

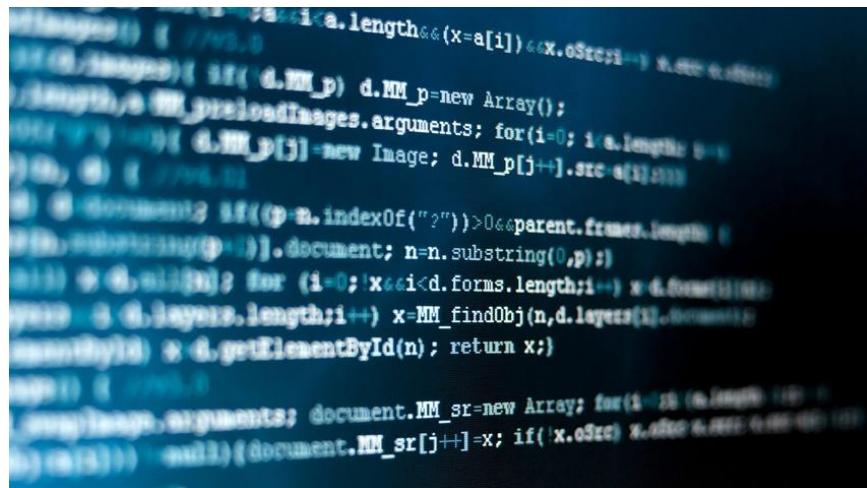
Vex Robotics Summer Program Day-2



Introduction to Programming

▶ What is programming?

- Programming is writing vocabulary and a set of grammatical rules for instructing a computer or machine to perform specific tasks.

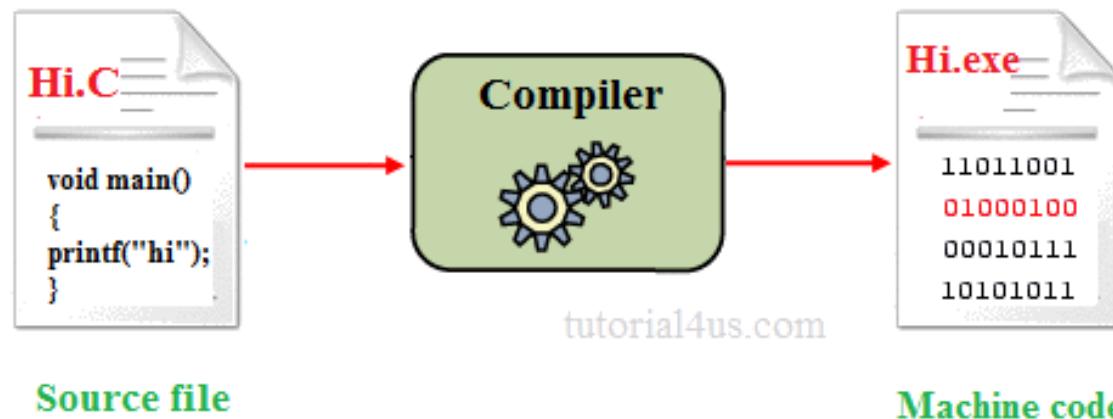


The image shows a dark-themed code editor window with a block of programming code. The code appears to be written in JavaScript, containing various functions and variables related to image loading and processing. The syntax highlighting includes blue for keywords like 'function' and 'for', red for strings, and green for comments. The code is scrollable, with a vertical scrollbar visible on the right side of the editor window.

Introduction to Programming

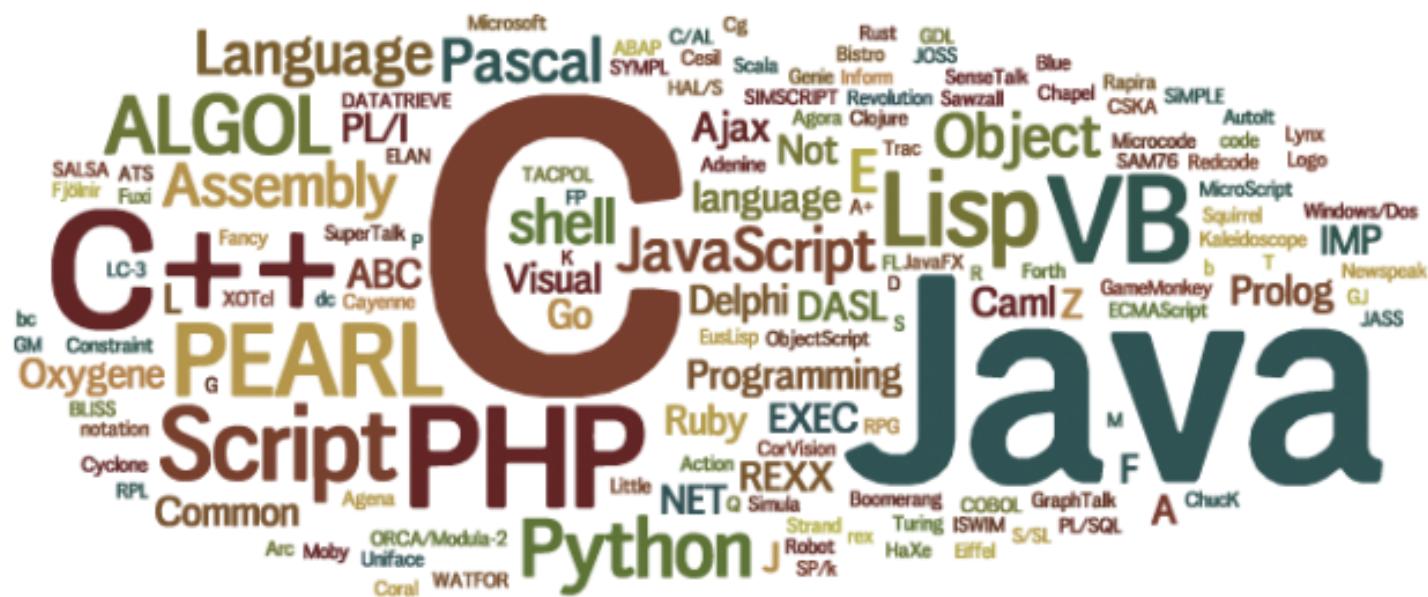
► Where do we program?

- Programmers program on a compiler. A compiler is a program that converts instructions into a machine-code or lower-level form so that they can be read and executed by a computer. It acts a translator for the computer.



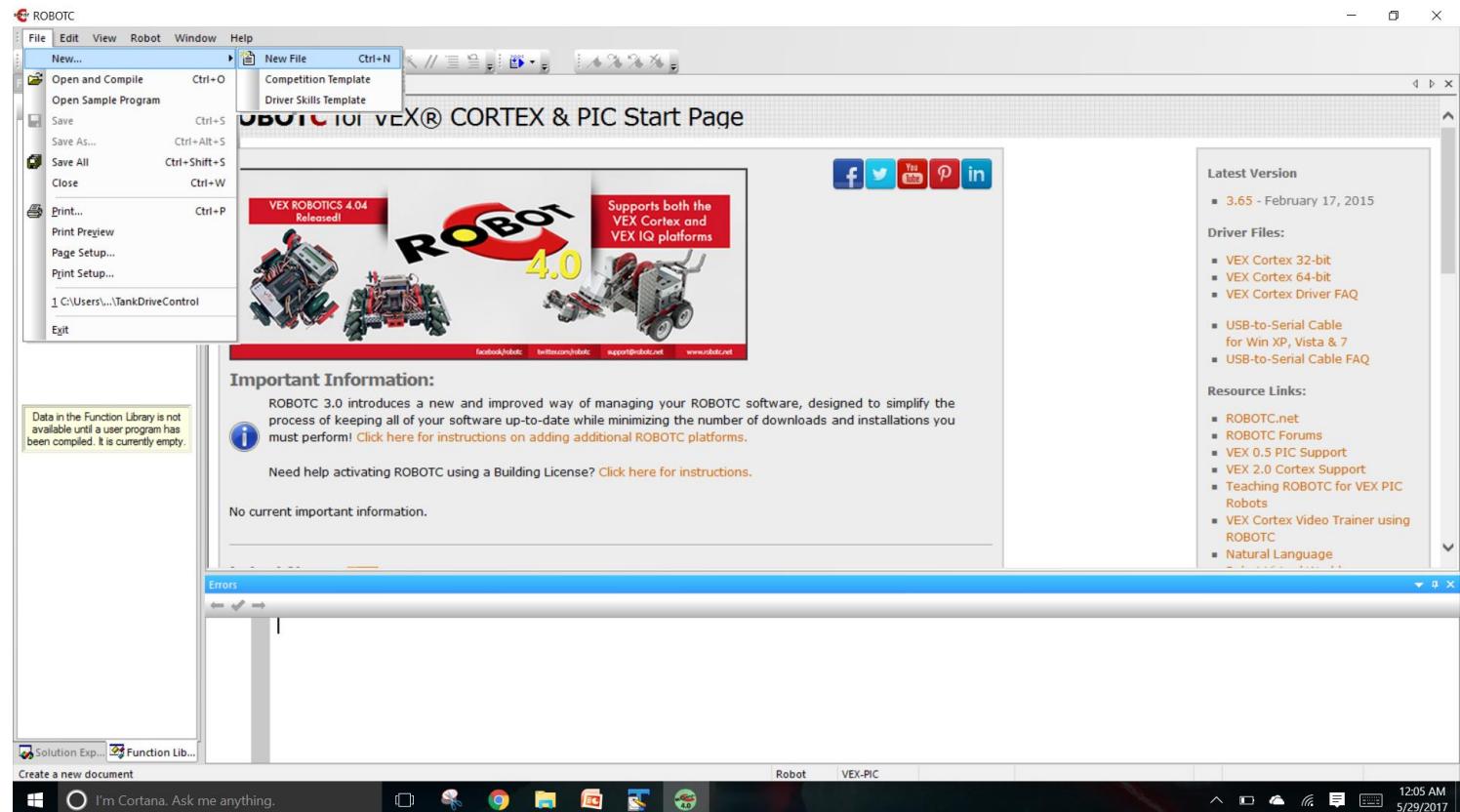
Introduction to Programming

- ▶ There are many programming languages but for Vex Robotics we will be focusing on C



