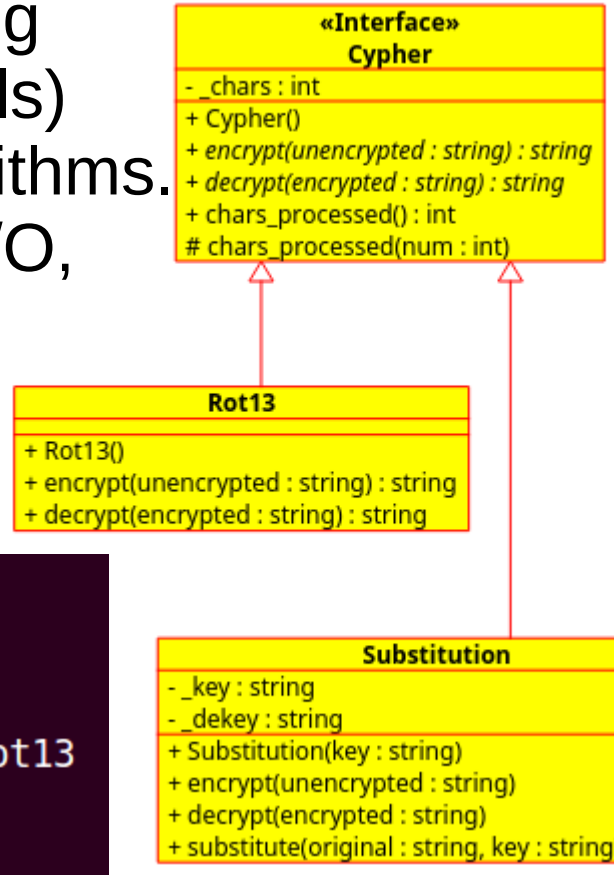
Homework #4
Cypher

This assignment involves writing a cryptography tool implementing multiple algorithms. We'll utilize a pure virtual class (the interface), and from it derive several classes to encrypt or decrypt a user-specified file using the Rot13, Substitution, and (at bonus levels) Exclusive-OR and asynchronous key algorithms. In doing so, we'll practice inheritance, file I/O, and static class members, too!

Due Thursday, February 15 at 8 am.




```
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ ./cypher
Enter filename: main.cpp
Select an encryption algorithm: (R)ot13  (S)ubstitution ==> r
Encrypted 2117 characters.
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ head main.cpp.rot13
#vapyhqr "ebg13.u"
#vapyhqr "fhofgvghgvba.u"
#vapyhqr <ppglcr>
#vapyhqr <vbfgernz>
```

Homework Retrospective
Full Credit – Cypher Class

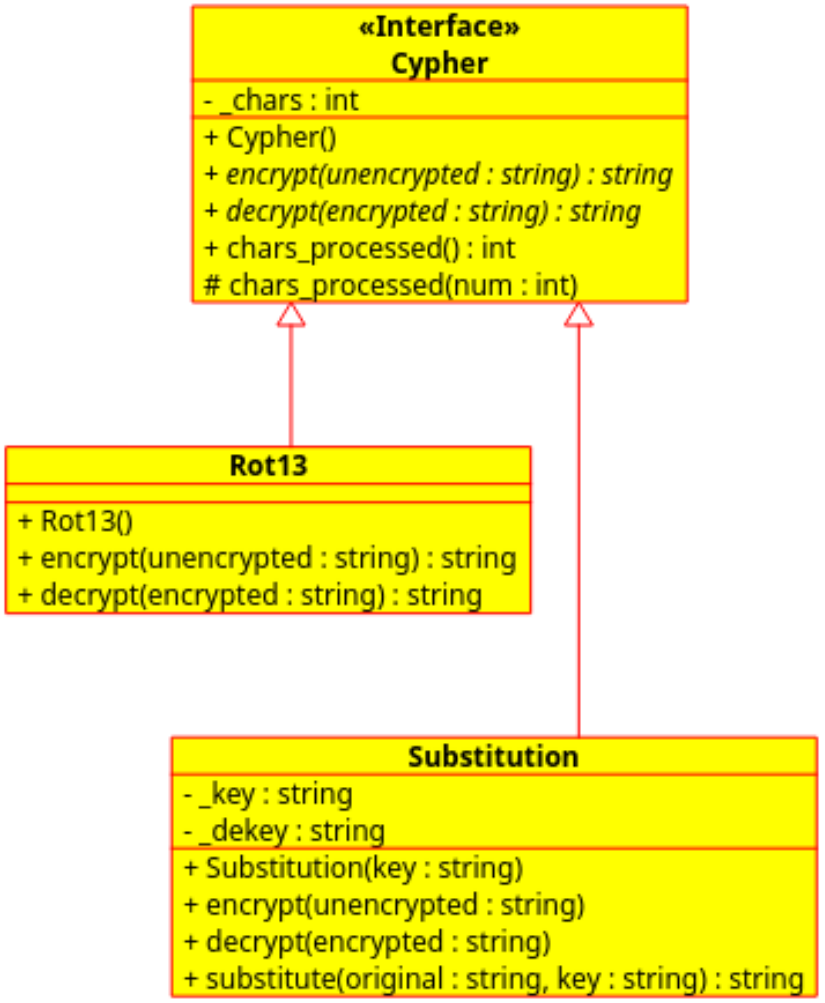
cypher.h

```
#ifndef _CYPHER_H
#define _CYPHER_H

#include <string>
using namespace std;

class Cypher {
  public:
    Cypher();
    virtual string encrypt(string unencrypted) = 0;
    virtual string decrypt(string encrypted) = 0;
    static int chars_encrypted();
  protected:
    static void chars_encrypted(int num);
    static int chars;
};
#endif
```

Explicit constructors weren't required, and it's fine if you didn't include them.



cypher.cpp

```
#include "cypher.h"

Cypher::Cypher() { } // Reserved
int Cypher::chars = 0;
void Cypher::chars_encrypted(int num) {chars += num;}
int Cypher::chars_encrypted() {return chars;}
```

