ENGR101: Lecture 8

Missing bits of C++. Project 1 review

March 2023

What we cover today?

- Review of Project1
- Improve your programming style

Review: types

Usually **int** type variables are faster. **double** is only used when really necessary.

```
#include <iostream>
using namespace std;
int main(){
 int a = 1;
 int b = 10;
 double c = a/b;
 cout<<"c="<<c<<endl;
}
```

One would expect c=0.1...

- Trust nothing
- Print everything
- Result of **int** divided by **int** is **int** (even though result of division was declared as double)

Type casting

Variable of one type can be converted (casted) into different type.

```
#include <iostream>
using namespace std;
int main(){
    int a = 1;
    int b = 10;
    double c = (double)a/(double)b;
    cout<<"c="<<c<<endl;
}
```

- There is such thing as **type casting**
- To convert the number from one type to another use **(typeiwant)** construct
- To convert to double, use **(double)**
- To convert to int - **(int)**, fractional part is discarded

What is **char**?

- **char** is one byte big variable
- Can take values from 0000 0000 (0) to 1111 1111(255)
- At the same time it is used to store Engilsh alphabet characters
- Encoding is done by ASCII table.
  `https://www.asciitable.com/`
- If program tries to print it - it becomes character according to ASCII encoding
- In other situations - it is a number. You can add, subtract it.

Working with **chars**. Notice single quotation mark as delimiter for **char**.

```
#include <iostream>

int main(){
  char a; // 1 byte, 8 bits, character
  a = '#';
  char b='-';
  char c = a+b;
  std::cout<<" a+b="<<c<<std::endl;
  return 0;
}
```

Strings

What is string in C++? String is an array(vector) of **chars**.

```
#include <iostream>
#include <string>

int main(){
 std::string str1 = "i am string";
 std::cout<<" str1="<<str1;
 std::cout<<" length="<<str1.length();
 std::cout<<" 3rd element="<<str1[3];
 return 0;
}
```

- **strings** are arrays of characters
- strings are not numbers, even though they can look like numbers
- There are C++ functions which convert strings into numerical values
- That being the challenge - look for these functions yourself. Hint: **stof**, **strtod()**.

How to stop the program?

Sometimes you want to stop the program and see what is happening.
Easy way is to make program wait for user to enter something.
cout operator is for output on the screen. **cin** is to wait for keyboard entry.

```
#include <iostream>

int main(){
  int wait =0;
  // print what you want here
  std::cin>>wait; // prgram stops and waits
  std::cout<<" Entry was "<<wait<<std::endl;
  return 0;
}
```

Vectors

Major advantage of **vector** over **array** is that elements can be added to the vector at run-time.

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
   std::vector<double> v1 = {2.4, 4.4, 5.3, 2.3, 1.4 , 2.1};
   std::cout<<"␣size␣of␣v1="<<v1.size()<<std::endl;
   for (unsigned int i = 0 ; i < v1.size() ; i++){
     std::cout<<"␣␣v1["<<i<<"]="<<v1[i]<<std::endl;
   }
}
```

What is this std:: thing?

std is a namespace. Namespace is a collection of functions, variables, classes and what not. Usually it is big.

```
        #include <iostream>
namespace Electronics{
    struct Stock{
         int TotalStock = 5;
         void print_Stock(){
                   std::cout<<"Electronics TotalStok="<<TotalStock<<std::endl;
        }
    };
    Stock stock;
}

namespace Shoes
{
    struct Stock{
       int TotalStock=10;;
       void print_Stock(){
          std::cout<<"Shoes TotalStok="<<TotalStock<<std::endl;
       }
   };
   Stock stock;
}

using namespace Shoes;

int main(){
```

- Namespace can be big. Names of functions, variables can be same. Use c
- You can avoid typing **std::** if you put **using namespace std;** at the start of the program
- It is not recommended because some

Modulo division

```
#include <iostream>
using namespace std;
int main(){
  double a = 20; //?
   for (int i = 0 ;  i < a ; i++){
    cout<<"i="<<i<<"␣modulo␣="<<i%20<<endl;
  }
}
```

The result of a modulo division is the remainder of an integer division of the given numbers.

$$5\%2 \tag{1}$$

Integer division result is 4. There is 1 **remaining** in original number after the division.

What is this *?

- It is an address of memory cell.
- It was mentioned that address of the variable, say, **a** can be obtained using **&a**.
- Address is a number. We can put it into another memory cell.
- We can not change address by assigning value to it, at least there is this safety guard.
- Why working with addresses is so dangerous? (Mac and Microsoft compilers give you warnings)?
- Program can overwrite something important, like operating system area of memory.

Loop inside the loop - nested

```
#include <iostream>
using namespace std;
int main(){
 for ( int i = 0 ; i < 5; i++){
  for ( int j = 0 ; j< 4 ; j++){
    cout<<" i="<<i<<"  j = "<<j<<endl;
   }
  }
}
```

- Inner loop (j) will repeat for each value of i
- Watch brackets

Shorter version **for** operator

If you want to do some calculation for all elements of the array(vector) you can use shorter version of **for**.

```
#include <iostream>

    int my_array[5] = {2, 3, 4, 5, 4};
    for (int a:my_array){
        std::cout<<a<<std::endl;
    }
}
```