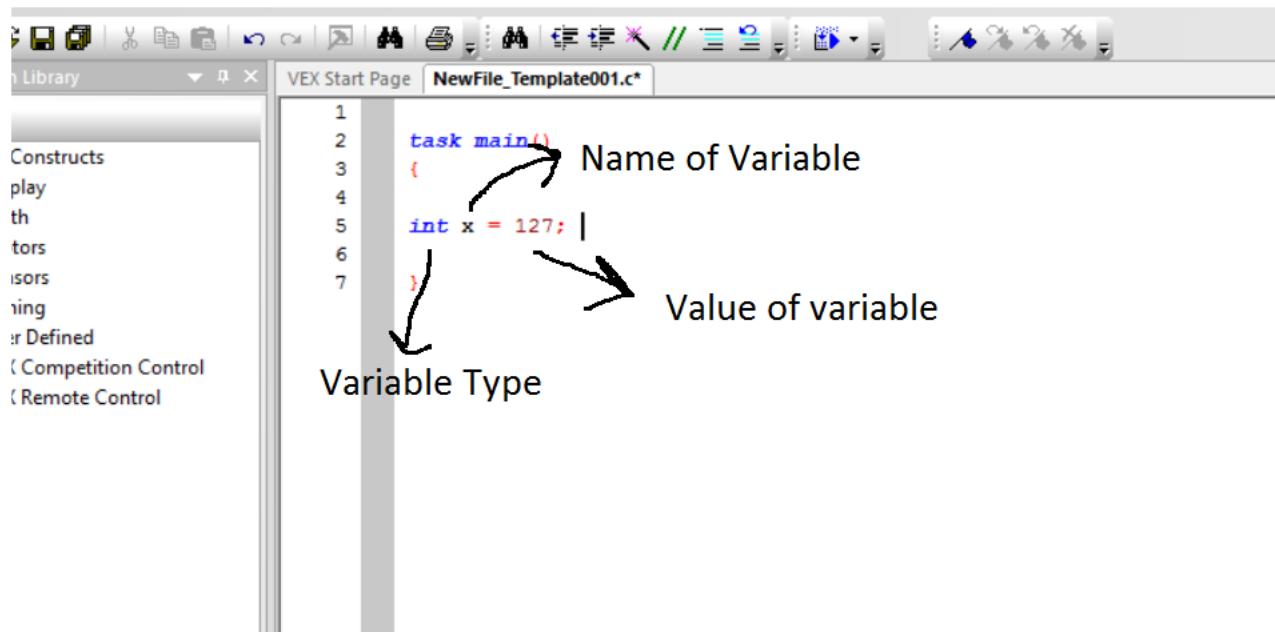
Basic Programming

- ▶ Open up Robot C and Create a new file for our first project



Basic Programming - Integers

- ▶ An int or an integer is a type of variable that stores a whole number between -32768 to 32768.



The screenshot shows a software interface for programming. On the left is a library browser with categories like Constructs, play, th, tors, isors, ring, user Defined, Competition Control, and Remote Control. The main window displays a C program:

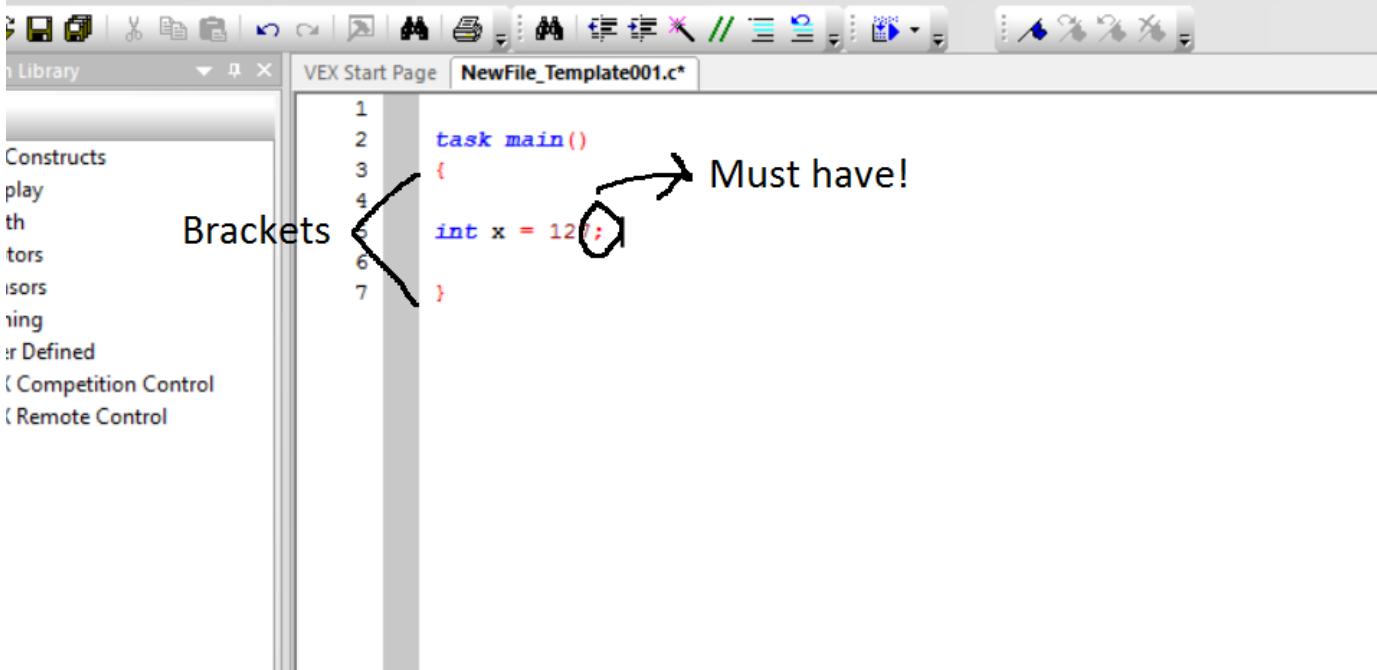
```
1 task main()
2 {
3     int x = 127;
4 }
```

Annotations with arrows point to specific parts of the code:

- A red arrow points to the word "x" in the line `int x = 127;`, labeled "Name of Variable".
- A red arrow points to the word "int" in the same line, labeled "Variable Type".
- A black arrow points to the number "127" in the same line, labeled "Value of variable".

Basic Programming - Quick Tips

- ▶ Make sure code is in the bracket, compiler only reads what is in the main task
- ▶ Never forget semicolon at the end of any statement!!



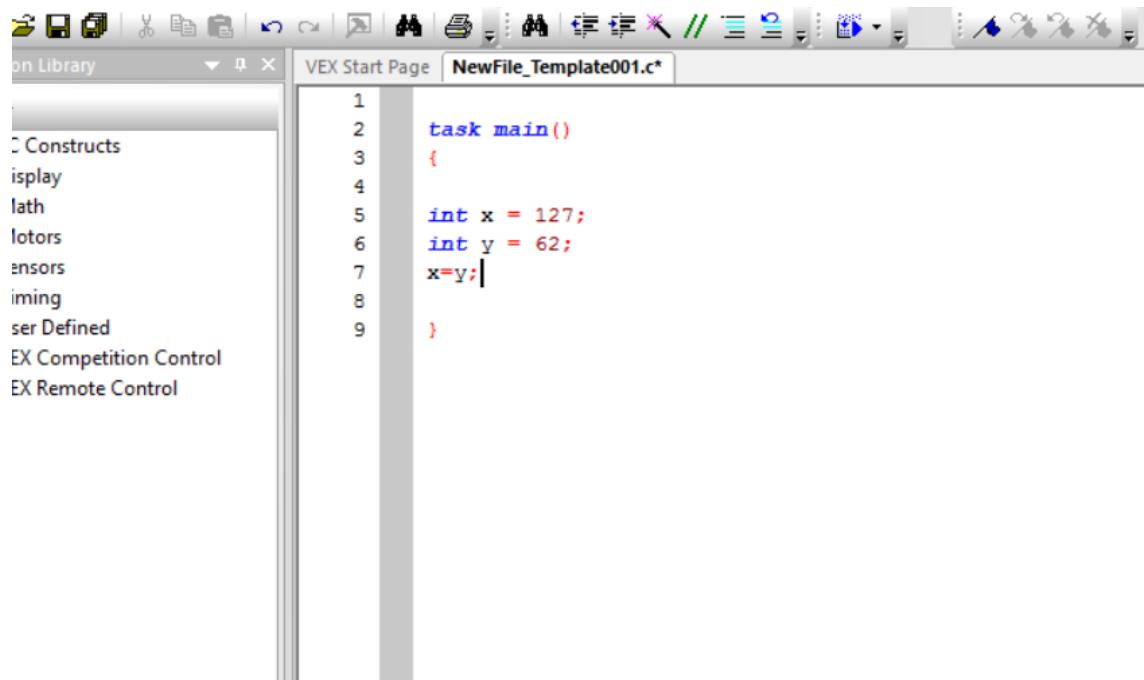
The screenshot shows a software interface for programming VEX robots. The menu bar includes 'File', 'Edit', 'View', 'Project', 'Tools', 'Help'. The left sidebar has sections like 'Constructs', 'play', 'th', 'tors', 'isors', 'ning', 'er Defined', '(Competition Control', and '(Remote Control'. The main window title is 'VEX Start Page' and the file name is 'NewFile_Template001.c*'. The code editor contains the following C code:

```
1
2     task main()
3     {
4
5         int x = 12; // Must have!
6
7     }
```

Two annotations are present: a bracket on the left side of the code area with the label 'Brackets' pointing to it, and an arrow pointing to the semicolon at the end of line 5 with the text 'Must have!'.