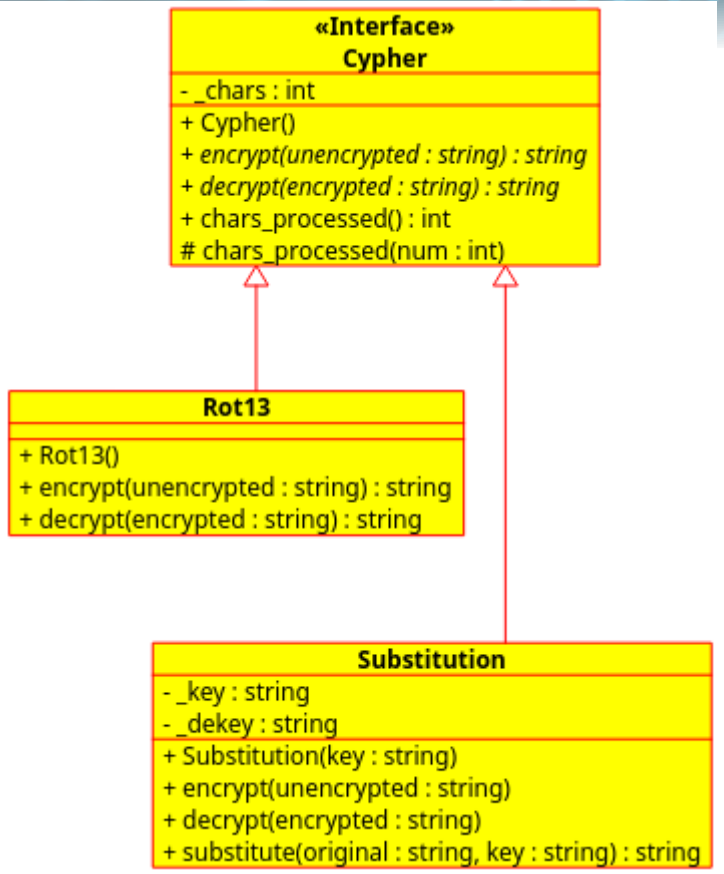
```
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ make cypher.o
g++ --std=c++14 -c cypher.cpp
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$
```

Homework Retrospective
Full Credit – Rot13 Class

rot13.h

```
#ifndef _ROT13_H
#define _ROT13_H
#include "cypher.h"

class Rot13 : public Cypher {
  public:
    Rot13();
    string encrypt(string unencrypted);
    string decrypt(string encrypted);
};
#endif
```



rot13.cpp

```
#include "rot13.h"

Rot13::Rot13() : Cypher() { }
string Rot13::decrypt(string encrypted) {
    return encrypt(encrypted);
}
string Rot13::encrypt(string unencrypted) {
    string encrypted;
    for(char c : unencrypted) {
        if ((c >= 'a' && c <= 'm') ||
            (c >= 'A' && c <= 'M')) encrypted.append(1, c += 13);
        else if ((c >= 'n' && c <= 'z') ||
                 (c >= 'N' && c <= 'Z')) encrypted.append(1, c -= 13);
        else encrypted.append(1, c);
    }
    Cypher::chars_encrypted(unencrypted.size());
    return encrypted;
}
```

Homework Retrospective
Full Credit – Rot13 Regression Test

test.cpp

```
#include "rot13.h"
#include "substitution.h"
#include <iostream>
#include <vector>
using namespace std;

class Test_vector {
  public:
    Test_vector(string unencrypted, string rot13)
        : _unencrypted{unencrypted}, _rot13{rot13} { }
    string unencrypted() {return _unencrypted;}
    string rot13() {return _rot13;}
  private:
    string _unencrypted;
    string _rot13;
};
int main() {
    bool pass = true; // optimistic

    vector<Test_vector> test_vectors = { { {"", "", ""},
        {"The quick brown fox jumps over the lazy dog",
         "Gur dhvpx oebja sbk whzcf bire gur ynml qbt"},
        {"Now is the time for all good students to come to the aid of their gpa",
         "Abj vf gur gvzr sbe nyy tbbq fghqragf gb pbzr gb gur nvq bs gurve tcn"},
        {"English punctuation includes , ; \" ' ! @ # $ % ^ & * ( ) - + = ` ! | \\ / ? . < and >",
         "Ratyvfu chapghngvba vapyhqrf , ; \" ' ! @ # $ % ^ & * ( ) - + = ` ! | \\ / ? . < naq >"},
    };
```

This is called a “test framework”. They make adding more test vectors easy, and thus encourage thorough testing. You were NOT required to use a framework – ad hoc is OK.

Homework Retrospective
Full Credit – Rot13 Regression Test

test.cpp

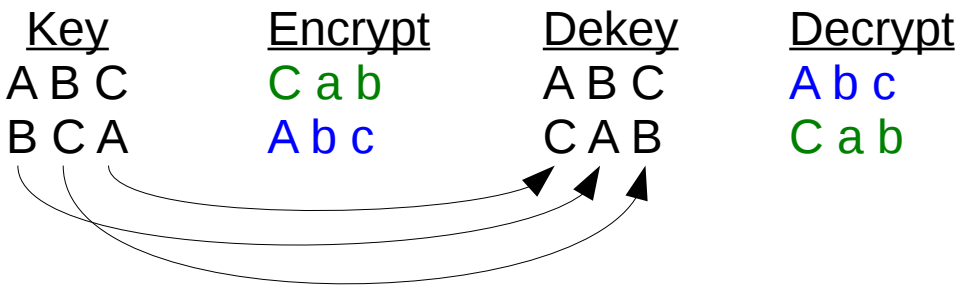
```
    Rot13 rot13;
    for (auto tv : test_vectors) {
        string encrypted = rot13.encrypt(tv.unencrypted());
        string decrypted = rot13.decrypt(encrypted);
        if (tv.rot13() != encrypted) {
           cerr << "Rot13: Encryption error" << endl;
           cerr << "    " << tv.rot13() << endl;
           cerr << "    " << encrypted << endl << endl;
           pass = false;
        }
        if (tv.unencrypted() != decrypted) {
           cerr << "Rot13: Decryption error" << endl;
           cerr << "    " << tv.unencrypted() << endl;
           cerr << "    " << decrypted << endl << endl;
           pass = false;
        }
    }

    if (!pass) {
        cerr << endl << "fail" << endl;
        return -1;
    }
}
```

```
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ make clean
rm -f *.o *~ test interactive cypher
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ make test
g++ --std=c++14 -c test.cpp
g++ --std=c++14 -c cypher.cpp
g++ --std=c++14 -c rot13.cpp
g++ --std=c++14 -c substitution.cpp
g++ --std=c++14 -o test test.o cypher.o rot13.o substitution.o
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ ./test
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$
```

