CS35L Software Construction Laboratory

Lab 5: Sneha Shankar

Week 4; Lecture 2

Randline.py

- Run it
  - ./randline.py –n 3 filename (need execute permission)
  - python randline.py –n 3 filename (no execute permission)
- randline.py has 3 command-line arguments:
  - n: specifies the number of lines to write
    - **option**
  - 3: number of lines
    - **option argument** to n
  - filename: file to choose lines from
    - **argument** to script
- Output: 3 random lines from the input file
- Python 3 is installed in /usr/local/cs/bin
  - export PATH=/usr/local/cs/bin:$PATH

Python Walk-Through

```
#!/usr/bin/python
import random, sys
from optparse import OptionParser
class randline:
  def __init__(self, filename):
  f = open (filename, 'r')
  self.lines = f.readlines()
  f.close ()
  def chooseline(self):
  return random.choice(self.lines)

def main():
        version_msg = "%prog 2.0"
        usage_msg = """%prog [OPTION]...
FILE Output randomly selected lines from
FILE."""
```

Tells the shell which interpreter to use

Import statements, similar to include statements

Import OptionParser class from optparse module

The beginning of the class statement: randline

The constructor

Creates a file handle

Reads the file into a list of strings called lines

Close the file

The beginning of a function belonging to randline

Randomly select a number between 0 and the size of lines and returns the line corresponding to the randomly selected number

The beginning of main function

Python Walk-Through

```
parser = OptionParser(version=version_msg,
usage=usage_msg) parser.add_option("-n", "--numlines",
action="store", dest="numlines",   default=1,
help="output NUMLINES   lines (default 1)")
options, args = parser.parse_args(sys.argv[1:])
try:
        numlines = int(options.numlines)
except:
        parser.error("invalid NUMLINES: {0}".
format(options.numlines))
if numlines < 0:
        parser.error("negative count: {0}".
format(numlines))
if len(args) != 1:
        parser.error("wrong number of operands")
input_file = args[0]
try:
        generator = randline(input_file)
        for index in range(numlines):
        sys.stdout.write(generator.chooseline())
except IOError as (errno, strerror):
        parser.error("I/O error({0}): {1}".
format(errno, strerror))
if __name__ == "__main__":
        main()
```

Creates OptionParser instance

Start defining options, action "store" tells optparse to take next argument and store to the right destination which is "numlines". Set the default value of "numlines" to 1 and help message.

options: an object containing all option args

args: list of positional args leftover after parsing options

Try block

get numline from options and convert to integer

Exception handling

error message if numlines is not integer type, replace {0} w/ input

If numlines is negative

error message

If length of args is not 1 (no file name or more than one file name)

error message

Assign the first and only argument to variable input_file

Try block

instantiate randline object with parameter input_file

for loop, iterate from 0 to numlines – 1

print the randomly chosen line

Exception handling

error message in the format of "I/O error (errno):strerror

In order to make the Python file a standalone program

comm.py

- Support all options for comm
  - 1, -2, -3 and combinations
  - Extra option –u for comparing unsorted files
- Support all type of arguments
  - File names and
  - – for stdin
- If you are unsure of how something should be output, run a test using existing comm utility!
  - Create your own test inputs
- Comm C source code :
  - comm C source code
  - This will give you an idea of the logic behind the operation that comm executes
- Python OptionParser link :
  - Python OptionParser
  - How to add your own options to the parser

comm.py...

- Assume C locale for sorting purpose
- Port comm.py to Python 3
- Change usage message to describe script behaviour
- Follow the instructions on Piazza:
  https://piazza.com/class/jc5z73cpj8rtt?cid=194

Supplement resources

- Python tutorial
  - https://docs.python.org/3.5/tutorial/
- Python Examples
  - https://www.programiz.com/python-programming/examples

Homework 3 Hints

- The comm options -123 are Boolean
  - –Which action should you use?
- Q4: Python 3 vs. Python 2
  - –Look up “automatic tuple unpacking”
- Python 3 is installed in /usr/local/cs/bin
  - –export PATH=/usr/local/cs/bin:$PATH

C Programming

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**

A simple C Program

```
#include <stdio.h>

int main() {
        printf("Hello World\n");
        getchar();
        return 0;
}
```

Format Specifiers in C

- Defines the type of data to be printed on standard output - printf("%d", 4); //%d is integer