Basic Programming - Integers

- ▶ Whenever compiler reads an integer, the first integer is set equal to the value of the second
- ▶ After running the program below, both x and y will be equal to 62

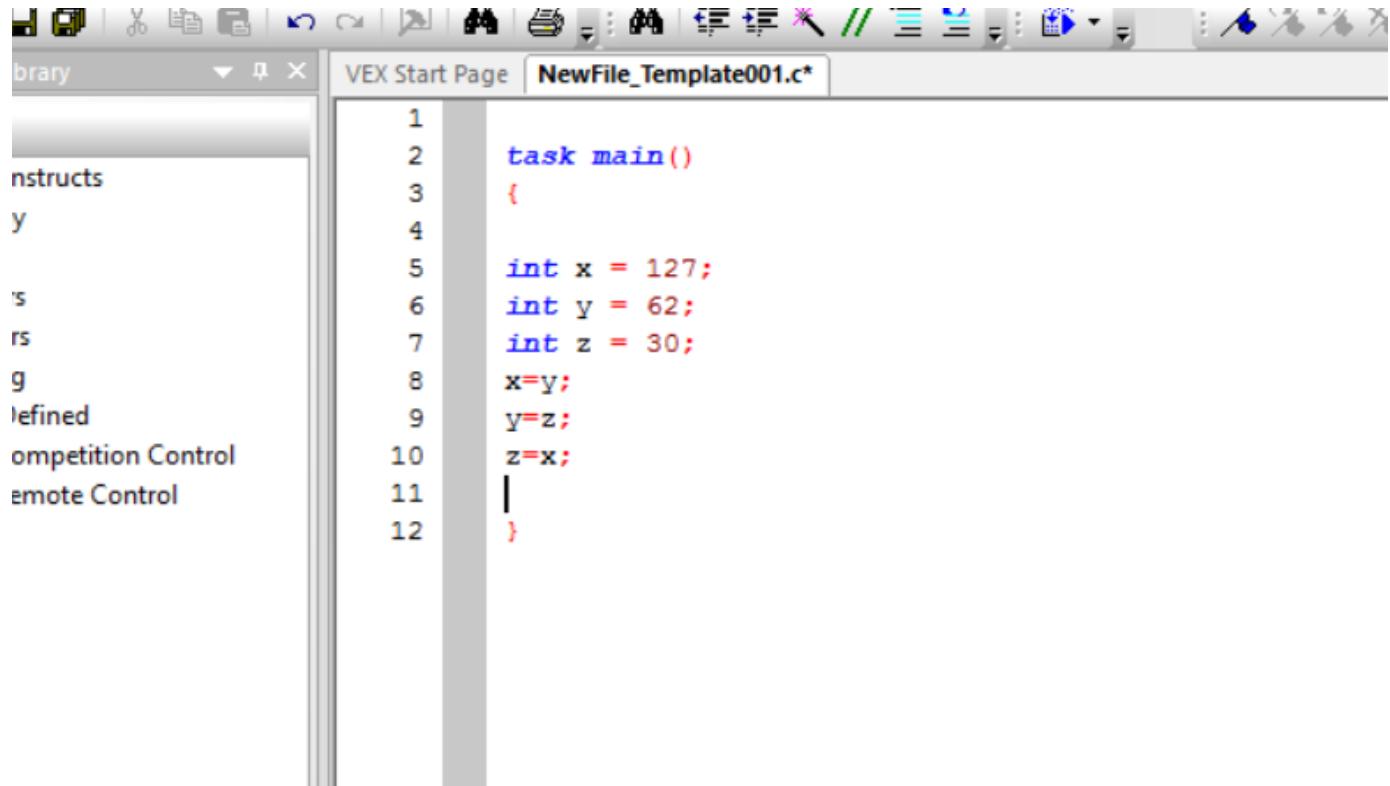


The screenshot shows a software interface for programming VEX robots. The title bar says "VEX Start Page" and "NewFile_Template001.c*". The left sidebar has a "Function Library" tab and a list of constructs: Constructs, isplay, lath, lotors, ensors, iming, ser Defined, EX Competition Control, and EX Remote Control. The main window displays the following C code:

```
1
2  task main()
3  {
4
5      int x = 127;
6      int y = 62;
7      x=y;
8
9 }
```

Basic Programming - Practice

- After running this program, what is the value of x, y, and z?



The screenshot shows the VEX Studio IDE interface. On the left is a library browser with categories like Instructs, Sensors, Motors, and Competition Control. The main window displays a code editor titled "NewFile_Template001.c*" containing the following C-like pseudocode:

```
1
2 task main()
3 {
4
5     int x = 127;
6     int y = 62;
7     int z = 30;
8
9     x=y;
10    y=z;
11    z=x;
12 }
```

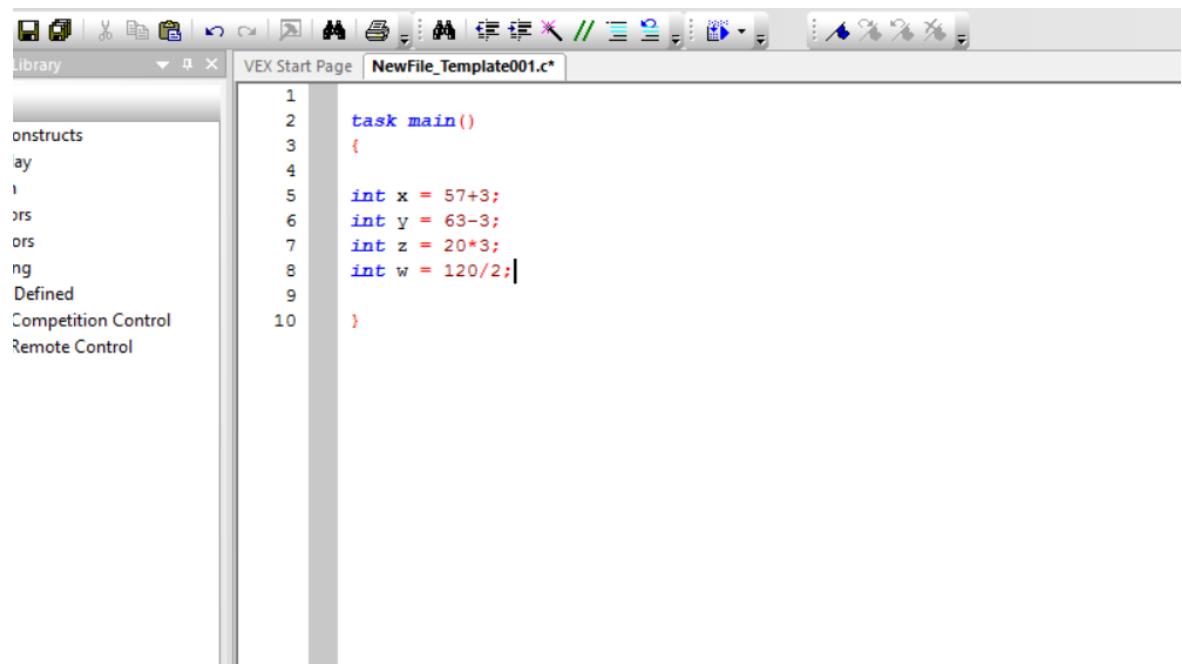
Basic Programming – Practice

▶ Answer

- $x = 62$
- $y = 30$
- $z = 62$

Basic Programming – Operations

- ▶ With integers, you can do basic mathematical operations such as addition, subtraction, multiplication, and division using “+”, “-”, “*”, and “/” operators
- ▶ In the scenario below, x, y, z, and w are all equal to 60

