

Programming – Constants and Variables

Aim: How can constants and variables make our programs easier to write?

Goals

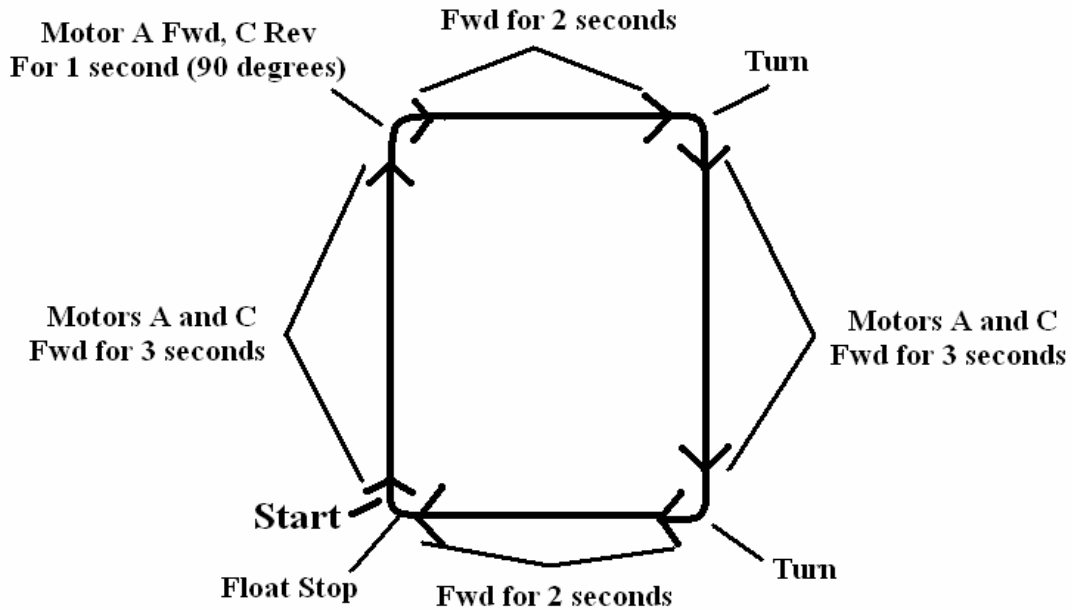
- Discuss how constants and variables can be used to make our programs easier to write
- Learn how the repeat function can make our programs more concise
- Become familiar with how certain mathematical operations are expressed in NQC
- Interpret programs that contain these various aspects

Do Now

- Explain what is going on in the following program.
- Assuming that a basic Tankbot would turn 90 degrees in one second at power level 5, what is the path of motion that it would take if this program was downloaded to it?

```
task main()  
{  
    SetPower(OUT_A + OUT_C, 5);  
    OnFwd(OUT_A + OUT_C);           Wait(300);  
    Rev(OUT_C);                     Wait(100);  
    Fwd(OUT_C);                     Wait(200);  
    Rev(OUT_C);                     Wait(100);  
    Fwd(OUT_C);                     Wait(300);  
    Rev(OUT_C);                     Wait(100);  
    Fwd(OUT_C);                     Wait(200);  
    Rev(OUT_C);                     Wait(100);  
    Float(OUT_A + OUT_C);  
}
```

Path of Motion



Problem With the Program

- What if we weren't sure of the amount of time it takes to make a 90 degree turn?
- How would that make this program annoying/difficult to write?

Introducing a Constant

- Instead of changing each '100' to the correct 'turn time,' we can simply replace them with a constant called 'TURN_TIME'
- The constant gets created before **task main** is written
- Now we can easily change each TURN_TIME in the program
- Constants are defined in the form:

```
#define CONSTANT_NAME value
```

```
#define TURN_TIME 100
```

Defines constant named
TURN_TIME which equals 100

```
task main()
{
    SetPower(OUT_A + OUT_C, 5);
    OnFwd(OUT_A + OUT_C);           Wait(300);
    Rev(OUT_C);                     Wait(TURN_TIME);
    Fwd(OUT_C);                     Wait(200);
    Rev(OUT_C);                     Wait(TURN_TIME);
    Fwd(OUT_C);                     Wait(300);
    Rev(OUT_C);                     Wait(TURN_TIME);
    Fwd(OUT_C);                     Wait(200);
    Rev(OUT_C);                     Wait(TURN_TIME);
    Float(OUT_A + OUT_C);
}
```

