**Programming Guide for VEXCode C++**

**Downloading:**

**Variables:**

Definition – something that stores data to be reused.

Declaration – the act of making a variable (does not include values). Make it like this datatype name;

Null – every variable starts with null data (this does not mean 0 this means literally no value whatsoever)

Initialization – the act of giving a variable non-null data for the 1st time. Name = correct\_data;

(in some cases use name = new DataType();)

Assigning – changing the value of a variable (every initialization is an assignment but not vice versa).

Name = correct\_data;

Incrementing/Decrementing – used for ints (mostly for loops). Variable++; or Variable--;

Data Types:

* Integers: -1, 0, 2, 5… basically any number without decimals (typical range [-2,147,483,647 to 2,147,483,647])
* Booleans: True or False. Used to check if some condition has been met such as if button is pushed or not
* Doubles: -1.1, 0.0, 3.67832… same as Integer but with decimals (and a much larger range, max: ~1.79769\*10^308)
* String: “cat”, “dog”, “mouse”… stores words and is shown with quotes around it. Main use for GUIs.
* Char: ‘c’, ‘?’, ‘$’… stores one character and represented with single quotes. Not very useful.

Constants – after initialization data can never be changed.

const datatype name = correct\_data;

//declaration (has no value a.k.a. null value)

int example1;

//initialization (has to be in some method or at the start)

void necessaryMethod( void ) {

example1 = 1;

//assigning new data

example1 = 2;

//incrementing (value will be 3)

example1++;

//decrementing (value wil be 2)

example1--;

}

//declaration and initialization in same step

int example2 = 4;

//data types (spacing for clarity)

int exampInt = -4;

bool exmpBool = true;

double exampDouble = 4.3478951478274;

std::string exampeString = "Thingy";

char exampChar = 'a';

Static (advanced and niche) – used with multiple Objects. Variable can be changed but there is only ever one of these. Static variable;

Private (adv. and niche) – only inside the class can the variable be accessed. Private variable:

Public (adv. and niche) – everything inside and outside class can access the variable. Public variable;

Vex specific dataTypes:

* Motor – can be made automatically with robot config button (no picture sorry, but near the top right).

motor name = motor(PORT#, ratio#\_#, Boolean);

* Vex::brain (done automatically)– used to access stuff on the cortex specifically (mostly the Screen). vex::brain name;
* Vex::competition (done automatically) – used so the people who run competition can access your code correctly.

vex::competition Competition;

* Controller (done automatically)– used to access buttons and joystick values. controller name = controller();

Your Unique Variables (very niche) – used if you make your own class and need to create an Object for that class. YourClassDataType name = new YourClassDataType(parameters);

**Loops:**

While loops – continually runs until the condition is false

//need to do loops inside of methods

void necessaryMethod(){

//variable not necessary in some cases but very useful

int i = 0;

/\*while( some condition ){

code;

}\*/

//while print "loop still running" 41 times

while( i<=40 ){

i++;

Brain.Screen.print("loop still running");

}

//example without ints (will only run once)

bool BtnPress = true;

while(BtnPress==true){

BtnPress=false;

}

//infinite loop

//will always print "still stuck" until program is forcefully ended

//generally avoided except for user\_control

while(1==1){

Brain.Screen.print("Still stuck");

}

}

For loops – more specific form of loop that is primarily used for iterating (like the first while loop)

//uses 3 steps as the condition

//runs 63 times

for(int i = 0; i<63; i++){

Brain.Screen.print("Hello");

}

//nested for loops

//loops over row and column

for( int r = 0; r<10; r++){

for( int c = 0; c<10; c++){

Brain.Screen.print(r + c);

}

}

**Classes:**

**GUIs (very advanced):**

**Math/Planning:**

**Commenting:**

**Methods/Functions:**

**Pre-Defined Functions:**

**Electronics:**

Yes, you are essentially responsible for this even though you do software not hardware.

With V5 motors wiring into specific ports matters a lot less (as far as I know) so have some consistency.

**UserControl:**

**Autonomous:**

**PreAutonomous:**