Deeper Thoughts on Substitution Cyphers

- Rot13 is just a Substitution cypher with key "nopqrstuvwxyzabcdefghijklm"
  - If you implemented Rot13 this way, good for you!
- Decrypting via substitution is the same as Encrypting with the “inverse key”



- Thus, you can implement the substitution algorithm once, and use it to encrypt *and* decrypt
  - Or, you could implement separately at no penalty

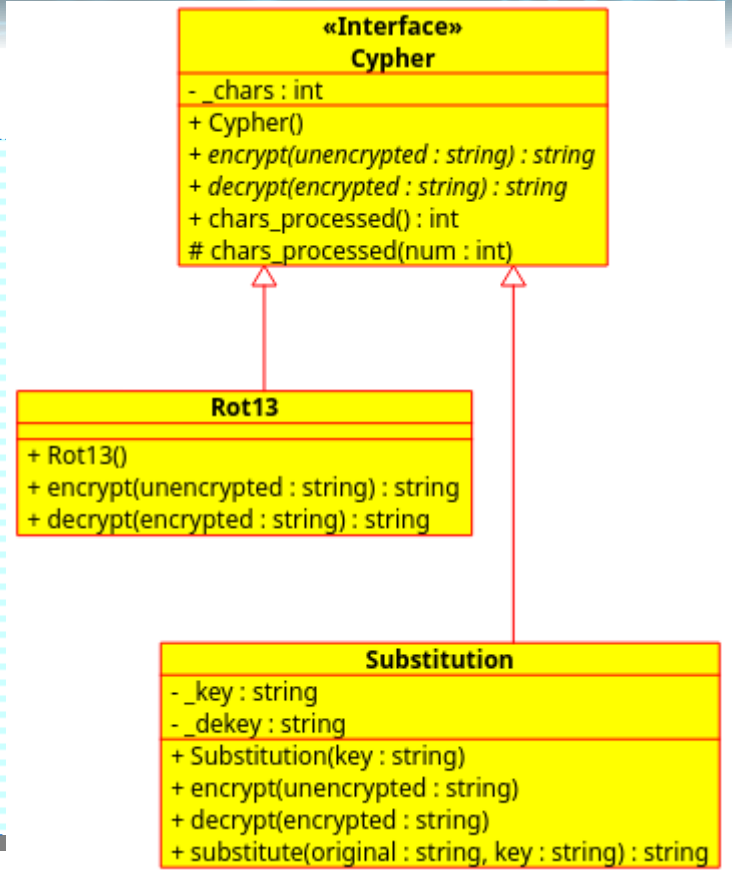
Homework Retrospective
Full Credit – Substitution Class

substitution.h

```
#ifndef _SUBSTITUTION_H
#define _SUBSTITUTION_H

#include "cypher.h"

class Substitution : public Cypher {
  public:
    Substitution(string key);
    string encrypt(string unencrypted);
    string decrypt(string encrypted);
  private:
    string substitute(string original, string key);
    string _key;
    string _dekey;  // inverse of _key
};
#endif
```



Homework Retrospective
Full Credit – Substitution Class

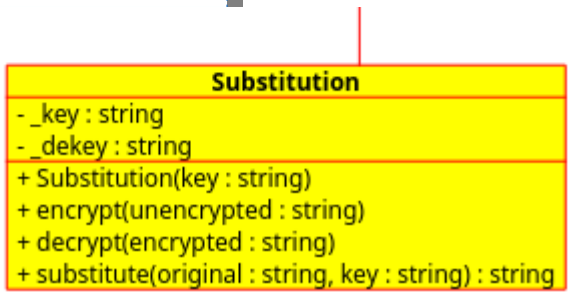
substitution.cpp

```
#include "substitution.h"
#include <algorithm>
#include <cassert>

Substitution::Substitution(string key) : Cypher(), _key{key}, _dekey{key}
{
  // Data validation: length of key
  assert(_key.size() == 26);

  // Data validation: key has exactly one of each lowercase character
  string sorted_key = key;
  sort(sorted_key.begin(), sorted_key.end());
  assert(sorted_key == "abcdefghijklmnopqrstuvwxyz");

  // Create reverse key for decryption
  char k = 'a';
  for(char c : _key) _dekey[c-'a'] = k++;
}
```



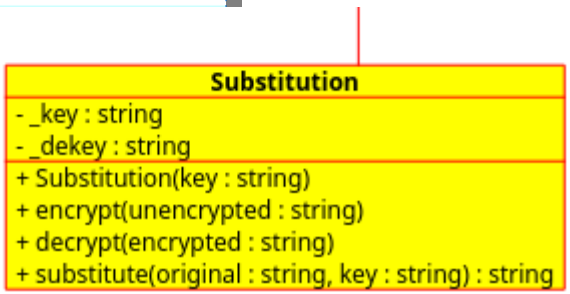
Homework Retrospective
Full Credit – Substitution Class

substitution.cpp

```
string Substitution::substitute(string original, string key) {
    string encrypted;
    for(char c : original) {
        int capitalization = 0;
        if (c >= 'A' && c <= 'Z') {
          capitalization = 'A' - 'a';
          c -= capitalization;
        }
        if (c >= 'a' && c <= 'z')
          encrypted.append(1, key[c - 'a'] + capitalization);
        else
          encrypted.append(1, c);
    }
    chars_encrypted(original.size());
    return encrypted;
}

string Substitution::encrypt(string unencrypted) {
    return substitute(unencrypted, _key);
}

string Substitution::decrypt(string encrypted) {
    return substitute(encrypted, _dekey);
}
```