Make More Constants

- How can we easily alter the dimensions of the rectangle?
- Create more constants...
 - LONG_SIDE
 - SHORT_SIDE
- We can conveniently change the value of the constants to make the rectangle smaller or larger
- Constants cannot change their value within a program

```
#define TURN_TIME 100
#define SHORT_SIDE 200
#define LONG_SIDE 300

task main()
{
    SetPower(OUT_A + OUT_C, 5);
    OnFwd(OUT_A + OUT_C);           Wait(LONG_SIDE);
    Rev(OUT_C);                     Wait(TURN_TIME);
    Fwd(OUT_C);                     Wait(SHORT_SIDE);
    Rev(OUT_C);                     Wait(TURN_TIME);
    Fwd(OUT_C);                     Wait(LONG_SIDE);
    Rev(OUT_C);                     Wait(TURN_TIME);
    Fwd(OUT_C);                     Wait(SHORT_SIDE);
    Off(OUT_A + OUT_C);
}
```

Looping in NQC

- We can repeat sections of a program by simply using the repeat () command
- The basic setup of the command is:
 - **repeat (# of times task will execute)**
 - {Task to Repeat }**
- This can help us simplify/shorten our program

Rectangle Program

- How can we shorten this program using the repeat command?
- Repeat
- In the rectangle program, we have one task that we repeat twice:
 - Go forward (long side), turn 90 degrees
 - Go forward (short side), turn 90 degrees

```
#define TURN_TIME 100
#define SHORT_SIDE 200
#define LONG_SIDE 300

task main()
{
    SetPower(OUT_A + OUT_C, 5);

    repeat(2)
    {
        OnFwd(OUT_A + OUT_C);      Wait(LONG_SIDE);
        Rev(OUT_C);                Wait(TURN_TIME);
        Fwd(OUT_C);                Wait(SHORT_SIDE);
        Rev(OUT_C);                Wait(TURN_TIME);
    }
    Off(OUT_A + OUT_C);
}
```

Repeat Brackets

- All repeats must have one open and one close bracket
- Indenting helps improve readability of a program
- The task that is being repeated can be easily seen

Nesting Repeats

- How can we modify the previous program so that we make 4 rectangles?

```

#define TURN_TIME 100
#define SHORT_SIDE 200
#define LONG_SIDE 300

task main()
{
    SetPower(OUT_A + OUT_C, 5);
    repeat(4)
    {
        repeat(2)
        {
            OnFwd(OUT_A + OUT_C);    Wait(LONG_SIDE);
            Rev(OUT_C);              Wait(TURN_TIME);
            Fwd(OUT_C);              Wait(SHORT_SIDE);
            Rev(OUT_C);              Wait(TURN_TIME);
        }
    }
    Off(OUT_A +OUT_C);
}

```

Constants in Operations

- Although constants cannot be given new values within a program, we can add, subtract, multiply, and divide them to make new numbers

- For example:

```

#define TURN_TIME 100
Wait(TURN_TIME*2); // Wait 2 sec

```

Multiplying Constants Example

```

#define TURN_TIME 100

task main()
{
    SetPower(OUT_A + OUT_C, 5);
    repeat(2)
    {
        OnFwd(OUT_A + OUT_C);    Wait(TURN_TIME*3);
        Rev(OUT_C);              Wait(TURN_TIME);
        Fwd(OUT_C);              Wait(SHORT_SIDE*2);
        Rev(OUT_C);              Wait(TURN_TIME);
    }
    Off(OUT_A +OUT_C);
}

```

Variables

- Memory locations in which we can store a value for later use: $X = ?$
- The value of a variable can be changed in the middle of a program
- What kinds of values might we store?
- Defining a variable: use keyword **int**
 - int stands for integer
- Variable names must start with a letter
 - May contain an underscore or number