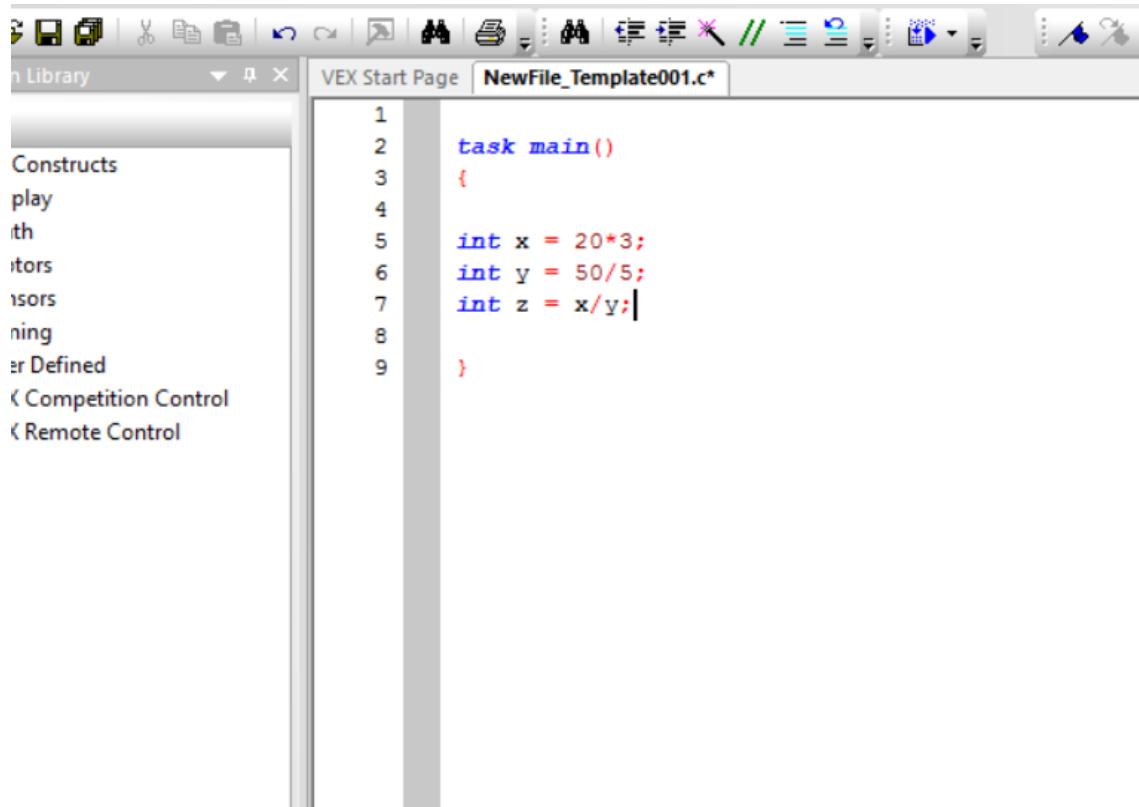


```
task main()
{
    int x = 57+3;
    int y = 63-3;
    int z = 20*3;
    int w = 120/2;
}
```

The screenshot shows the VEX Start Page software interface. On the left is a library pane containing sections like Constructs, Sensors, Motors, and others. The main window displays a C code editor titled "NewFile_Template001.c*". The code shown is a simple task main() block that performs four integer calculations: addition (x = 57+3), subtraction (y = 63-3), multiplication (z = 20*3), and division (w = 120/2). All variables are declared as integers.

Basic Programming - Practice

- ▶ What does z equal to?



The screenshot shows the VEX Start Page in STEAM Works Studio. The interface includes a toolbar at the top, a library browser on the left, and a code editor on the right. The code editor window title is "VEX Start Page NewFile_Template001.c*". The code itself is:

```
1  task main()
2  {
3
4
5      int x = 20*3;
6      int y = 50/5;
7      int z = x/y;
8
9 }
```

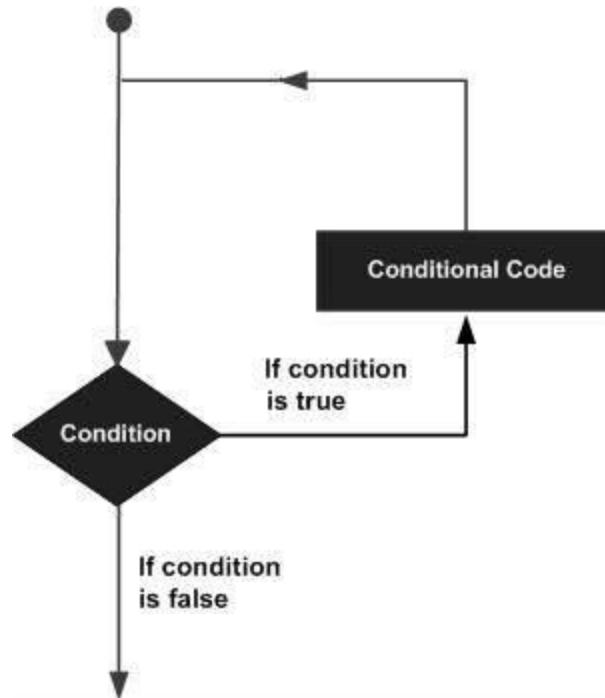
Basic Programming – practice

► Answer:

- $z = 6$

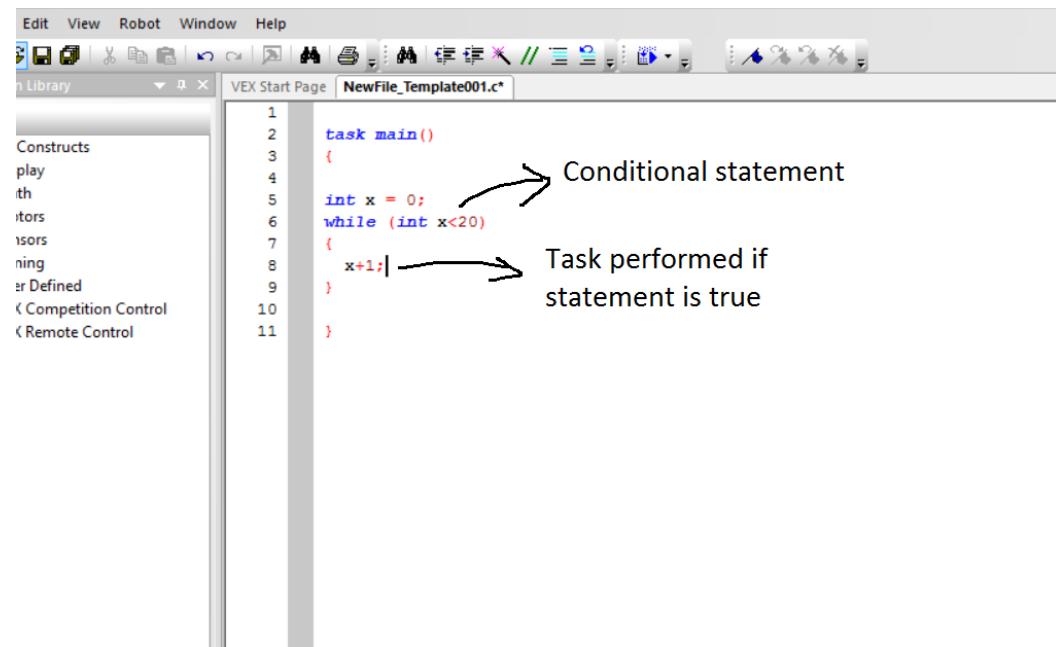
Basic Programming - Loops

- ▶ What is a loop?
 - A loop is a sequence of instructions that is continually repeated until a certain condition is reached



Basic Programming - While Loops

- In a while loop, the statement in parenthesis is the condition and what's in the brackets is the operation
- In the scenario below, x starts at zero, and the loop will continuously add 1 to x until it reaches 20. Thus the statement loops 20 times



The screenshot shows a software interface for programming a VEX robot. The menu bar includes Edit, View, Robot, Window, and Help. A toolbar with various icons is visible above the code editor. The code editor window is titled "VEX Start Page" and contains the file "NewFile_Template001.c". The code is as follows:

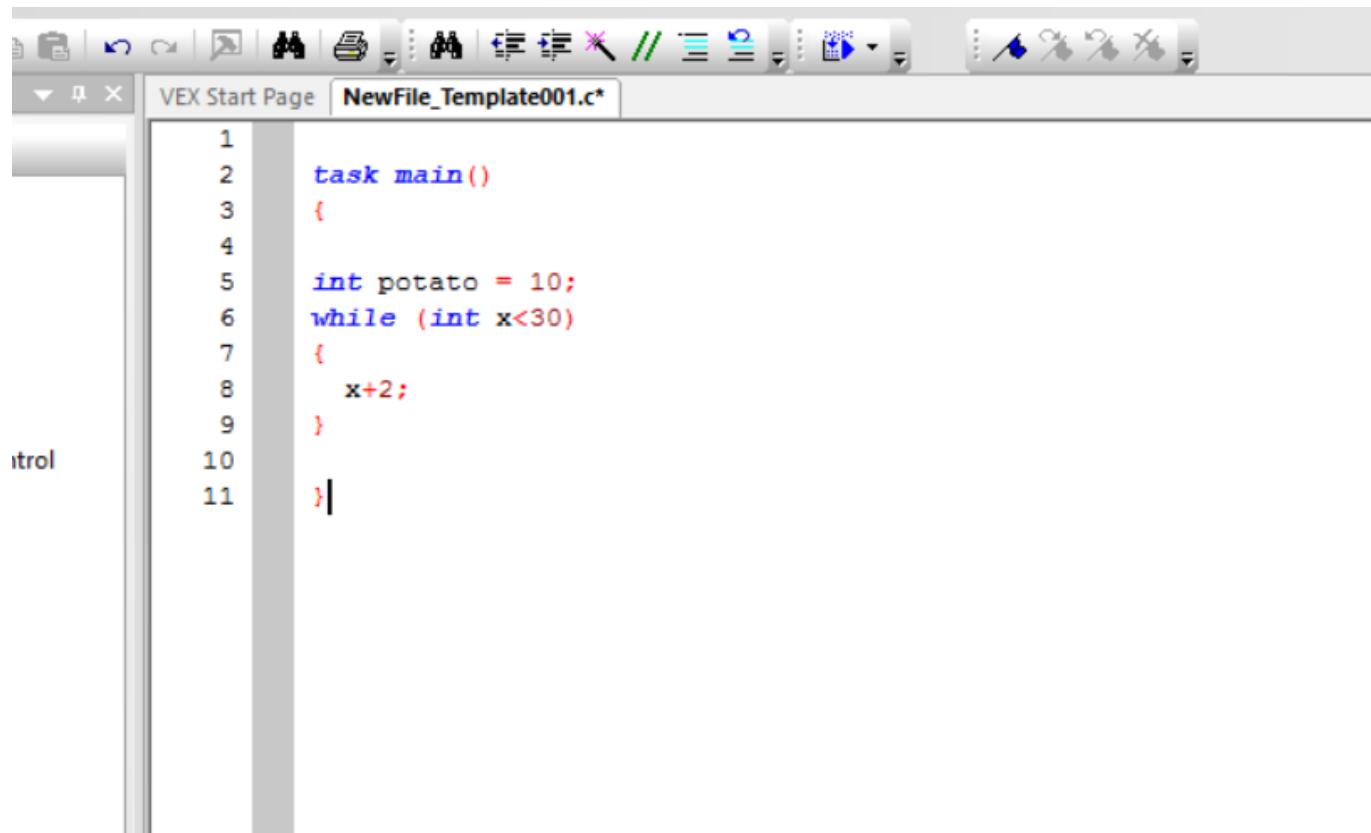
```
1  task main()
2  {
3
4      int x = 0;
5      while (int x<20)
6      {
7          x+1;
8      }
9
10 }
11
```

Annotations with arrows explain the code:

- An arrow points to the "while" keyword with the label "Conditional statement".
- An arrow points to the "x+1;" line with the label "Task performed if statement is true".