Homework Retrospective
Full Credit – Substitution Test

test.cpp

```
class Test_vector {
  public:
    Test_vector(string unencrypted, string rot13, string substitution)
        : _unencrypted{unencrypted}, _rot13{rot13}, _substitution{substitution} { }
    string unencrypted() {return _unencrypted;}
    string rot13() {return _rot13;}
    string substitution() {return _substitution;}
  private:
    string _unencrypted;
    string _rot13;
    string _substitution;
};

int main() {
    bool pass = true; // optimistic

    vector<Test_vector> test_vectors = { {"", "", ""},
        {"The quick brown fox jumps over the lazy dog",
         "Gur dhvpx oebja sbk whzcf bire gur ynml qbt",
         "Sem lakdx ftrzq ory iawcp rumt sem nbgv hrj",},
        {"Now is the time for all good students to come to the aid of their gpa",
         "Abj vf gur gvzr sbe nyy tbbq fghqragf gb pbzr gb gur nvq bs gurve tcn",
         "Qrz kp sem skwm ort bnn jrrh psahmqsp sr drwm sr sem bkh ro semkt jcb",},
        {"English punctuation includes , ; \" ' ! @ # $ % ^ & * ( ) - + = ` ! | \\ / ? . < and >",
         "Ratyvfu chapghngvba vapyhqrf , ; \" ' ! @ # $ % ^ & * ( ) - + = ` ! | \\ / ? . < naq >",
         "Mqjnkpe caqdsabskrq kqdnahmp , ; \" ' ! @ # $ % ^ & * ( ) - + = ` ! | \\ / ? . < bqh >",},
    };
```

The test framework can be easily extended to also test substitution algorithms.

Homework Retrospective
Full Credit – Substitution Test

test.cpp

```
    string key = "bfdhmojekixnwqrcltpsauzyvg";
    Substitution substitution{key};
    for (auto tv : test_vectors) {
        string encrypted = substitution.encrypt(tv.unencrypted());
        string decrypted = substitution.decrypt(encrypted);
        if (tv.substitution() != encrypted) {
            cerr << "Substitution: Encryption error" << endl;
            cerr << "    " << tv.substitution() << endl;
            cerr << "    " << encrypted << endl << endl;
            pass = false;
        }
        if (tv.unencrypted() != decrypted) {
            cerr << "Substitution: Decryption error" << endl;
            cerr << "    " << tv.unencrypted() << endl;
            cerr << "    " << decrypted << endl << endl;
            pass = false;
        }
    }

    if (!pass) {
        cerr << endl << "fail" <<
        return -1;
    }
}
```

```
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ make clean
rm -f *.o *~ test interactive cypher
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ make test
g++ --std=c++14 -c test.cpp
g++ --std=c++14 -c cypher.cpp
g++ --std=c++14 -c rot13.cpp
g++ --std=c++14 -c substitution.cpp
g++ --std=c++14 -o test test.o cypher.o rot13.o substitution.o
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ ./test
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$
```

Homework Retrospective
Full Credit – Interactive Test

- The suggested solution also includes an interactive test for the algorithms

NOT required,
but pretty useful!

```
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ make interactive
g++ --std=c++14 -c interactive.cpp
g++ --std=c++14 -c cypher.cpp
g++ --std=c++14 -c rot13.cpp
g++ --std=c++14 -c substitution.cpp
g++ --std=c++14 -o interactive interactive.o cypher.o rot13.o substitution.o
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ ./interactive
(E)ncrypt or (D)ecrypt? e
Enter text to encrypt: Where in the World is Carmen San Diego?
Select an encryption algorithm
  (R)ot13
  (S)ubstitution
> s
Enter a 26-character key (Enter for default):
Zemtm kq sem Zrtnh kp Dbtwmq Pbq Hkmjr?
Encrypted 39 characters.
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ ./interactive
(E)ncrypt or (D)ecrypt? d
Enter text to decrypt: Zemtm kq sem Zrtnh kp Dbtwmq Pbq Hkmjr?
Select an encryption algorithm
  (R)ot13
  (S)ubstitution
> s
Enter a 26-character key (Enter for default):
Where in the World is Carmen San Diego?
Decrypted 39 characters.
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$
```

Homework Retrospective
Full Credit – Main

main.cpp

```
//includes omitted
int main() {
  string filename, filename_out, aline;
  char algorithm = ' ';
  bool do_encrypt = true;
  string key = "bfdhmojekixnwqrcltpsauzyvg";

  cout << "Enter filename: ";
  getline(cin, filename);

  if (filename.size() < 6) {
    do_encrypt = true;
  } else if (filename.substr(filename.size()-6) == ".rot13") {
    do_encrypt = false;
    algorithm = 'r';
  } else if (filename.substr(filename.size()-6) == ".subst") {
    do_encrypt = false;
    algorithm = 's';
  } else {
    do_encrypt = true;
    while (true) {
        cout << "Select an encryption algorithm: (R)ot13  (S)ubstitution ==> ";
        cin >> algorithm;
        cin.ignore();
        algorithm = tolower(algorithm);
        if (algorithm == 'r' or algorithm == 's') break;
        else cerr << "### Invalid key: Please press 'r' or 's'" << endl;
    }
  }
```

We can determine whether to encrypt or decrypt based on the filename extension.
For decryption, we also know the algorithm, but for encryption we have to ask.