Data Types Revised

Data Type	Range	Bytes	Format Specifiers
char	-128 to 127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32,768 to 32,767	2	%d
short unsigned int	0 to 65,535	2	%u
signed int	-32768 to 32767	4	%d
unsigned int	0 to 65535	4	%u
long signed int	-2147483648 to 2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to 3.4e38	4	%f
double	-1.7e308 to 1.7e308	8	%lf
long double	-1.7e4932 to 1.7e4932	8	%Lf
Note: The sizes in this figure are for 32 bit compiler			

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

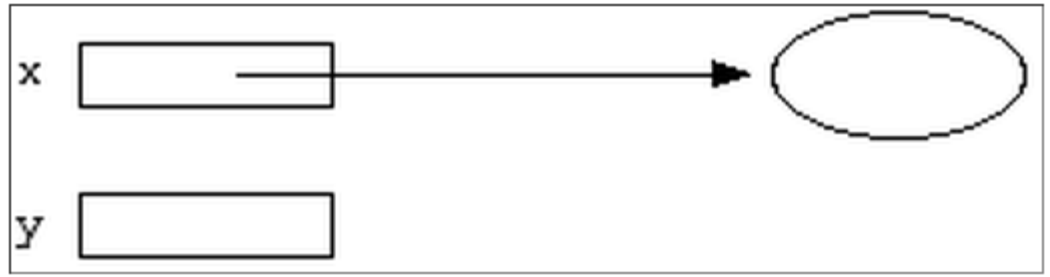
- Example:
  - double x, *ptr;
  - ptr = **&**x; // let ptr point to x
  - *****ptr = 7.8; // assign the value 7.8 to x

Pointer Example

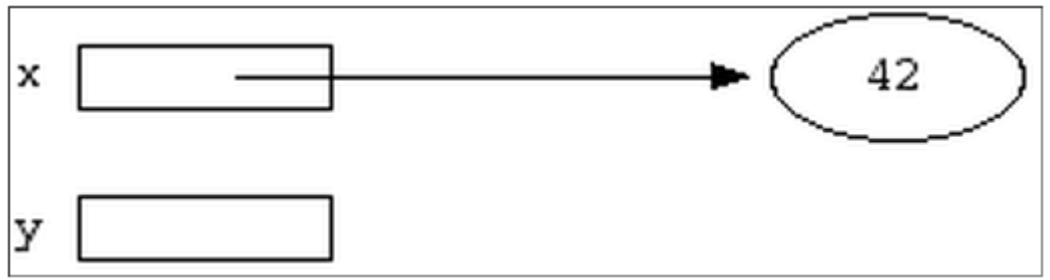
int *x;

int *y;

int var; x = &var;

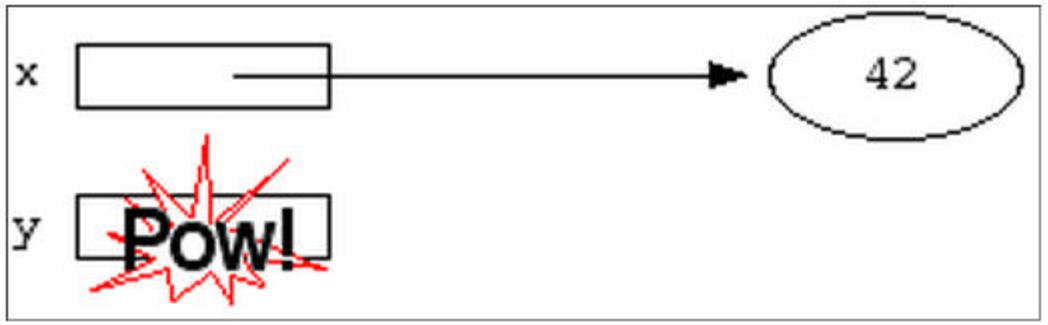


*x = 42;