Basic Programming - Practice

- ▶ How many times does the loop run below?



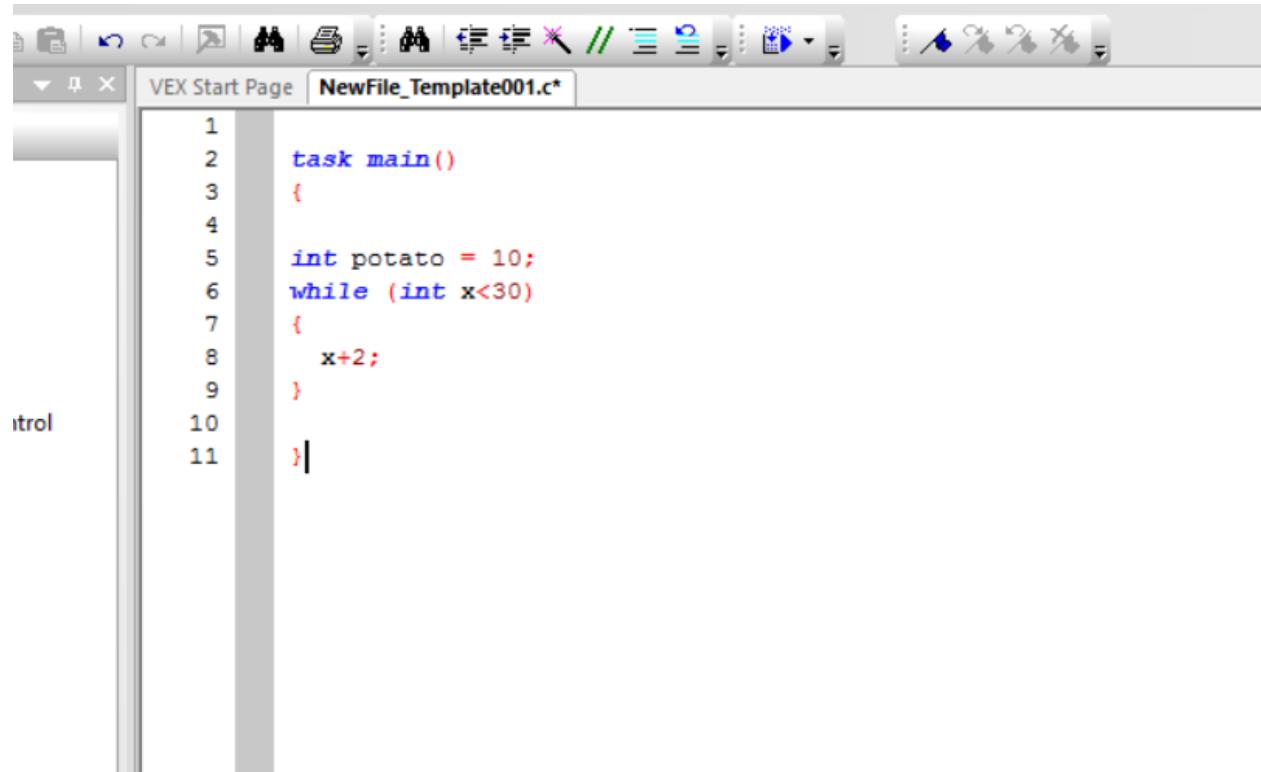
The screenshot shows a C code editor window titled "VEX Start Page" with the file name "NewFile_Template001.c*". The code is as follows:

```
1
2     task main()
3     {
4
5         int potato = 10;
6         while (int x<30)
7         {
8             x+2;
9         }
10    }
```

The code defines a task named "main". It initializes an integer variable "potato" to 10. A while loop runs as long as the value of "x" is less than 30. Inside the loop, the expression "x+2;" is evaluated. The cursor is located at the end of the loop's closing brace on line 10.

Basic Programming - Practice

- ▶ Answer:
 - 10 times



The screenshot shows the VEX Studio IDE interface with a code editor window titled "NewFile_Template001.c*". The code is as follows:

```
task main()
{
    int potato = 10;
    while (int x<30)
    {
        x+2;
    }
}
```

The code defines a task named "main". It initializes an integer variable "potato" to 10. A while loop runs as long as the value of "x" is less than 30. Inside the loop, the expression "x+2;" is present, which appears to be a typo for "x+=2;".

Basic Programming - if/else statements

- ▶ The if/else statement executes a block of code if a specified condition is true. If the condition is false, the else block of code can be executed
- ▶ In conditional statements, we use double equal signs instead of single
- ▶ In the code below, what is the value of x after running the code?

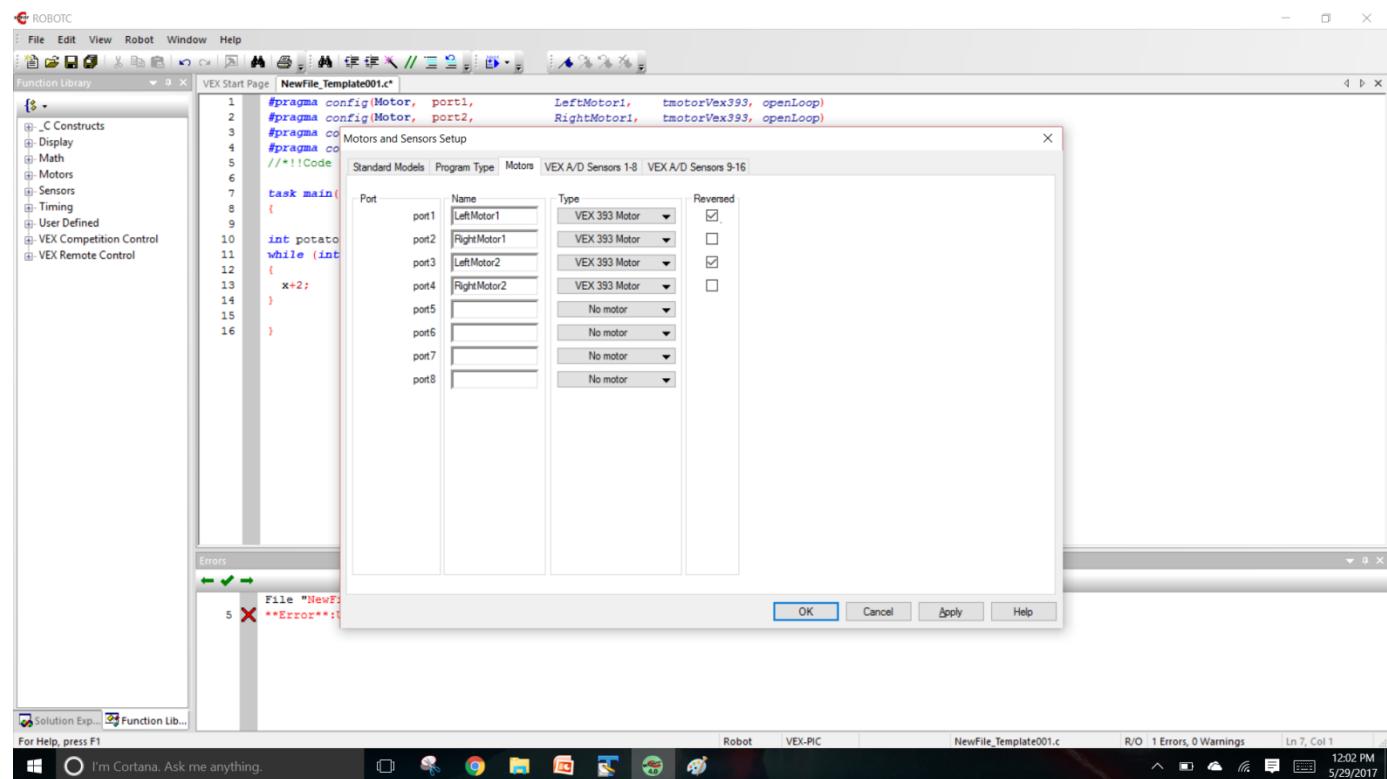
```
3 #pragma config(Motor, port3, LeftMotor2, tmotorVex39
4 #pragma config(Motor, port4, RightMotor2, tmotorVex39
5 //!!!Code automatically generated by 'ROBOTC' configuration wizard
6
7 task main()
8 {
9     int x = 20;
10    if(int x == 20)
11    {
12        x =30;
13    }
14    else
15    {
16        x=35;|
17    }
18}
19}
```

Basic Programming - if/else statements

- ▶ Answer:
 - 30

Robot Programming - Motors

- ▶ First thing we will program are the motors
- ▶ To set it up, click “robot” → “motors and sensors setup”, and name four motors as named below. Set each of the motors to type “393 motor” and reverse two of them.