Homework Retrospective
Full Credit – Main

main.cpp

```
  if (do_encrypt)
    filename_out = filename + ((algorithm == 'r') ? ".rot13" : ".subst");
  else
    filename_out = filename.substr(0, filename.size() - 6);
  ifstream ifs{filename};
  ofstream ofs{filename_out};

  if (algorithm == 'r') {
       Rot13 rot13;
       while(getline(ifs, aline))
           ofs << (do_encrypt ? rot13.encrypt(aline) : rot13.decrypt(aline)) << endl;
       cerr << (do_encrypt ? "Encrypted " : "Decrypted ")
            << rot13.chars_encrypted() << " characters." << endl;
  } else if (algorithm == 's') {
       while (true) {
           cout << "Enter a 26-character key (Enter for default): ";
           string temp_key;
           getline(cin, temp_key);
           if (temp_key.size() == 26) {key = temp_key; break;}
           else if (temp_key.size() == 0) break;
           else cerr << "Invalid key" << endl;
       }
       Substitution sub{key};
       while(getline(ifs, aline))
           ofs << (do_encrypt ? sub.encrypt(aline) : sub.decrypt(aline)) << endl;
       cerr << (do_encrypt ? "Encrypted " : "Decrypted ")
            << sub.chars_encrypted() << " characters." << endl;
   }
}
```

Homework Retrospective
Full Credit – Makefile

Makefile

```
# Makefile for Cypher
CXXFLAGS += --std=c++14

all: main test interactive

debug: CXXFLAGS += -g
debug: all

main: main.o cypher.o rot13.o substitution.o *.h
	$(CXX) $(CXXFLAGS) -o cypher main.o cypher.o rot13.o substitution.o
main.o: main.cpp *.h
	$(CXX) $(CXXFLAGS) -c main.cpp
test: test.o cypher.o rot13.o substitution.o *.h
	$(CXX) $(CXXFLAGS) -o test test.o cypher.o rot13.o substitution.o
test.o: test.cpp *.h
	$(CXX) $(CXXFLAGS) -c test.cpp
cypher.o: cypher.cpp *.h
	$(CXX) $(CXXFLAGS) -c cypher.cpp
rot13.o: rot13.cpp *.h
	$(CXX) $(CXXFLAGS) -c rot13.cpp
substitution.o: substitution.cpp *.h
	$(CXX) $(CXXFLAGS) -c substitution.cpp
interactive: interactive.o cypher.o rot13.o substitution.o *.h
	$(CXX) $(CXXFLAGS) -o interactive interactive.o cypher.o rot13.o substitution.o
interactive.o: interactive.cpp *.h
	$(CXX) $(CXXFLAGS) -c interactive.cpp
clean:
	-rm -f *.o *.gch *~ test interactive cypher
```

Example File Encrypt / Decrypt

```
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ make
g++ --std=c++14 -c main.cpp
g++ --std=c++14 -c cypher.cpp
g++ --std=c++14 -c rot13.cpp
g++ --std=c++14 -c substitution.cpp
g++ --std=c++14 -o cypher main.o cypher.o rot13.o substitution.o
g++ --std=c++14 -c test.cpp
g++ --std=c++14 -o test test.o cypher.o rot13.o substitution.o
g++ --std=c++14 -c interactive.cpp
g++ --std=c++14 -o interactive interactive.o cypher.o rot13.o substitution.o
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ cp main.cpp test.txt
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ ./cypher
Enter filename: test.txt
Select an encryption algorithm: (R)ot13  (S)ubstitution ==> s
Enter a 26-character key (Enter for default):
Encrypted 2117 characters.
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ head test.txt.subst
#kqdnahm "trs13.e"
#kqdnahm "pafpsksaskrq.e"
#kqdnahm <ddsvcm>
#kqdnahm <krpstmbw>
#kqdnahm <opstmbw>
#kqdnahm <ddsvcm>  // srnrzmt
apkqj qbwmpcbdm psh;

kqs wbkq() {
  pstkqj oknmqbwm;
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ rm test.txt
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ ./cypher
Enter filename: test.txt.subst
Enter a 26-character key (Enter for default):
Decrypted 2117 characters.
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$ diff test.txt main.cpp
ricegf@pluto:~/dev/cpp/201801/P4/full_credit$
```

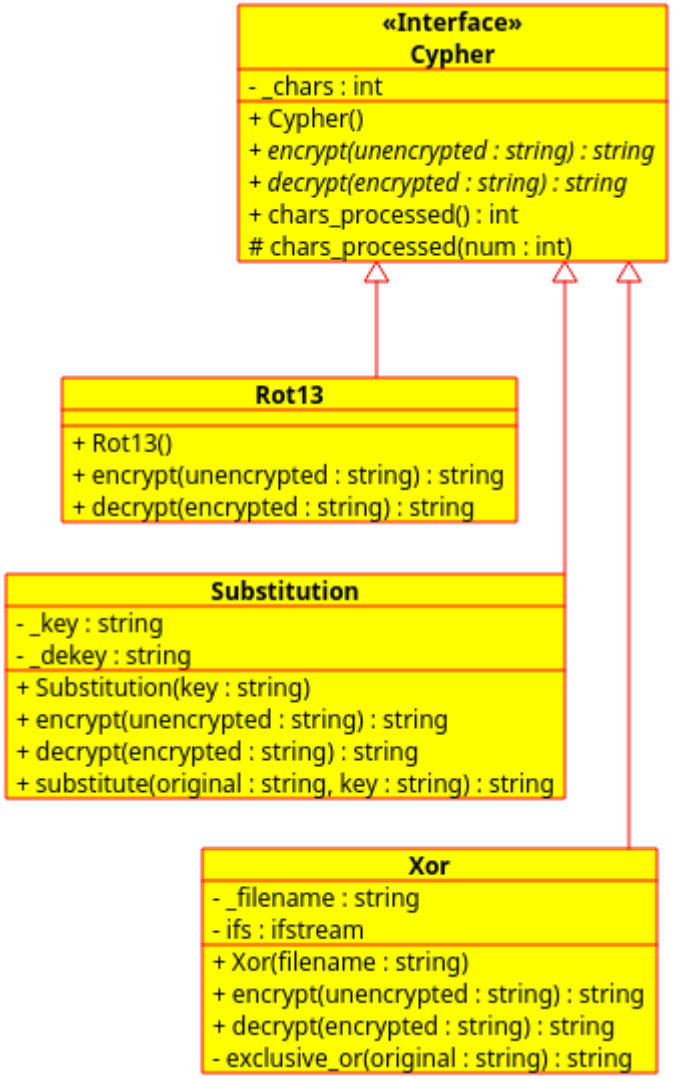
Bonus – Exclusive Or

xor.h

```
#ifndef _XOR_H
#define _XOR_H

#include "cypher.h"
#include <fstream>

class Xor : public Cypher {
  public:
    Xor(string filename);
    string encrypt(string unencrypted);
    string decrypt(string encrypted);
  private:
    string exclusive_or(string original);
    string _filename;
    ifstream _ifs;
};
#endif
```



