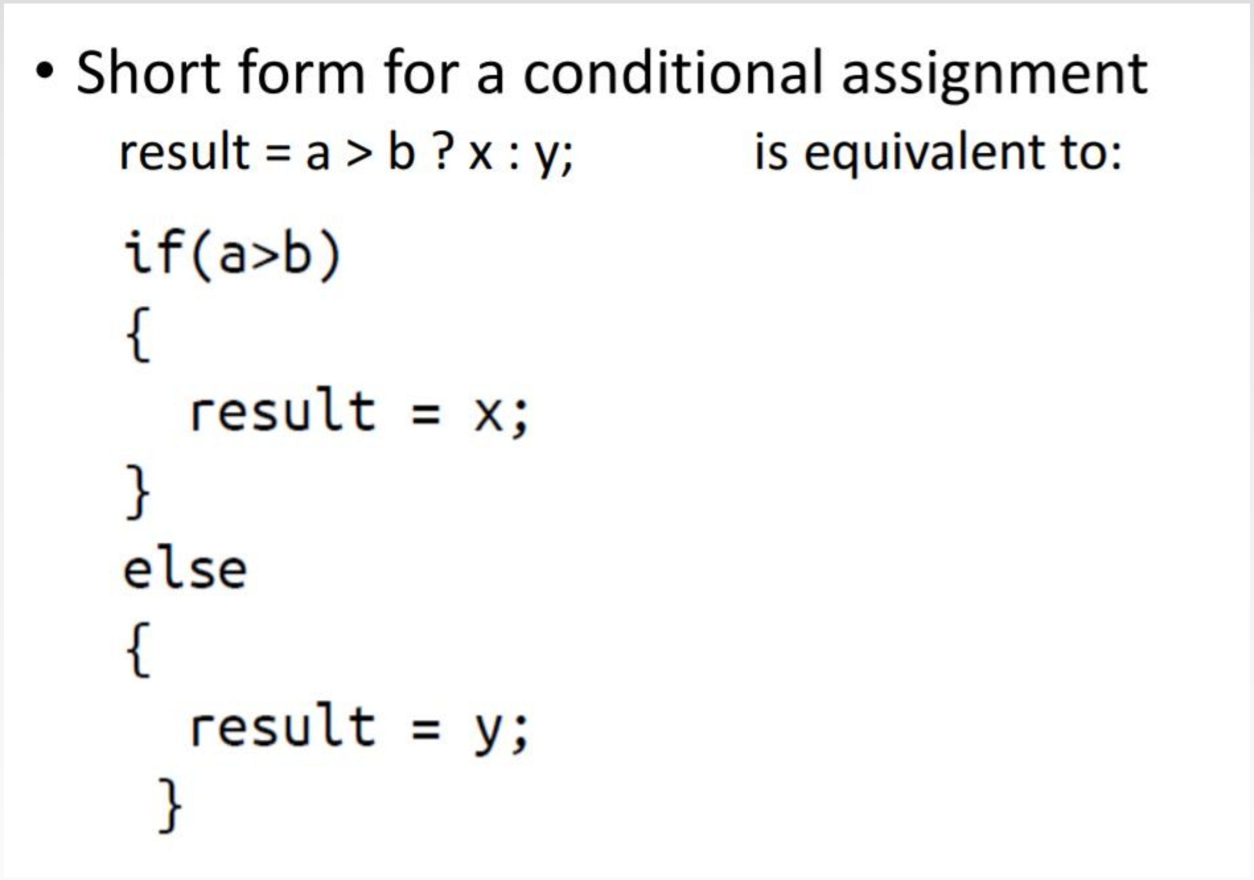
CS35L – Spring 2019

Slide set:	4.2
Slide topics:	C programming (continued)
Assignment:	4

Ternary Operator



- Short form for a conditional assignment

  result = a > b ? x : y;   is equivalent to:

```
if(a>b)
{
  result = x;
}
else
{
  result = y;
 }
```

Pointers to Functions

A pointer that points to a 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);

Function Pointers

Variable which stores address to a function’s executable code in memory.

```
#include <stdio.h>
void fun(int a)
{
	printf("Value of a is %d\n", a);
}
int main()
{
void (*fun_ptr)(int) = &fun;
(*fun_ptr)(10);
return 0;
}
```

Qsort Example

void qsort (void* base, size_t num, size_t size, int (*compar)(const void*,const void*));

Return value meaning for comparator function:

< 0	The element pointed by p1 goes before the element pointed by p2
= 0	The element pointed by p1 is equivalent to the element pointed by p2
> 0	The element pointed by p1 goes after the element pointed by p2

```
#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

typedef Declarations

- Easy way to use types with complex names