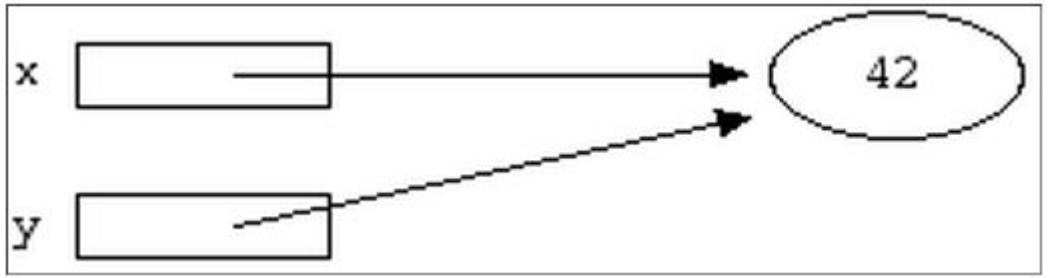


Pointer Example

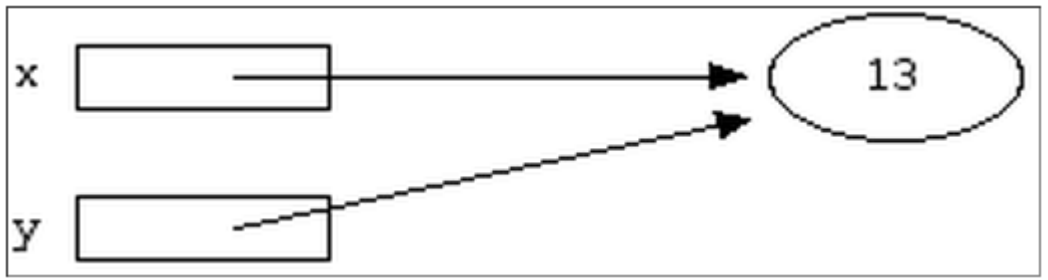
*y = 13;



y = x;

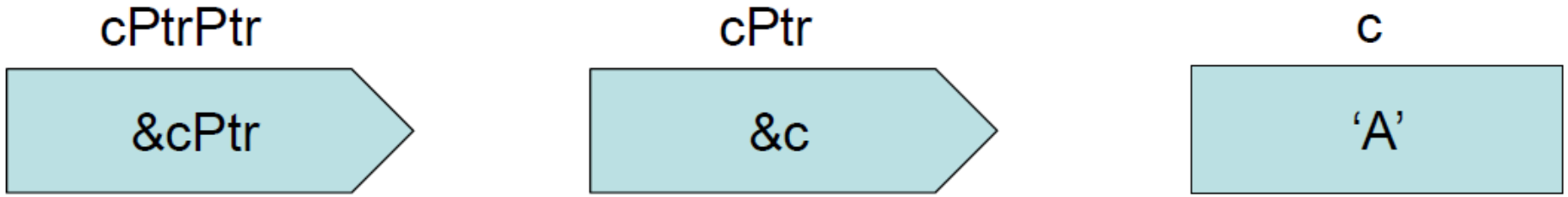


*x = 13; or
*y = 13;



Pointers to Pointers

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



Loops

```
int i; //initialize outside the loop unlike C++
for(i=0;i<10;i++){
}

int i=0; //initialize outside the loop unlike C++
while(i<10){
i++; }
```

Functions

- Function Name
- Return Type
- Arguments/Parameters
- Function Body

```
int func(int a){
        //function body
    return 0;
}
```

Parameter Passing

- **Pass by value**
  - The data associated with the actual parameter is copied into a separate storage location assigned to the formal parameter.
  - Any modifications to the formal parameter variable inside the called function or method affect only this separate storage location and will therefore *not* be reflected in the actual parameter in the calling environment

```
int add(int a, int b) {
        return a+b;
}
void main() {
        int x=4,y=8;
        int z = add(x,y);
        printf("%d",x);
}
```

Parameter Passing...

- **Pass by reference**

  The formal parameter receives a pointer to the actual data in the calling environment. Any changes to the formal parameter *are* reflected in the actual parameter in the calling environment.

```
void swap(int *a, int *b) {
  int tmp = *a;
  *a = *b;
  *b = tmp;
}
void main() {
    int a = 1;
    int b = 2;
    printf("before swap a = %d\n", a);
    printf("before swap b = %d\n", b);
    swap(&a, &b);
    printf("after swap a = %d\n", a);
    printf("after swap b = %d\n", b); }
```

Task 1

- Create a function s.t. it takes three numbers ‘a’, ‘b’ and ‘c’ as arguments, computes a^b and store the results in ‘c’. It should not return any value. Call this function from main() and print the answer in main().

  Hint: pass by reference

  Hint: you may want to see the pow function [check the return type and library] (or compute the exponent yourself <- better)

Task 1 solution

```
#include <stdio.h>
#include <math.h> //library import
void exponent(int a, int b, double *c){
        *c=pow(a,b); //pow returns a pointer
}
int main(void) {
        int a=2;
        int b=2;
        double z;
        exponent(a,b,&z);
        printf("%f", z);
        return 0;
}
```