Robot Programming – Motors

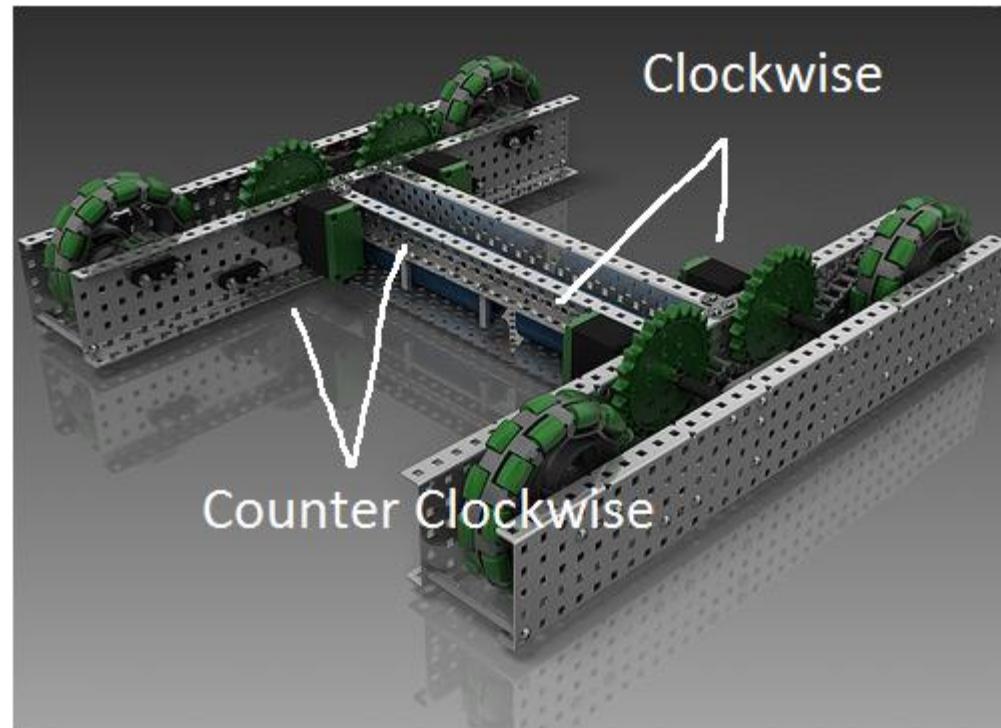
- ▶ If we are programming the base, then why are two of the motors in reverse? How does the structure impact the direction of rotation?



Robot Programming – Motors

▶ Answer

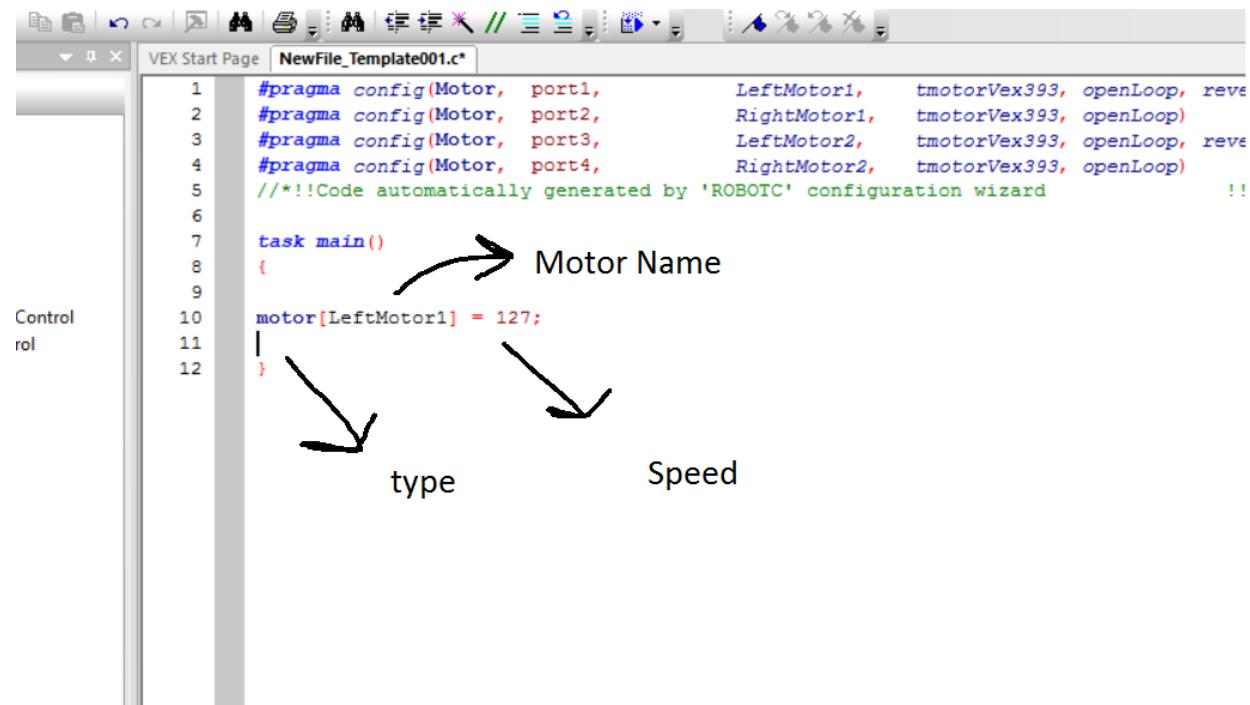
- Because of the way the motors set up mechanically, they have to spin in opposite directions to move forward



Robot Programming – Motors

- To program the motors, in the main task, we write “`motor[motorName] = speed;`”. The value we put for the speed is any number between -127 and 127. 127 being the fastest clockwise and -127 being the fastest counter clockwise.

Control
rol



```

VEX Start Page NewFile_Template001.c
1 #pragma config(Motor, port1, LeftMotor1, tmotorVex393, openLoop, rev
2 #pragma config(Motor, port2, RightMotor1, tmotorVex393, openLoop)
3 #pragma config(Motor, port3, LeftMotor2, tmotorVex393, openLoop, rev
4 #pragma config(Motor, port4, RightMotor2, tmotorVex393, openLoop)
5 /*!!Code automatically generated by 'ROBOTC' configuration wizard !!
6
7 task main()
8 {
9
10    motor[LeftMotor1] = 127;
11 }

```

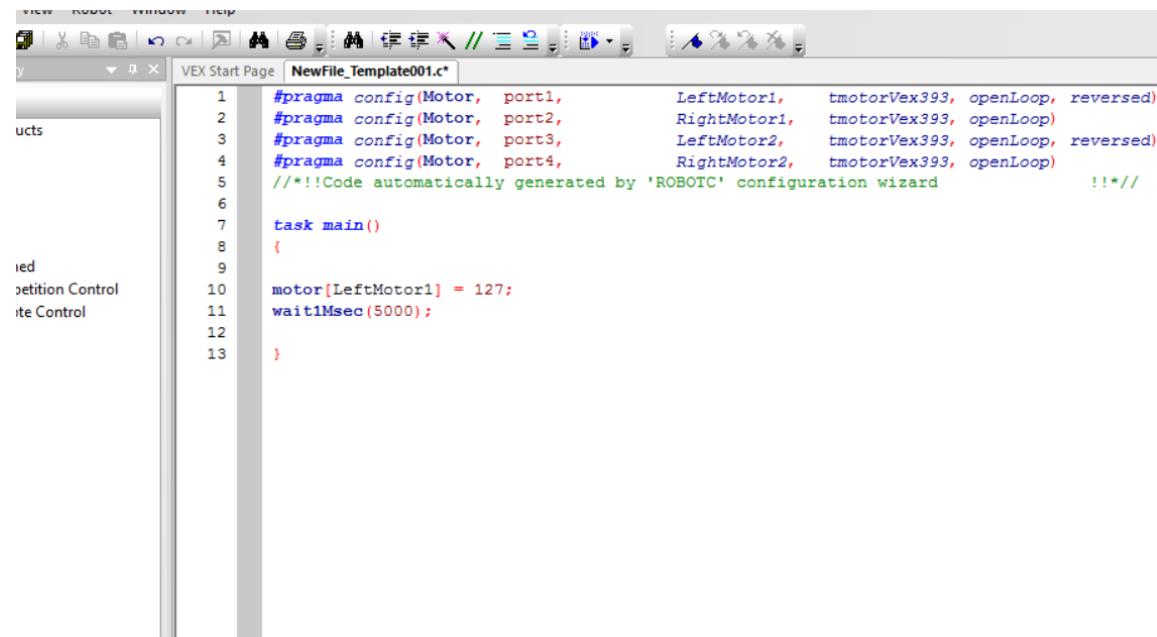
Motor Name

type

Speed

Robot Programming – wait1Msec

- ▶ wait1Msec
 - Method that tells motors how long to spin
 - Measure in milliseconds
 - The code on the bottom will move the motor for 5 seconds