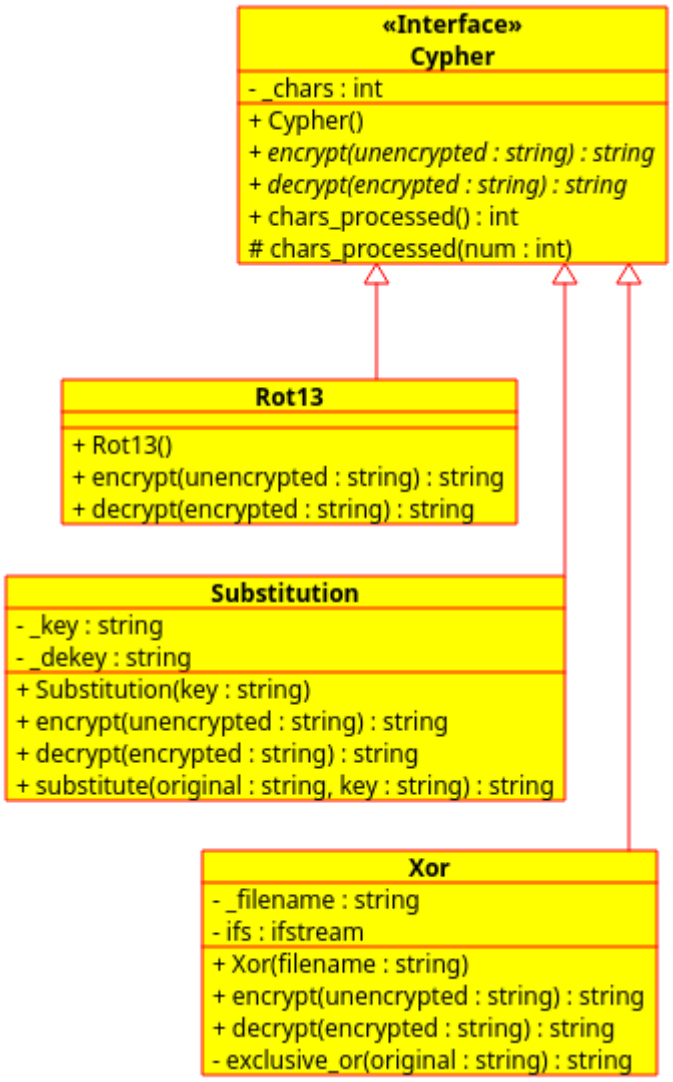
Bonus – Exclusive Or

xor.cpp

```
#include "xor.h"
#include <iostream>

Xor::Xor(string filename) : Cypher(),
_filename{filename},
    _ifs{_filename, ios::binary} { }

string Xor::exclusive_or(string original) {
    string encrypted;
    char from_file;
    for(char c : original) {
        if (_ifs.eof()) {
            _ifs.clear(); // clear eof condition
            _ifs.seekg(0, ios::beg);
        }
        _ifs.get(from_file);
        encrypted.append(1, c^(from_file|0x80));
    }
    chars_encrypted(original.size());
    return encrypted;
}
string Xor::encrypt(string unencrypted) {
    return exclusive_or(unencrypted);
}
string Xor::decrypt(string encrypted) {
    return exclusive_or(encrypted);
}
```



Bonus – Exclusive Or

main.cpp

```
// ...
  string filename_key = "main.cpp";  // for Xor (any file will do, but zip or image
                                     // files are preferred – better byte variety
// ...

  if (do_encrypt)
    filename_out = filename + ((algorithm == 'r') ? ".rot13" :
                               ((algorithm == 's') ? ".subst" : ".xor"));
  else
    filename_out = filename.substr(0, filename.size() - ((algorithm == 'x') ? 4 : 6));

// ...

  } else if (algorithm == 'x') {
      cout << "Enter a filename to use as the key (Enter for default): ";
      string filename_temp;
      getline(cin, filename_temp);
      if (filename_temp.size() > 0) filename_key = filename_temp;

      Xor exor{filename_key};
      while(getline(ifs, aline))
          ofs << (do_encrypt ? exor.encrypt(aline) : exor.decrypt(aline)) << endl;
      cerr << (do_encrypt ? "Encrypted " : "Decrypted ")
           << exor.chars_encrypted() << " characters." << endl;
  }
```

Example File Encrypt / Decrypt

```
ricegf@pluto:~/dev/cpp/201801/P4/bonus$ make main
g++ --std=c++14 -c main.cpp
g++ --std=c++14 -c cypher.cpp
g++ --std=c++14 -c rot13.cpp
g++ --std=c++14 -c xor.cpp
g++ --std=c++14 -c substitution.cpp
g++ --std=c++14 -o cypher main.o cypher.o rot13.o xor.o substitution.o
ricegf@pluto:~/dev/cpp/201801/P4/bonus$ cp main.cpp test.txt
ricegf@pluto:~/dev/cpp/201801/P4/bonus$ ./cypher
Enter filename: test.txt
Select an encryption algorithm
  (R)ot13
  (S)ubstitution
  (X)or
> x
Enter a filename to use as the key (Enter for default): P4.zip
Encrypted 3042 characters.
ricegf@pluto:~/dev/cpp/201801/P4/bonus$ head test.txt.xor
[illegible]

ricegf@pluto:~/dev/cpp/201801/P4/bonus$ rm test.txt
ricegf@pluto:~/dev/cpp/201801/P4/bonus$ ./cypher
Enter filename: test.txt.xor
Enter a filename to use as the key (Enter for default): P4.zip
Decrypted 3042 characters.
ricegf@pluto:~/dev/cpp/201801/P4/bonus$ diff main.cpp test.txt
ricegf@pluto:~/dev/cpp/201801/P4/bonus$
```

