Assignment_1_PROG3

Assignment 1 Solutions

Wolff Andreas Erhard

Exercise 1.1

```cpp
#include <cmath>
#include <iostream>

double f(double x) { return exp(-x * x); }

double integral(double a, double b, int n) {
  double totalArea = 0.0;
  double width = (b - a) / n;

  for (int i = 0; i < n; ++i) {
    double midPoint = a + (i + 0.5) * width;
    totalArea += f(midPoint) * width;
  }

  return totalArea;
}

void integration_test(double from, double to, int chunks) {
  double result = integral(from, to, chunks);
  std::cout << "integration res: " << result << std::endl;
}
```

Exercise 1.2

```cpp
#include <algorithm>
#include <cctype>
#include <iostream>
#include <string>

void stripWhitespaceFromString(std::string &text) {
  text.erase(std::remove_if(text.begin(), text.end(), ::isspace),
text.end());
}
```

```
std::string toLower(std::string s) {
  std::transform(s.cbegin(), s.cend(), s.begin(),
                 [](unsigned char c) { return std::tolower(c); });
  return s;
}

std::string toUpper(std::string s) {
  std::transform(s.cbegin(), s.cend(), s.begin(),
                 [](unsigned char c) { return std::toupper(c); });
  return s;
}

std::vector<int> whiteSpacePositions(std::string text) {
  std::vector<int> wsPositions;
  int idx = 0;
  (void)std::count_if(text.begin(), text.end(),
                      [&wsPositions, &idx](unsigned char c) {
                        if (std::isspace(c)) {
                          wsPositions.push_back(idx);
                        }
                        idx++;
                        return std::isspace(c);
                      });

  return wsPositions;
}

void assureEqualLengthOfStrings(std::string &plaintext,
                                std::string &encryption_key) {
  for (int i = 0; i < plaintext.length(); ++i) {
    if (encryption_key.length() < plaintext.length()) {
      encryption_key += encryption_key[i % plaintext.length()];
    }
  }
}

std::string text, enc_key;

void vigenere_encrypt() {
  std::cout << "Enter the text that you want to be encrypted: ";
  std::getline(std::cin, text);
  std::cout << "Enter encryption keyword: ";
  std::getline(std::cin, enc_key);

  std::vector<int> wsPositions = whiteSpacePositions(text);
```

```
  stripWhitespaceFromString(text);
  stripWhitespaceFromString(enc_key);

  assureEqualLengthOfStrings(text, enc_key);

  std::string encrypted_text;
  for (int i = 0; i < text.length(); i++) {
    char x = ((toLower(text)[i] - 'a') + (toUpper(enc_key)[i] - 'A')) % 26;
    x += 'A';
    encrypted_text.push_back(x);
  }

  for (int pos : wsPositions) {
    encrypted_text.insert(pos, " ");
  }

  std::cout << "Encrypted Text: " << encrypted_text << std::endl;
}

void vigenere_decrypt() {
  std::cout << "Enter the text that you want to be decrypted: ";
  std::getline(std::cin, text);
  std::cout << "Enter encryption keyword: ";
  std::getline(std::cin, enc_key);

  std::vector<int> wsPositions = whiteSpacePositions(text);

  stripWhitespaceFromString(text);
  stripWhitespaceFromString(enc_key);

  assureEqualLengthOfStrings(text, enc_key);

  std::string decrypted_text;
  for (int i = 0; i < text.length(); i++) {
    char x = ((toLower(text)[i] - 'a') - (toUpper(enc_key)[i] - 'A') + 26) %
26;
    x += 'a';
    decrypted_text.push_back(x);
  }

  for (int pos : wsPositions) {
    decrypted_text.insert(pos, " ");
  }

  std::cout << "Decrypted Text: " << decrypted_text << std::endl;
```

```
}

void vigenere_cypher(int process) {
  switch (process) {
  case 1:
    vigenere_encrypt();
    break;

  case 2:
    vigenere_decrypt();
    break;

  default:
    std::cout << "⟶ Invalid argument value. The Vigenere Cypher requires
an "
                 "argument (1 for encryption, 2 for decryption)."
              << std::endl;
    break;
  }
}
```

Exercise 1.3

```
#define SQUARESUM(a, b) ((a) * (a) + (b) * (b));
```

Exercise 1.4

```
class CA
{
  public:
    int ia1;

  private:
    int ia2;
    void ma1()
    {
      ia1 = 10;
    }

  public:
    void ma2(CA obj)
    {
      obj.ia1 = 20;
      ia2 = obj.ia2;
```

```
    }
};

class CB
{
  private:
    CA *obja;

  public:
    CB()
    {
      obja = new CA();
    }

    void mb()
    {
      obja->ia1 = 11;
      obja->ia2 = 22;
      obja->ma1();
      obja->ma2(*obja);
    }
};

/*
Answer:
Class CB is wrong because we are instantiating a new CA object inside then
trying to access its private members and methods (ia2 and ma1();).
*/
```

Exercise 1.5

```
class KL1
{
  private:
    string name;

  public:
    int i;
    KL1()
    {
      name = "### not defined ###";
      i = 0;
    }
    KL1(string n, int v) : name(n), i(v) {}
```

```
    void print()
    {
      cout << name << ": " << i << endl;
    }
};

class KL2
{
  private: void meth2(KL1 &ok, int wert)
    {
      ok.i = wert;
      wert = 888;
    }

  public:
    void test()
    {
      KL1 k0("obj1", 10);
      KL1 k1("obj2", 20);
      KL1 &k2 = k1;
      KL1 feld[3];
      feld[0] = k0;
      feld[1] = k1;
      feld[2] = k2;
      feld[0].print();
      feld[1].print();
      feld[2].print();
      int j = 42;
      meth2(feld[0], j);
      feld[2].i = 999;
      feld[0].print();
      feld[1].print();
      feld[2].print();
      cout << "j = " << j << endl;
    }
};

/*
Q: Which output is generated by the following call to the test method:
  KL2 o2;
  o2.test();

A: 10, 20, 20, 42, 20, 999, 42
*/
```