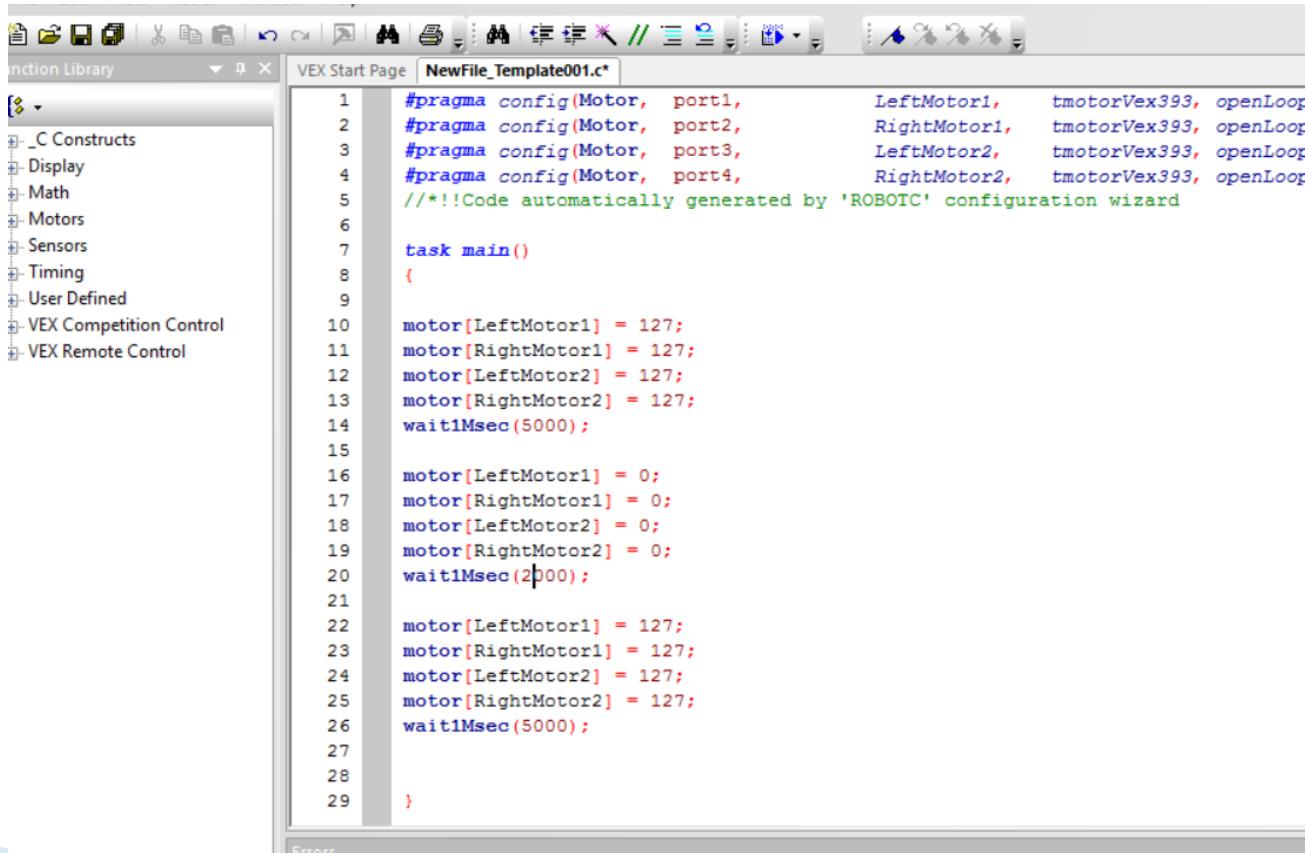
```
VEX Start Page NewFile_Template001.c*
1 #pragma config(Motor, port1, LeftMotor1, tmotorVex393, openLoop, reversed)
2 #pragma config(Motor, port2, RightMotor1, tmotorVex393, openLoop)
3 #pragma config(Motor, port3, LeftMotor2, tmotorVex393, openLoop, reversed)
4 #pragma config(Motor, port4, RightMotor2, tmotorVex393, openLoop)
5 //!!!Code automatically generated by 'ROBOTC' configuration wizard !!!
6
7 task main()
8 {
9
10    motor[LeftMotor1] = 127;
11    wait1Msec(5000);
12
13 }
```

Programming – wait1Msec

- ▶ Program base to move for 5 seconds, stop for 2 seconds, and move for 5 seconds. Then upload the code and test on robot

Robot Programming - wait1Msec

- ▶ Code should look like this



The screenshot shows the VEX Works Studio IDE interface. On the left is a Function Library panel with categories like C Constructs, Display, Math, Motors, Sensors, Timing, User Defined, VEX Competition Control, and VEX Remote Control. The main window displays a code editor titled "NewFile_Template001.c". The code is a template for a VEX competition program:

```
#pragma config(Motor, port1, LeftMotor1, tmotorVex393, openLoop)
#pragma config(Motor, port2, RightMotor1, tmotorVex393, openLoop)
#pragma config(Motor, port3, LeftMotor2, tmotorVex393, openLoop)
#pragma config(Motor, port4, RightMotor2, tmotorVex393, openLoop)
///*!Code automatically generated by 'ROBOTC' configuration wizard

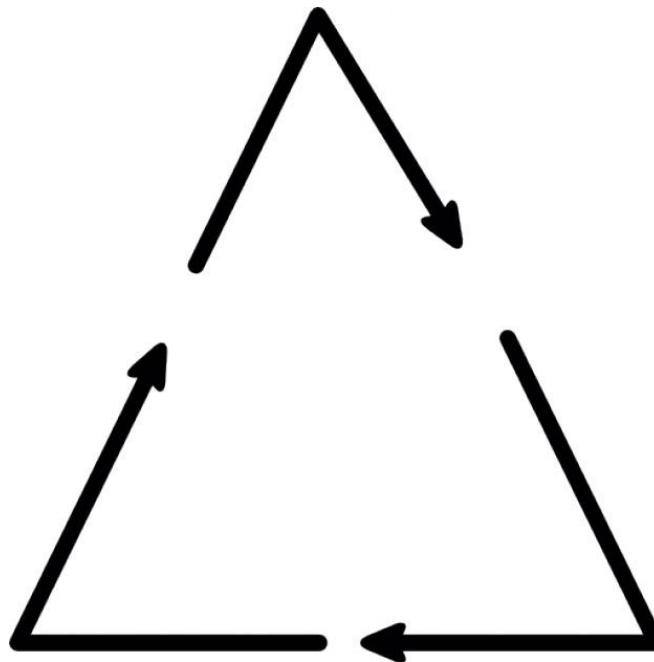
task main()
{
    motor[LeftMotor1] = 127;
    motor[RightMotor1] = 127;
    motor[LeftMotor2] = 127;
    motor[RightMotor2] = 127;
    wait1Msec(5000);

    motor[LeftMotor1] = 0;
    motor[RightMotor1] = 0;
    motor[LeftMotor2] = 0;
    motor[RightMotor2] = 0;
    wait1Msec(2000);

    motor[LeftMotor1] = 127;
    motor[RightMotor1] = 127;
    motor[LeftMotor2] = 127;
    motor[RightMotor2] = 127;
    wait1Msec(5000);
}
```

Robot Programming - Challenge

- ▶ Program the robot to move in a triangle



Robot Programming – Driver Control

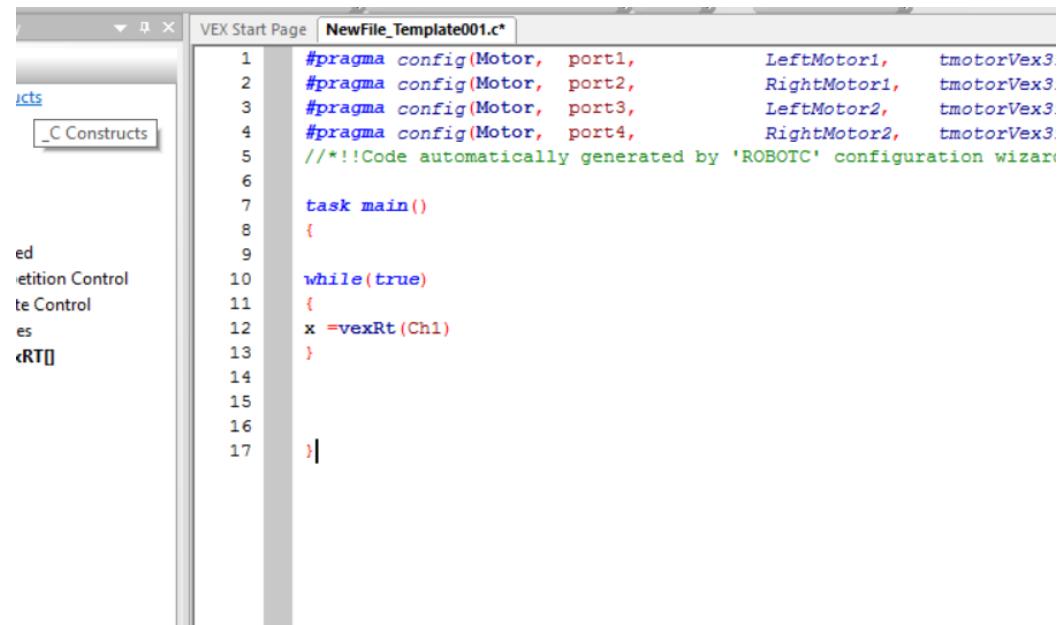
- ▶ Now we get into how to program with remote control
- ▶ Everything must first be in a continuous while loop, we set while(true) to always run it



Robot Programming – Driver Control

▶ Method VexRT()

- This method returns a value between -127 and 127 depending on how far the thumb sticks are pressed
- In the bottom code, if thumb stick is pressed all the way up, x will equal 127, if it is pressed all the way down, x will equal -127
- In the parenthesis, we write the channel number, which can range from ch1, ch2, ch3, ch4 depending on what the axis



The screenshot shows the STEAM Works Studio IDE interface. On the left, there's a sidebar with options like 'File', 'Edit', 'Search', 'Recent Projects', 'Project Control', 'Build Control', 'Help', and 'VEX RT[]'. A dropdown menu labeled 'C Constructs' is open. The main window displays a code editor titled 'NewFile_Template001.c*' under the 'VEX Start Page' tab. The code is as follows:

```

1 #pragma config(Motor, port1, LeftMotor1, tmotorVex3);
2 #pragma config(Motor, port2, RightMotor1, tmotorVex3);
3 #pragma config(Motor, port3, LeftMotor2, tmotorVex3);
4 #pragma config(Motor, port4, RightMotor2, tmotorVex3);
5 //!!!Code automatically generated by 'ROBOTC' configuration wizard!!!
6
7 task main()
8 {
9
10 while(true)
11 {
12     x =vexRt(Ch1)
13 }
14
15
16
17 }