Extreme Bonus – Async

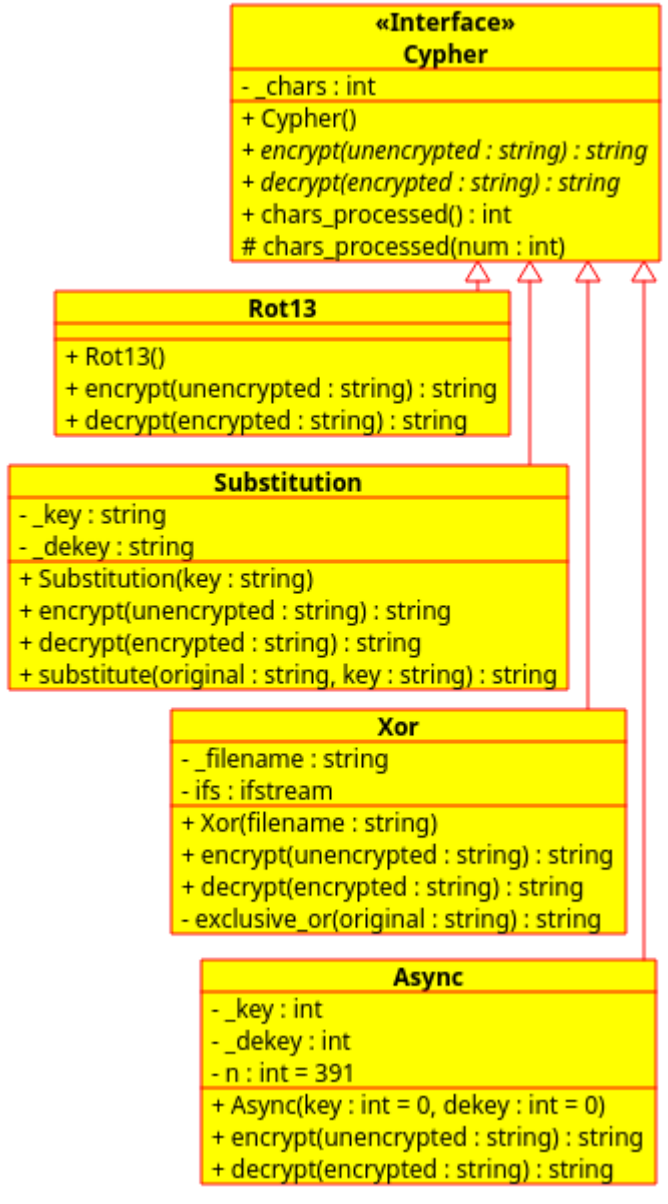
async.h

```
#ifndef _ASYNC_H
#define _ASYNC_H

// Algorithm adapted from
// https://stackoverflow.com/questions/10005124/public-
private-key-encryption-tutorials

#include "cypher.h"
#include <functional>

class Async : public Cypher {
  public:
    Async(int key = 47, int dekey = 15);
    string encrypt(string unencrypted);
    string decrypt(string encrypted);
  private:
    int _key;
    int _dekey;
};
#endif
```

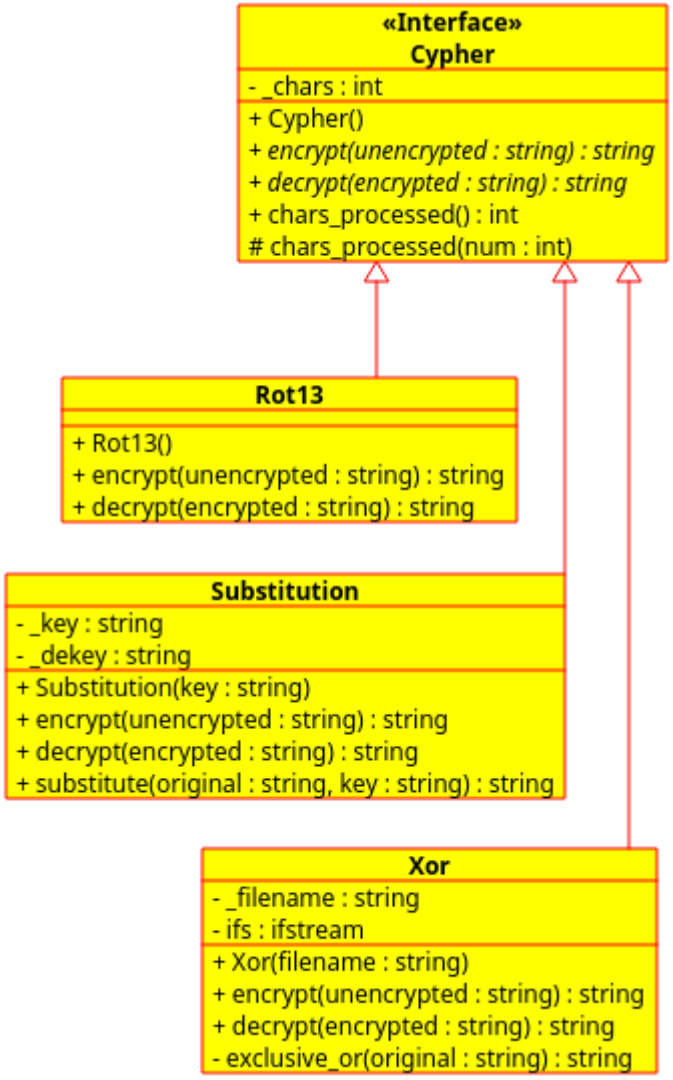


Extreme Bonus – Async

async.cpp

```
Async::Async(int key, int dekey) : Cypher(), _key{key},
_dekey{dekey} { }

struct crypt : std::binary_function<int, int, int> {
    int operator()(int input, int key) const {
        int n=391;
        int result = 1;
        for (int i=0; i<key; i++) {
            result *= input;
            result %= n;
        }
        return result;
    }
};
```