Mathematical Operators

- Variables can be used to change a task during a program
- Ex: Increasing the dimensions of the rectangle each time it repeats
- Operators used to alter the value of a variable
 - **LONG_SIDE += 5;** – increase LONG_SIDE by 5
 - **LONG_SIDE -= 5;** – decrease LONG_SIDE by 5
 - **LONG_SIDE /= 5;** – divide by 5
 - **LONG_SIDE *= 5;** – multiply by 5

Variables

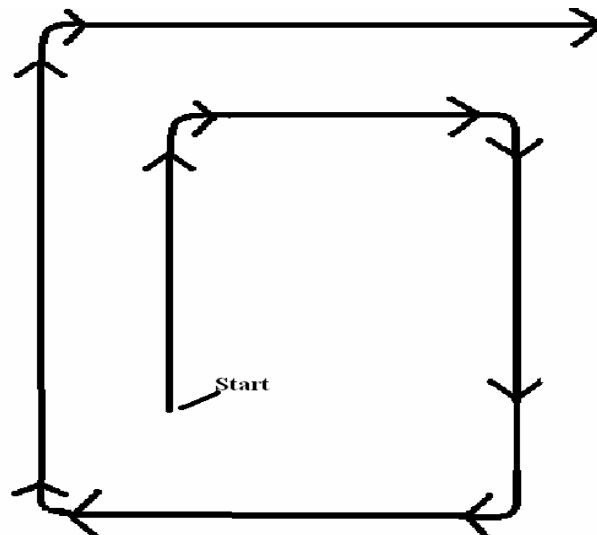
- Must be given an initial value
- The initial value can be assigned in the definition command
- Then, using mathematical operators, the variable can be adjusted as the program runs

What Shape Will The Program Make?

```
#define TURN_TIME      100

int LONG_SIDE; int SHORT_SIDE;      // Variables can be defined on 1 line

task main( )
{
    SetPower(OUT_A + OUT_C, 5);
    LONG_SIDE = 300;
    SHORT_SIDE = 200;
    repeat (4)
    {
        OnFwd(OUT_A + OUT_C);
        Wait(LONG_SIDE);
        Rev(OUT_C);
        Wait(TURN_TIME);
        Fwd(OUT_C);
        Wait(SHORT_SIDE);
        Rev(OUT_C);
        Wait(TURN_TIME);
        LONG_SIDE += 50;
        SHORT_SIDE += 50;
    }
    Off(OUT_A + OUT_C);
}
```



```

int move_time;

task main( )
{
    move_time = 60;
    repeat(4)
    {
        OnFwd(OUT_A + OUT_C);
        Wait(move_time);
        Rev(OUT_A + OUT_C);
        Wait(move_time);
        move_time *= 2;
    }
    Off(OUT_A + OUT_C);
}

int aaa;
int bbb; int ccc;

task main()
{
    aaa = 10;
    bbb = 20*5;
    ccc = bbb;
    ccc /= aaa;
    ccc -= 5;
    aaa = 10 * (ccc + 3);
}

// What is the final value of aaa?

```

Random Initial Value

- Instead of giving a variable a known initial value, you can generate a random initial value
- ie: `LONG_SIDE = Random(300);`
 - `LONG_SIDE` will initially be a number between 0 and 300

```

#define TURN_TIME    100

int LONG_SIDE = Random(300);
int SHORT_SIDE = Random(200);

task main()
{
    SetPower(OUT_A + OUT_C, 5);
    repeat(4)
    {
        OnFwd(OUT_A + OUT_C);           Wait(LONG_SIDE);
        Rev(OUT_C);                       Wait(TURN_TIME);
        Fwd(OUT_C);                       Wait(SHORT_SIDE);
        Rev(OUT_C);                       Wait(TURN_TIME);
        LONG_SIDE += 50;
        SHORT_SIDE += 50;
    }
    Off(OUT_A + OUT_C);
}

```

Power Level Example

- Starting at minimum power, increase the power level of your motors by 1 every second until at full power

```

int power = 0;

task main()
{
    repeat(8)
    {
        SetPower(OUT_A + OUT_C, power);

        OnFwd(OUT_A + OUT_C);           Wait(100);

        power += 1;
    }

    Off(OUT_A + OUT_C);
}

```