```

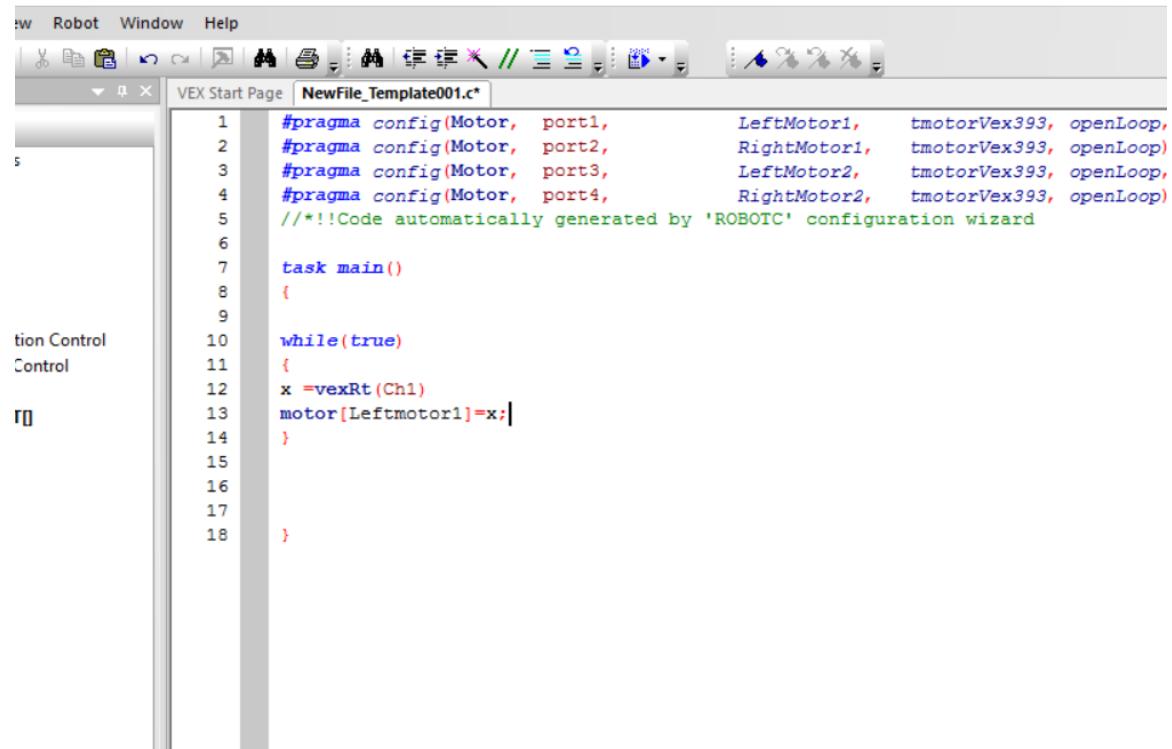
Robot Programming – Practice

How would you use the method VexRT and variables to move a motor when you the move thumb sticks on a remote?

Robot Programming – Practice

▶ Answer:

- Setting the variable equal to VexRT and setting the speed of the motor equal to that variable



The screenshot shows the VEX Works Studio IDE interface. The menu bar includes 'File', 'Robot', 'Window', and 'Help'. The toolbar contains various icons for file operations and project management. The main window displays a code editor with a tab labeled 'VEX Start Page' and 'NewFile_Template001.c'. The code is a template for a VEX robot, starting with pragmas for motor configurations and a main task loop that reads from VexRT channel 1 and sets the speed of LeftMotor1.

```
#pragma config(Motor, port1, LeftMotor1, tmotorVex393, openLoop,
#pragma config(Motor, port2, RightMotor1, tmotorVex393, openLoop)
#pragma config(Motor, port3, LeftMotor2, tmotorVex393, openLoop,
#pragma config(Motor, port4, RightMotor2, tmotorVex393, openLoop)
//!!!Code automatically generated by 'ROBOTC' configuration wizard

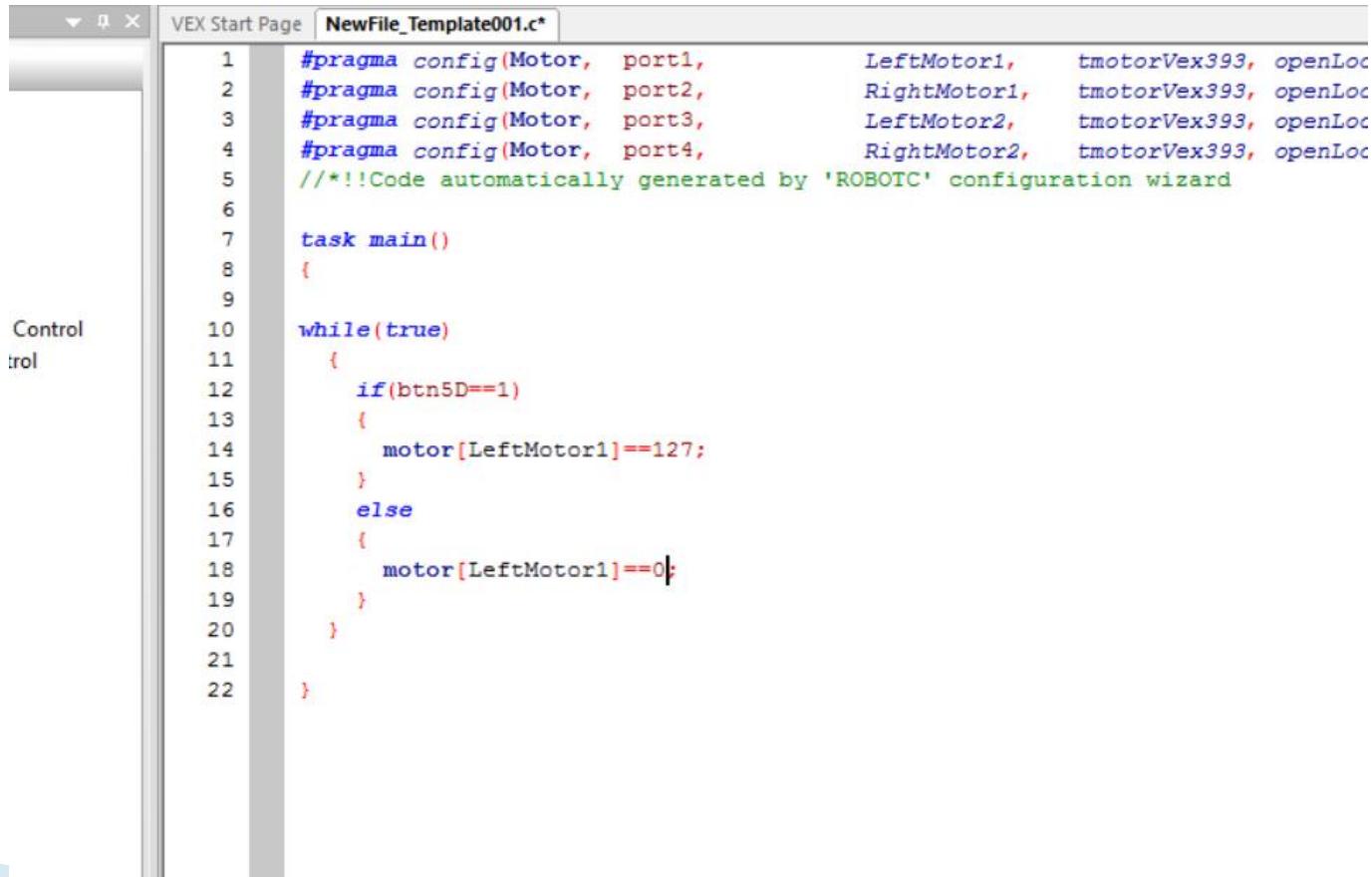
task main()
{
    while(true)
    {
        x = vexRt(Ch1)
        motor[Leftmotor1]=x;
    }
}
```

Robot Programming – Button

- ▶ Buttons are like thumb sticks except they take return value of 1 or 0
- ▶ We use Btn5U to indicate the uppermost button in group 5, and Btn7R for right most button in group 7 and so forth.
- ▶ Using if statements, program a motor to move forward if the lower button in group 5 is pressed

Robot Programming – Button

▶ Answer:



The screenshot shows a computer screen displaying a VEX Start Page window titled "NewFile_Template001.c". The code is written in C and controls a motor based on a button input. The code includes pragmas for motor configuration and a main task loop that checks a button and sets a motor speed accordingly.

```
1 #pragma config(Motor, port1, LeftMotor1, tmotorVex393, openLoc
2 #pragma config(Motor, port2, RightMotor1, tmotorVex393, openLoc
3 #pragma config(Motor, port3, LeftMotor2, tmotorVex393, openLoc
4 #pragma config(Motor, port4, RightMotor2, tmotorVex393, openLoc
5 /*!!Code automatically generated by 'ROBOTC' configuration wizard
6
7 task main()
8 {
9
10 while(true)
11 {
12     if(btn5D==1)
13     {
14         motor[LeftMotor1]==127;
15     }
16     else
17     {
18         motor[LeftMotor1]==0;
19     }
20 }
21
22 }
```

Robot Programming – Challenge

- ▶ Program the entire base to move forward, backwards, left, and right with remote control. Incorporate buttons and thumb sticks.

