lab III.2: surveyor robot class, programmer defined functions

Name:

information

• You will have one class period to work on this lab: Thurs MARCH 20. There may also be some extra time if needed.

• The assignment is due at the end of class (HARD COPY) on MON MARCH 31

• The assignment is worth 10 points

programming assignment (TO BE DONE WITH A PARTNER)

• MY PARTNER’S NAME IS: [ ]

• After you get each program to work, write the code in the boxes provided. Partial credit will be given!

• Demonstrate each working program for your instructor.

1 Released under the Copyleft provision.

Produced at Brooklyn College by John Cummins with assistance from M. Q. Azhar, and supervision from Professor Sklar.
vocabulary

- Surveyor
- XBee USB Dongle
- text editor
- compile
- link
- command prompt
- make command
- cd (change directory) command
- class
- object
- member function

materials

Make sure you have all of the following materials before you start the lab:

- Surveyor robot
  - The Surveyor is a cute little tracked robot that communicates with a PC via a XBee USB dongle.

- c++ compiler, such as mingw (Windows) or g++ (Mac)
- codeblock, text editor, such as Notepad (Windows) or TextEdit (Mac) or pico (Unix)
- you must be in
  - \dev\examples directory in Windows
  - /dev/examples directory In Mac or Unix or Linux computer.
- Please note that Windows computers use back slash (\) to separate directory (e.g., \dev\examples) while all other operating systems (e.g., Linux, Unix and Mac) use forward slash (/) to separate directory (e.g., /dev/examples)
- use cd command in all machine to change directory. ask if you need help)
instructions

1. create your first program
   • Enter the following program using a text editor:

```c
#include "SVR.h"

#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <assert.h>
using namespace std;

/* make a global Surveyor object called robot */
Surveyor robot (ADDRESS);

int main()
{
    robot.drive(50, 50, 100);
    return 0;
}
```

In your editor, make sure you save the file as `begin.cpp` in `\dev\examples` directory!

2. dissecting the code
   • `#include "SVR.h"`;
     This line always needs to be included at the top of all programs written for the Surveyor. Whenever you use a command in a program like `robot.drive(50,50,100)`, the Surveyor has to know what that means. The `#include` line tells the program where to find a “library” of all the surveyor specific terms. Please note that this header is not part of standard C++ library. `SVR.h` is specially written for Surveyor robot.

   • `#include <stdio.h>`;
     This line is required for standard input output

   • `#include <stdlib.h>`;
     This header defines several general purpose functions, including dynamic memory management, random number generation, communication with the environment, integer arithmetics, searching, sorting and converting.

   • `#include <iostream>`;
     This header is an object-oriented library that provides input and output functionality using streams.
• #include <assert.h>
   This header defines one macro that can be used as a standard debugging tool.

• using namespace std;
   This line lets you use cout and cin without additional syntax (e.g., std::cout).

• Surveyor robot (ADDRESS);

   – Creating class object: This line creates an object of class type Surveyor called robot. You may use this object (e.g., robot) to send commands (e.g., (drive(50, 50, 100)) to your robot.
   – Global declaration: This is a global declaration and can be used by the main() program and as well as your defined functions in this program.

• int main ( )
   This line needs to be in every program. This denotes the main function. This is what will be run when the program starts.

• {
   This is a beginning curly bracket. Everything between the beginning ( ) and ending ( ) curly brackets is part of main. Curly brackets are used to delineate not only the main but also the function and blocks within a program.

• robot.drive(50, 50, 100);
   – drive() is a member function of Surveyor class.
   – Function Parameters: drive(int left, int right, int duration) member function of Surveyor class takes three integer parameters. For example, in our drive(50, 50, 100) member function call:
     * first parameter asks the robot to set its left track speed at 50 (the first 50)
     * second parameter asks the robot to its right track speed at 50 and
     * third parameter asks the robot to let it run for 100 hundredths of a second or 1 second.
   – Creating class object: Usually, you need to have class object to call member function. For example,
     Surveyor robot (ADDRESS);
     created an object called robot.
   – Calling member function: Use the dot operator ( . ) to call your member function object-Name.MemberFunction() (e.g., robot.drive(50, 50, 100));
   – A member function (e.g., drive()) is an entity in an object-oriented program that contains instructions for the object (e.g., robot) to do something, like perform an action (e.g., move) or set the value of a variable (data field).

