -wb tests/${out[i]} output/${tests[i]} &> /dev/null
  res=$?

  if [[ $res -eq 0 ]]; then
    echo "Test" $((i + 1))": passed"
    ((score += 1))
  else
    echo "Test" $((i + 1))": failed"
  fi
done

echo "Score: "$score"/"${#tests[@]}
[potok@xps-debian: tests]$
```

```
[potok@xps-debian: gold]$ make
mkdir -p obj
mkdir -p bin
mkdir -p output
gcc -g -Wall -std=c99 -c -o obj/dice.o src/dice.c
gcc -g -Wall -std=c99 -c -o obj/main.o src/main.c
gcc -g -Wall -std=c99 -o bin/dice obj/dice.o obj/main.o
[potok@xps-debian: gold]$ make test
zsh: correct 'test' to 'tests' [nyae]? n
bash ./tests/runtests.sh
Test 1: passed
Test 2: passed
Test 3: passed
Score: 3/3
[potok@xps-debian: gold]$ diff output/test1 tests/goldout1
[potok@xps-debian: gold]$
```

LAB 5

```
[potok@xps-debian: tests]$ cat runtests.sh
#!/bin/bash

score=0
tests=("3 3 3" "9 17 14" "0 0 0")

end=$((${#tests[@]}-1))
for i in $(seq 0 1 $end); do
  ./bin/mat_mult ${tests[i]} | grep -q 'Correct!'
  res=$?

  if [[ $res -eq 0 ]]; then
    echo "Test" $((i + 1))": passed"
    ((score += 1))
  else
    echo "Test" $((i + 1))": failed"
  fi
done

echo "Score: "$score"/"${#tests[@]}
```

MP5 – Conway's Game of Life

Conway's Game of Life
- Cellular automaton whose evolution is determined by its initial state

Cell states
- Alive – black
- Dead – white

Neighborhood
- Adjacent 8 cells around the central cell

Rules
- Any live cell with 2 or less neighbors dies (under population)
- Any live cell with 2 or 3 neighbors lives
- Any live cell with 3 or more neighbors dies (over population)
- Any dead cell with 3 live neighbors is reborn (reproduction)

MP5 – Game Board

Game board
- 2D array represented as a 1D array

Array access of green element
- 2D – array[1][2]
- 1D – array[6]

How to translate between 2D and 1D arrays?
- $idx = row * WIDTH + col$

Updating board game
- Iterate through every cell in the game board
- Given a central cell, access eight neighbors around it
  - Count how many are alive/dead
- Determine if central cell should live/die according to the Game of Life rules

2D array

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

1D array

$$[0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0]$$

Neighborhood

$$\begin{bmatrix} N & N & N \\ N & C & N \\ N & N & N \end{bmatrix}$$

Lab5 – Matrix Multiplication

Multiply two matrices together

- $(m, k) \times (k, n) = (m, n)$
- $\begin{bmatrix} 2 & 4 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} -1 & 2 & -2 \\ 5 & 3 & -1 \end{bmatrix} = \begin{bmatrix} 18 & 16 & -8 \\ 16 & 7 & -1 \end{bmatrix}$

Computing every cell in the output matrix

- $C_{i,j} = \sum_{l=0}^{k-1} A_{i,l} * B_{l,j}$

All matrices represented as 1D arrays

- $idx = row * WIDTH + col$ equation to access correct elements