Jason Li
CISC3142
Professor Katherine Chuang

Extra Credit

Access Control - modifiers that define control, meaning it allows access to variables within a program. ETC. Public, Private and Protected

```
public:
        int x;
private:
        int y;
protected:
        int z;
```

Classes - User defined data containing several attributes working as a blueprint to create objects

```
class newClass {
    public:
        string str;
}
```

Class scope - member variables of a class that is declared within the class can only be accessed in a class.

```
class newClass {
    String myStr= "Hello World";
}
```

~~newClass.myStr;~~ Not Possible

Constructor functions - used to create and initializing an object instance of the class;

```
class newClass{
    public:
        newClass(){
            cout << "This is a constructor";
        }
}
```

```
int main(){
        newClass classObj;
        return 0;
}
```

Data Member - variables and functions that are used in a class

```
int main() {
        int num;
        bool flagged;
        float floatNum; //data members
}
```

Default Arguments - a parameter within the original function that is assigned by the compiler if the function call does not provide the value for the argument.

```
int subtract(int x, int y=0){
        return(x-y);
}

int main(){
        subtract(20); // returns 20
        subtract(20,10) // returns 20-10 which is 10
        return 0;
}
```

Destructor function - uses ~(member function) to automatically delete when an object goes out of scope.

```
class newClass{
        public:waa
                String myStr;
                ~String(); // destructor
}
```

Dynamic Allocation - to manually allow memory allocation to be done by the programmer meaning it allows them to set array sizes dynamically during run time rather than when it is compilingsssss
wsa

```
int main() {
        int size;
        int *ptr;

        cout << "Enter size of array" << endl;
        cin << size;
        ptr = new int [size];
}
```

Encapsulation - writing and combining data members/functions in a single class

```
class theShirt {
        public:
                char size;
                string color;

        string getInfo(){
                return "Shirt size: " << size << " Shirt color:" << color;
        }
}
```

File Scope - a variable that is allowed to be accessed by all of the source file through the use of a global variable

```
static int x =100;

int main() {
        x  = x+2;
}
```

Function members - operators and functions that are declared of a class

```
class myClass {
        public:
                int sub()
                {return a+b};
        private:
                int a=20, b=21;
}
```

Function member definition-

```
void myMsg(){
        string msg="Hello World";
        cout << msg;
}
```

Function scope - variables that are allowed only access and used within the scope of the function it was declared/used in.

```
void funcScope() {
        int x =20; //function variable
}

int main(){
        int x =21; // not manipulated
}
```

Inheritance - defining a class in terms of another class

```
class shape{
        ...
}

class triangle :: public shape { // deriving from the class
        ...
}
```

Member Access - determines if a class member is accessible in an expression or declaration

Names - another name for identifiers which are used to identify functions and members

```
int main(){
        int num;
        int Num; // different identifier
}
```

Namespace - provides a scope for identifiers

```
namespace first{
        int val = 20;
}
int main(){
        int val = 200;
        cout << first::val << '\n';
        return 0;
}
```

Objects - stores states in fields and methods expose its behavior

```
class shape{
        ...
}
int main(){
        shape Triangle; // creating an object
}
```

Operator functions - also known as an overloaded function, which is a redefined function that is built in C++

```
class printInfo{
        public:
                void print(str s){
                        cout << "Message: " << s << endl;
                }
}

int main(){
        string world = "Hello World";
        printInfo i;
        i.print(world);
}
```

Overloading - functions that use the same name but with different parameters

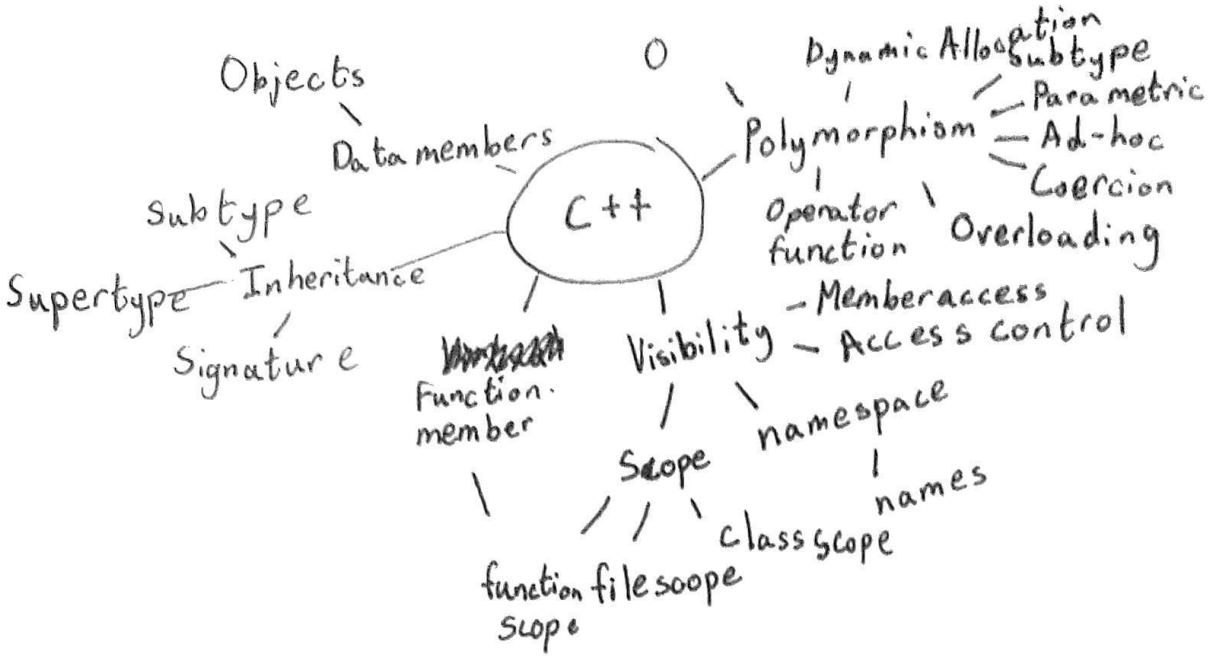
```
struct newStruct{
        int newNum;
};
```

Subtype - a form of data that is a datatype that is related to another datatype

Supertype - a form of data that is the parent of the datatype

Virtual functions - function that is used to be redefined in the derived class.

Visibility - shows whether functions and classes can access certain variables and seeing it values can be changed


C++
Objects
Data members
Subtype
Supertype
Inheritance
Signature
O
Dynamic Allocation
Polymorphism
Subtype
Parametric
Ad-hoc
Coercion
Operator function
Overloading
Visibility
- Member access
- Access control
Function member
Scope
namespace
names
function scope
file scope
class scope