```
typedef struct { double x, y; } Point_t;

typedef struct
{
    Point_t top_left;
    Point_t bottom_right;
} Rectangle_t;
```

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*

Initializing array using Malloc

```
int *arr = malloc (sizeof (int) * n); /* n is the length of the array */
int i;

for (i=0; i<n; i++)
{
  arr[i] = 0;
}
```

Reading/Writing Characters

- **int getchar();**
  - Returns the next character from stdin. EOF when input ends or error encountered.
- **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”;
    - fp = fopen(“file.txt”, “w+”)
    - fprintf(fp, **“%s has %d points.\n”, player, score**);

Task 1

Write a program in C to take in as input the number of rows and columns of a matrix.

(Do the following with and without pointer arithmetic)

1. Dynamically allocate memory into this matrix
2. Reset the matrix (set all its elements to 0) – use an inbuilt C function
3. Print the matrix in a proper format (r rows and c columns)

Task 2

Write a program in C to take in as input the number of students in a class, their scores (on 100) in the 5 courses that are offered and any other relevant details (use appropriate prompts for each input).

1. Find the mean score of the class per course
2. Output the name of the topper of the class
3. Print the Dean’s list (just Student names and UIDs of 3 top-scoring students in the class)

Task 3

/*Using structures to calculate the area of a rectangle*/

Create two structs for Rectangle and Point.

Calculate the area of the rectangle using the given coordinates (top left and bottom right)

Use the below structure:

```
typedef struct {
   Point topLeft;   /* top left point of rectangle */
   Point botRight;  /* bottom right point of rectangle */
} Rectangle;
```

Task 4

Write a C program using getchar() and putchar() which continuously takes user input and prints it on the screen. This should keep on happening till the user inputs a string containing ‘#’ and Enters.

Hint: use while(getchar() != #)

Task 5

Write the following line in a file called file.txt

The value stored is 100

Use fscanf to read the value 100 from file.txt and store it in a variable <var>.

Then write this value to another file file1.txt “Value read is <var>” using fprintf

Lab 4

- Download old version of coreutils with buggy ls program
  - Untar, configure, make
- Bug: ls -t mishandles files whose time stamps are very far in the past. It seems to act as if they are in the future

```
$ tmp=$(mktemp -d)
$ cd $tmp
$ touch -d ‘1918-11-11 11:00 GMT’ wwi-armistice
$ touch now
$ sleep 1
$ touch now1
$ cd <your install-dir>/bin
$ ./ls –lt $tmp
```

Output:

```
-rw-r--r-- 1 eggert eggert 0 Nov 11 1918 wwi-armistice
-rw-r--r-- 1 eggert eggert 0 Feb 5 15:57 now1
-rw-r--r-- 1 eggert eggert 0 Feb 5 15:57 now
```

Goal: Fix the Bug

- **Reproduce the Bug**
  - Follow steps on lab web page

- **Simplify input**
  - Run ls with –l and –t options only

- **Debug**
  - Use gdb to figure out what’s wrong
  - $ gdb ./ls
  - (gdb) run –lt wwi-armistice now now1

  (run from the directory where the compiled ls lives)

- **Patch**
  - Construct a patch “lab4.diff” containing your fix
  - It should contain a ChangeLog entry followed by the output of diff -u

Lab Hints

- Use “info functions” to look for relevant starting point
- Use “info locals” to check values of local variables
- Compiler optimizations: -O2 -> -O0
  - ./configure CFLAGS="…-O0“
  - Or, during make: make CFLAGS=‘-g –O0’

Homework 4

- Write a C program called *sfrob*
  - Reads stdin byte-by-byte **(getchar)**
    - Consists of records that are newline-delimited
  - Each byte is frobnicated (XOR with dec 42)
    - Sort records without decoding (**qsort, frobcmp**)
    - Output result in frobnicated encoding to stdout **(putchar)**
  - Dynamic memory allocation (**malloc, realloc, free**)

Example

- Input: printf 'sybjre obl’
  - $ printf 'sybjre obl**\n**' | ./sfrob
- Read the records: sybjre, obl
- Compare records using *frobcmp* function
- Use *frobcmp* as compare function in *qsort*
- Output: obl

  sybjre

Homework Hints

- Array of pointers to char arrays to store strings (char ** arr)
- Use the right cast while passing frobcmp to qsort
  - cast from void * to char ** and then dereference because frobcmp takes a char *
- Use realloc to reallocate memory for every string and the array of strings itself, dynamically
- Use *exit*, not *return* when exiting with error