• more about drive member function

   – The drive() member function of the Surveyor class is very useful for driving your surveyor robot.
   – The left and right track speeds can be in the range -128 to +127.
   – Use negative left or right track speeds (e.g., -50) to go in opposite direction.

   – The duration is in the range 0 to 255.
   – A duration of 0 means until the next drive command. The duration parameter is best for precise control, any value longer than a few hundredths of a second is probably best done by using a duration of 0 and having your PC do the timing. Examine the sleep functions available on your system.
3. compile your program

- From the Windows command window or the Mac terminal window, at the command-line prompt (`prompt>`), type:
  ```
  prompt> make begin
  ```
  This will compile and link the program.
- If there are errors in your program code, fix them and then try compiling and linking again—until the program compiles and links without any errors.

4. run your program

- Make sure the **XBee dongle** is be same channel as your robot and is connected to your computer’s USB port.
- Connect it to your computer’s USB port (ask if you need help finding it).
- The Surveyor must be turned on.
- On Windows, at the command-line prompt (`prompt>`), type:
  ```
  prompt> begin
  ```
  or on the Mac (or Linux or Unix) terminal window, at the command-line prompt (`prompt>`), type:
  ```
  prompt> ./begin
  ```
  This will run your program and send commands to the robot.
- What did your robot do? Is that what you expected? Write your answer below.

---

**troubleshooting**
- If you have trouble, check these things:
  1. The Surveyor must be turned on during communication.
  2. LEDs on the robot and dongle should be on.
  3. the `make` command should be in your path (ask if you need help).
  4. you should be in the right directory:
     * On Windows:
       - `\dev\examples` directory in Windows.
     * On Mac:
       - `/dev/examples` directory In Mac or Unix or Linux computer.
  5. Make sure the **XBee dongle** is be same channel as your robot and is connected to your computer’s USB port.

5. modify the program

- Now change the program to make your robot go backwards for 4 seconds and then stop.
- **Edit** the program using your text editor, compile it, run it again to test it.
- Repeat this process until it works!

6. programming challenges

- Complete as many of the following programs as you can.
- After you get each program to work, demonstrate your working program for your instructor.
• Make sure that you save the program as begin.cpp and follow the procedures as described earlier to test your program.

(a) Program the robot to go forward for 2 seconds and then go backward for 2 seconds and then stop. Is it back where it started from? (Hints: use negative left or right track speed to move the robot in opposite direction)

(b) Make your robot to select a random direction and travel a random distance in that direction (hints: use rand and srand functions).

(c) Program your robot to go in a square.
   • Write a function called square() that is designed to make the robot go in a square
   • Write a main() function that does the following:
     – Call the square() function

(d) Program the robot to go in a spiral pattern like this:

(e) Program your robot to go in a square, triangle, circle and spiral.
   • Write a function called square(), triangle(), circle(), and spiral() which are designed to make the robot go in a square, triangle, circle and spiral respectively.
   • All four functions have no parameters and return void
   • Write a main() function that does the following:
     i. add five letters to the set of user commands that will allow the user to tell the surveyor to go in a square (or triangle or circle or spiral). Use the following letters:
        | Letter | Command                        |
        |--------|--------------------------------|
        | Q      | quit the program               |
        | s      | make the robot go in a square  |
        | t      | make the robot go in a triangle|
        | c      | make the robot go in a circle  |
        | p      | make the robot go in a spiral  |

     Note the distinction between upper and lower case letters!
     ii. each user’s choice will call the corresponding function to make the robot move according to user’s choice.