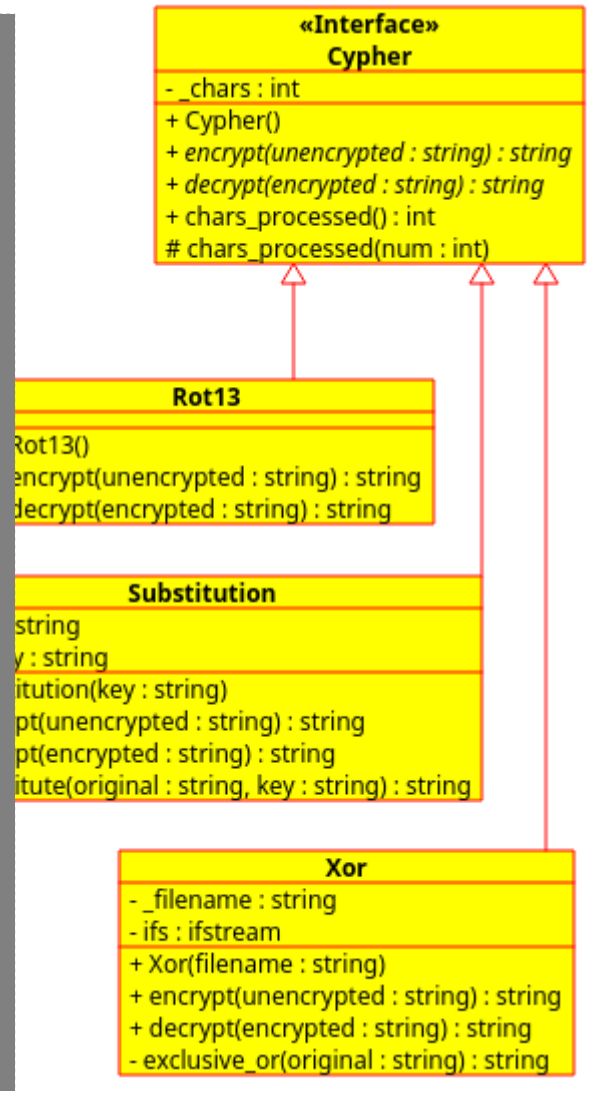


Extreme Bonus – Async

async.cpp

```
string Async::encrypt(string unencrypted) {
    std::vector<int> encrypted;
    string result;
    std::transform(unencrypted.begin(),
        unencrypted.end(),
        std::back_inserter(encrypted),
        std::bind2nd(crypt(), _key));
    for (int i : encrypted) result += " " + std::to_string(i);
    chars_encrypted(unencrypted.size());
    return result;
}

string Async::decrypt(string encrypted) {
    std::vector<int> v_encrypted;
    int i;
    istringstream iss{encrypted};
    ostringstream result;
    while(iss >> i) v_encrypted.push_back(i);
    std::transform(v_encrypted.begin(), v_encrypted.end(),
        std::ostream_iterator<char>(result, ""),
        std::bind2nd(crypt(), _dekey));
    chars_encrypted(result.str().size());
    return result.str();
}
```



Extreme Bonus – Async

main.cpp

```
// ...

  if (do_encrypt)
    filename_out = filename + ((algorithm == 'r') ? ".rot13" :
                               ((algorithm == 's') ? ".subst" :
                               ((algorithm == 'x') ? ".xor" : ".async")));
  else
    filename_out = filename.substr(0, filename.size() - ((algorithm == 'x') ? 4 : 6));

// ...

  } else if (algorithm == 'a') {
       Async async;
       while(getline(ifs, aline))
           ofs << (do_encrypt ? async.encrypt(aline) : async.decrypt(aline)) << endl;
       cerr << (do_encrypt ? "Encrypted " : "Decrypted ")
            << async.chars_encrypted() << " characters." << endl;
  }
```

Example File Encrypt / Decrypt

```
ricegf@pluto:~/dev/cpp/201801/P4/extreme_bonus$ make
g++ --std=c++14 -c main.cpp
g++ --std=c++14 -c cypher.cpp
g++ --std=c++14 -c rot13.cpp
g++ --std=c++14 -c substitution.cpp
g++ --std=c++14 -c xor.cpp
g++ --std=c++14 -c async.cpp
g++ --std=c++14 -o cypher main.o cypher.o rot13.o substitution.o xor.o async.o
g++ --std=c++14 -c test.cpp
g++ --std=c++14 -o test test.o cypher.o rot13.o substitution.o xor.o
g++ --std=c++14 -c interactive.cpp
g++ --std=c++14 -o interactive interactive.o cypher.o rot13.o substitution.o
ricegf@pluto:~/dev/cpp/201801/P4/extreme_bonus$ cp main.cpp test.txt
ricegf@pluto:~/dev/cpp/201801/P4/extreme_bonus$ ./cypher
Enter filename: test.txt
Select an encryption algorithm
  (R)ot13
  (S)ubstitution
  (X)or
  (A)sync
> a
Encrypted 3626 characters.
ricegf@pluto:~/dev/cpp/201801/P4/extreme_bonus$ head test.txt.async
 256 380 151 113 71 8 144 16 246 204 367 189 300 280 102 299 26 204
 256 380 151 113 71 8 144 16 246 204 171 189 367 299 26 204
 256 380 151 113 71 8 144 16 246 204 276 8 55 276 300 380 300 8 300 380 189 151 299 26 204
 256 380 151 113 71 8 144 16 246 204 10 276 9 151 113 299 26 204
 256 380 151 113 71 8 144 16 246 53 113 113 300 9 318 16 48
 256 380 151 113 71 8 144 16 246 53 380 189 276 300 367 16 10 175 48
 256 380 151 113 71 8 144 16 246 53 34 276 300 367 16 10 175 48
 256 380 151 113 71 8 144 16 246 53 113 113 300 9 318 16 48 246 246 208 208 246 300 189 71 189
 340 16 367
 8 276 380 151 273 246 151 10 175 16 276 318 10 113 16 246 276 300 144 219

ricegf@pluto:~/dev/cpp/201801/P4/extreme_bonus$ rm test.txt
ricegf@pluto:~/dev/cpp/201801/P4/extreme_bonus$ ./cypher
Enter filename: test.txt.async
Decrypted 3626 characters.
ricegf@pluto:~/dev/cpp/201801/P4/extreme_bonus$ diff main.cpp test.txt
ricegf@pluto:~/dev/cpp/201801/P4/extreme_bonus$
```

Important Note

- These are all “toy encryptions”
  - They don’t protect anything at all
  - They are only an educational exercise
- If you have information to protect, use *professional* grade encryption
  - Never, *ever* “roll your own”
- https://www.cryptopp.com/wiki/Security_Level has more information for the interested student