

Schedule

Monday

Robot Parade: getting started

Office hours: 1:30-3:00 (first-come, first-served)

Thursday Extras completed for semester: To be useful (and to get extra credit),
talk responses needed by Today at 5:00 pm

Tuesday

Office hours: 2:30-4:00 (first-come, first-served)

Wednesday

Last in-class quiz!

Status Reports likely

Robot Parade: working toward completion

Office hours: 1:30-3:00 (first-come, first-served)

Thursday

Office hours (no sign-ups): 2:30-4:00 (in lab)
and 5:00-7:00 (in lab) (first-come, first-served)

Friday

Robot Parade in class

Office hours: 2:30-4:00 (first-come, first-served)

All coursework (extra credit labs & sup. problems, projects)

due by 5:00 pm CST official MathLAN time (both paper, email)

Submission $\geq$ 5:01 pm not accepted/graded

Don't even think about being late

Several clicker questions highlight recent course observations

Suppose the character 'r' requires 1 unit of storage.
How much storage is required for the string "r"?

- A. 1 unit
- B. 2 units
- C. >2 units
- D. amount may vary, according to the local machine

Why is there no MyroC function that allows one to copy one Picture variable to another (similar to strcpy for strings)?

- A. Writing such a function is a wonderful student exercise
- B. In brainstorming functions for MyroC, there did not seem to be a need for copying Pictures
- C. Walker likes to be difficult [sometimes?] [much of the time?]
- D. The students helping Walker develop MyroC wanted to challenge CSC 161 students, and this seemed a great way to do that!
- E. No such function is needed, since C allows easy copying of structs

In MyroC, rTakePicture returns a picture taken by the robot:

```
Picture pic = rTakePicture (); // struct, not pointer, returned
```

However, rDisplayPicture requires a pointer to a Picture, rather than the Picture itself:

```
void rDisplayPicture(Picture * pic, double duration,  
                    const char * windowTitle);
```

Why is the pic parameter a pointer, rather than the Picture itself?

- A. Using a pointer in some functions and the Picture itself in others provides good practice.
- B. Passing the pointer is much faster than passing the entire Picture.
- C. Copying a pointer parameter requires much less memory than copying the entire struct
- D. 2 of A, B, and C
- E. all of A, B, and C

In the images lab (long ago), a procedure was given the center (x_0, y_0) of a circle, and a radius r , and we wanted to know if a point (x, y) was located within the circle.

Four approaches are proposed:

1. $xDist = x - x_0;$
 $yDist = y - y_0;$
 if $(xDist * xDist + yDist * yDist \leq r * r) \dots$
2. $xDist = x - x_0;$
 $yDist = y - y_0;$
 if $(\sqrt{xDist * xDist + yDist * yDist} \leq r) \dots$
3. $xDist = x - x_0;$
 $yDist = y - y_0;$
 if $(\text{pow}(xDist, 2) + \text{pow}(yDist, 2) \leq r * r) \dots$
4. $xDist = x - x_0;$
 $yDist = y - y_0;$
 if $(\sqrt{(\text{pow}(xDist, 2) + \text{pow}(yDist, 2))} \leq r) \dots$

Which approach is most efficient?

- | | | |
|--------------|------|------|
| A. all about | B. 1 | D. 3 |
| the same | C. 2 | E. 4 |

Consider the same 4 approaches to determine if point (x, y) is within a circle of radius 4 from the point (x0, y0)

1. xDist = x - x0;
 yDist = y - y0;
 if (xDist*xDist + yDist*yDist <= r*r) . . .
2. xDist = x - x0;
 yDist = y - y0;
 if (sqrt(xDist*xDist + yDist*yDist) <= r) . . .
3. xDist = x - x0;
 yDist = y - y0;
 if (pow(xDist, 2) + pow(yDist) <= r*r) . . .
4. xDist = x - x0;
 yDist = y - y0;
 if (sqrt((pow(xDist, 2) + pow(yDist))) <= r) . . .

Vote 1: Roughly how much better is 1 than 2

- | | | | | |
|--------|---------|---------|---------|---------|
| A. 2.6 | B. 11.4 | C. 27.9 | D. 30.2 | E. 40.6 |
| times | times | times | times | times |

Vote 2: Roughly how much better is 1 than 2

Vote 3: Roughly how much better is 1 than 3

Vote 4: Roughly how much better is 1 than 4

Robot Parade Project

Overview

- robots line up
- when robots in front clear out
 - one lab partner at the keyboard types character(s)
 - robot moves forward slowly
- when robot reaches starting line
 - one lab partner at the keyboard types character(s)
 - robot follows commands read from a file

Overall, robots will

- start in classroom
- turn, beep, move forward in stages
- progress into hallway
- turn right
- move down hallway in front of classroom window

Robot Parade Project

3 main parts

- program to respond to input commands
 - based on robot command project
- adjust input for either user or file
 - based on command-line argument
- allow logging
 - option to record each robot action
 - motivation:
 - if user types sequence of commands
 - record each command (with parameters) in file
 - option details based on command-line argument

Robot Parade Project

likely development process

- start with robot command project
- add variables for file input/logging
 - defaults: stdin/NULL
 - use command-line arguments to set file variables
 - see Wednesday's class notes for command-line processing
- change input prompts/scanf to
 - if inputFile == stdin, then prompt
 - scanf becomes fscanf
 - clear line after processing each line
- add logging statement when loggingFile != NULL
 - print line with command and any needed parameters
- just one read/write section, with fscanf and possible logging for each command (not multiple read/write sections)