Introduction to Computer Science
W 1113 – Lab (C)
Lab5

Suhit Gupta
2/26/04

Questions about the previous lab

2

Questions about HW2

3

Recap from Lab 3

- Math operators
- Arrays (assignment and reference)
- Strings
  - string manipulation
  - fgets
  - sscanf

4

Recap from Lab 4

- Function prototypes
- Conditional statements
  - if
  - switch
- Loops
  - while
  - do while
  - for

5

Quick quiz…

- What does the following do in a for loop
  - && or ||
- What are double and long?

6

Function prototypes revisited

- Usually, you declare variables before you can use them
  - similar with functions
  - however, you can
    - declare a function prototype at the beginning of the program
    - define the actual function workings later on
- Example
  - int add (int a, int b);
- This is important in HW2

7

Function prototypes – code I

```
#include <stdio.h>

int add (int first_number, int second_number) {
	int total;

	total = first_number + second_number;
	return total;
}

int main(int argc, char *argv[]) {
	int c, x, y;
	x=atoi(argv[1]);
	y=atoi(argv[2]);
	c=add(x, y);
	printf("The total of %d and %d is %d\n", x, y, c);
}
```

8

Function prototypes – code II

```
#include <stdio.h>

int add (int a, int b);

int main(int argc, char *argv[]) {
	int c, x, y;
	x=atoi(argv[1]);
	y=atoi(argv[2]);
	c=add(x, y);
	printf("The total of %d and %d is %d\n", x, y, c);
}

int add (int first_number, int second_number) {
	int total;

	total = first_number + second_number;
	return total;
}
```

9

Some more examples

```
#include <stdio.h>

//defining all my function prototypes
int add2 (int a, int b);
int sub2 (int a, int b);
int mult2 (int a, int b);
float div (int a, int b);

int main(int argc, char *argv[]) {
    //defining all my variables
    int x, y, addanswer, minusanswer, multanswer, a, z;
    float divanswer;
    //reading in all the input
    x=atoi(argv[1]);
    y=atoi(argv[2]);
    //performing calculations and printing the result
    addanswer=add2(x, y);
    minusanswer=sub2(x,y);
    multanswer=mult2(x, y);
    divanswer=div(x,y);
    printf("The respective calculations of %d and %d are %d, %d, %d and %f\n", x, y, addanswer,
minusanswer, multanswer, divanswer);
}

//The add function
int add2 (int first_number, int second_number) {
    int total;

    total = first_number + second_number;
    return total;
}

//The subtraction function
int sub2 (int first_number, int second_number) {
    int total;

    total = first_number - second_number;
    return total;
}

//The multiplication function
int mult2 (int first_number, int second_number) {
    int total;

    total = first_number * second_number;
    return total;
}

//The division function - note that this one returns a float
float div (int first_number, int second_number) {
    float total;

    total = (float) first_number /(float) second_number;
    return total;
}
```

10

Here is a problem – use functions

- Brainstorming (real world example)
  - Planning your trip to Europe
  - Changing currency during your Eurotrip
  - Booking Flights
  - Booking Hotel Room and/or Youth Hostels
  - Sightseeing
  - Look up the weather
- What are the different methods?

11

Conditionals revisited

- Conditional statements
  - if
  - switch

12

Conditionals

- Conditional statements
  - if
    - need to know <, >, ==, !=, <=, >=
    - &&, ||
    - usage: *if (expr) {stmt…}*
      else if (expr) {stmt…}
      else {stmt}
    - when do you not need {}
  - if followed by another if

```
if (something) do something;
if (something else) do something else;
```

  - The default case is the final *else*
  - Correctness
    - if (strcmp(string1, string2)) do something?
    - if (strcmp(string1, string2)==0) do something?

13

Conditionals II

- Switch

```
switch (val) {
    case 1:
        do some work;
        break;
    case 2:
        do some work;          // you don't have to necessarily have
        break;                 // stuff here
    case 3:
        do some work;
        break;
    default:                   //if needed
        do some work;
        break;
}
```

- What is the break statement?
- What happens if you don't use break?

14

Loops

- Iteration/loops
  - While
  - For
  - Do while
- Difference between conditionals and loops

15

Loops II

- While
  - usage:
    - *while (cond) {stmt…}*
  - break;
  - continue;
- code

```
while(current_number<100) {
    do something;                    //what is wrong
}
```

16

Loops II

- While
  - usage:
    - *while (cond) {stmt…}*
  - break;
  - continue;
- code

```
while(current_number<100) {
    do something;                    //what is wrong
    i++; // or i-- as the case may be
}
```

17

Loops III

- Do while
  - usage:

```
do {
    blah;
} while (i>0);
```

  - Again, remember that the value of ‘i’ needs to be changed

18

Loops IV

- For
  - usage:
    - for (initial statement ; condition ; iteration statement) {
      do something here;
      }
  - There is other acceptable syntax (sort of)
- BTW, this is where the ++i and i++ becomes relevant and useful
- Everything in for can be done in a while
  - Think about it

19

Loops V

- The comma operator
  - Things are evaluated from left to right
- for (sum=0, i=1; i<=n; ++i)
  sum += i;
- for (sum=0, i=1; i<=n; sum += i, ++i)
  ;
- for (sum=0, i=1; i<=n; ++i, sum += i)
  ; *// this may give wrong results as i is*
  // incremented before added to sum

20

Loops VI

- Why can we use the ; just like that
- Infinite loops – beware
  - while (1) { …}
  - for (; ;) {…}
    - Use it at your own risk (system administrator may kill ;-))
    - Use it instead of running your program again and again

21

What does the following do?

```
for (i = 1; i <= 10; ++i )
        ;
sum += i;
```

22

Back to the Europe Trip example

- Now that we know loops, how would we use them to call our methods nicely

23

Assignment

- Read Ch. 8 and 9 from the Practical C Programming book
- Start reading Ch. 7
- **HW2**
  - Due soon.

24