ENGR110 T2 2019

Engineering Modeling and Design
Project 2 - Solar tracker. Coding review.

Arthur Roberts

School of Engineering and Computer Science
Victoria University of Wellington

If you can not see opening and corresponding closing curly bracket without scrolling - it is WRONG.
spfinaldone.cpp

Long list of variable declarations

Listing 1: long list of declarations

```
int xMotor = 1;
int yMotor = 5;

//angles of the motors, and the max and min angles they can have
int defaultXSetting = 46;
int defaultYSetting = 54;
int xMotorSetting = 49;
int yMotorSetting = 48;
int motorsMin = 32;
int motorsMax = 65;
double anglePerSetting = 180.0/33.0;
double angleSizeX, angleSizeY;

bool sunIsThere = true;

int mostRedRow = -1;
int mostRedCol = -1;

//boundaries of the middle box
int middleBoxLeft = 140;
int middleBoxRight = 180;
int middleBoxTop = 102;
int middleBoxBottom = 13
```

Make it the class. If you think that class is too much - tthere is a **struct**.

struct is lighter version of the class - no functions, everything public

Listing 2: class

```
class Pixel
{
public:
  int xPos;
  int yPos;
  // constructo should be there
};
```

same as

Listing 3: struct

```
struct Pixel
{ int xPos;
  int yPos;
}; // no need for constructor
```

Listing 4: using struct

```
struct Pixel pix;

pix.xPos = 8;
```

Return the array from the function

C++ does not allow it.
There are ways to do that though.

Can not return it (Counterintuitive) - make it an argument then

Listing 5: Caption

```
#include <iostream>
using namespace std;
void fun(int arr[]){
  arr[0] = 0;
  arr[1] = 1;
  arr[2] = 2;
  arr[3] = 3;
  //return arr;
}

int main(){
  int arr[4] = {0,0,0,0};
  for ( int i = 0 ; i < 4 ;i++){
    cout<<" before: "<<" i = "<<i<<" arr="<<arr[i]<<endl;
  }
  fun(arr); //
  for ( int i = 0 ; i < 4 ;i++){
    cout<<" after: "<<" i = "<<i<<" arr="<<arr[i]<<endl;
  }
}
```

Make it part of the structure - struct is like class without functions and all members public

Listing 6: Caption

```
#include <iostream>
using namespace std;

struct Arr{
  int array[4];
};

Arr fun(Arr a){
  Arr rb;
  rb.array[0] = 0;
  rb.array[1] = 1;
  rb.array[2] = 2;
  rb.array[3] = 3;
  return rb;
}

int main(){
  Arr a;
  a.array[0]= 0; a.array[1]= 0;a.array[2]= 0;a.array[3]= 0;//,0,0,0};
  for ( int i = 0 ; i < 4 ;i++){
   cout<<"a: "<<" i = "<<i<<" arr="<<a.array[i]<<endl; // chaining outputs
  }
  Arr b;
  b = fun(a); // EXPENSIVE in terms of time
  for ( int i = 0 ; i < 4 ;i++){
    cout<<" b: "<<" i = "<<i<<" arr="<<b.array[i]<<endl; // chaining outputs
  }

}
```

If you have constants - make them constant.

Listing 7: Caption

```
// instead of
int motor2 = 5;
// use
const int motor2 = 5;
```

If you try to change (by mistake) value of **motor2** - it will produce error at compile time.