Чернівецький національний університет імені Юрія Федьковича

ННІФТКН

Кафедра Математичних проблем управління і кібернетики

Лабораторна робота № 1

з дисципліни

« *Об’єктно-орієнтоване програмування мовою С++***»**

*«Побітові операції.»*

Варіант № 6

Виконав: студент 141 групи

ПІБ: Сватенко Степан Олександрович

Перевірив: Ассистент, PhD Літвінчук Ю. А.

Чернівці – 2025

**Github: https://github.com/VLazorykOOP/lab2oop-24-UnknownStepan**

**Завдання 1.**

Задано цілі числа a, b, c та d. Обчислити вираз без використання операцій множення та ділення(замінивши на їх операцій зсувів).



int multiply(int a, int b) {

int result = 0;

bool is\_Negative = false;

is\_Negative = (((a < 0) and (b < 0)) or ((a > 0) and (b > 0))) ? false : true;

a = (a < 0) ? -a : a;

b = (b < 0) ? -b : b;

while (b != 0) {

if (b & 1) {

result += a;

}

a <<= 1;

b >>= 1;

}

if (is\_Negative) {

result = -result;

}

return result;

}

int divide(int a, int b) {

if (b == 0) {

return 0;

}

int result = 0;

int sign = ((a < 0) and (b < 0)) or ((a > 0) and (b > 0)) ? 1 : -1;

unsigned int a\_de = (a < 0) ? -a : a;

unsigned int b\_do = (b < 0) ? -b : b;

while (a\_de >= b\_do) {

int temp = b\_do;

int multiplier = 1;

while ((temp << 1) <= a\_de) {

temp <<= 1;

multiplier <<= 1;

}

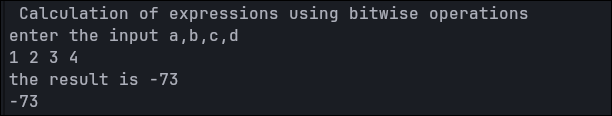
a\_de -= temp;

result += multiplier;

}

return result \* sign;

}



Операцію множення можна описати додаванням, але так як у нас число в двійковому вигляді, і використовувати в просту додавання n-кількості разів, неможливе. Тому використовуючи 2 тип переведення, двійкове число, можна описати у вигляді доданків степенів двійки, пропускаючи всіх парних доданків, ми позбавляємо всіх нулів, і зайвих множників.

Ділення зводиться до віднімання максимального вмістимого в ділене число, та множення відповідно дільника, без внутрішнього циклу, операція звелась би до звичайного віднімання по чергово.

У виводі ми бачимо 1 рядок після, виводу це ввід користувача, 4 цифри що позначатимуть, a b c d, що позначенні в завданні. Наступний рядок, це вивід використовуючи побітові зсуви, і останній це використовуючи стандартні арифметичні операції.

**Завдання 2.**

Створити дві консольні програми. Перша програма - введення інформації та шифрування за вказаним алгоритмом у варіанті. Шифрування здійснювати з використанням побітових операцій. Результат записується у бінарний файл. Початкова інформація може бути у текстовому файлі. Друга програма – читає із файлу зашифровану інформацію, розшифровує та виводить у вікно консолі та інший файл.

6. Задано текст до 128 символів. Доповнити пробілами до 128 символів. Шифрувати тексти таким чином, щоб кожний символ тексту записувався у два байти. Два байти мають таку структуру:

0 біт – біт парності (1 біт)

у бітах 1-8 ASCII - код букви (8 біт),

у бітах 9-15 позиція букви у рядку (7 біти)

void crypt() {

for (int i = 0; i < 128; i++) {

cout << a[i] << " ";

char b = a[i];

int counts = 0;

for (int j = 0; j < 16; j++) {

if (b & (1 << j)) {

counts++;

}

if (i & (1 << j)) {

counts++;

}

}

if (counts % 2 == 0) {

data\_is[i] = 0;

} else {

data\_is[i] = 1;

}

data\_is[i] <<= 8;

data\_is[i] |= b;

data\_is[i] <<= 7;

data\_is[i] |= i;

}

}

void decrypt() {

cout << "decrypt" << endl;

for (int i = 0; i < 128; i++) {

int sets = 0;

cout << format("{:0>16b}", data\_is[i]) << endl;

for (int j = 0; j < 15; j++) {

if (data\_is[i] & (1 << j)) {

sets++;

}

}

if ((sets & 1) == (data\_is[i] >> 15)) {

cout << "integirity passed";

} else {

cout << "integirity failed";

continue;

}

int index = (data\_is[i] & 127);

char b = (data\_is[i] >> 7) & 255;

output[index] = b;

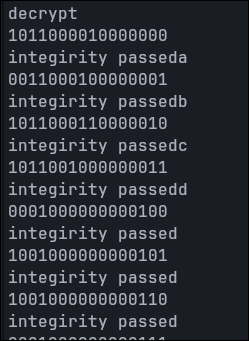
cout << output[index];

cout << endl;

}

}





Так як операція шифрування відбувається побітово, тоді можна розкреслити собі 2 байти для зручного розуміння. В шифрування я проходжусь по всьому списку літер, які я зчитую з файлу або клавіатури. Спочатку проходжусь по списку p 16 бітів щоб підрахувати кількість бітів в символі та номері рядка. Записую це значення(тільки перший біт), на перше місце в рядку. Зсуваю рядок на 8 символів вліво, та записую 8 бітів символа. Зсуваю на ще 7 вліво, та записую номер в рядку. Результат записую в список.

Дешифрую за схожим чином тільки в окремому порядку. Зчитую символ з списку, який зчитав з файлу відповідно. Рахую кількість істинних бітів, пропускаючи перший біт зліва. Та роблю перевірку чи парне це число бітів, якщо крайній правий біт сходиться з крайнім лвим бітом символа, тоді інформація істинна і я можу її зчитати. В індекс я записую символ що помножений на 127, на стільки ж одиничок скільки в мене максимальний номер рядка, щоб знищити все що виходить за ці межі, та отримати чистий індекс. Роблю зсув вправо, та множу на 255 (8 одиничок), і таким же чином позбавляюсь зайвої інформації зліва від символа.

У виводі можна побачити, не суть важливу інформацію в шифруванні, але більш детально можна побачити процесс в дешифрації. Бачимо кожну комірку, та те чи пройшли вони перевірку.

**Завдання 3.**

Реалізувати завдання 2 з використанням структур з бітовими полями та об’єднаннями

typedef struct cryptstruct {

union return\_value {

struct {

unsigned int num : 7;

unsigned int ch : 8;

unsigned int set\_of\_bits : 1;

};

unsigned int return\_value;

} return\_value;

} cryptstruct;

void crypt2() {

for (int i = 0; i < 128; i++) {

char b = a[i];

int counts = 0;

for (int j = 0; j < 16; j++) {

if (b & (1 << j)) {

counts = !(counts);

}

if (i & (1 << j)) {

counts = !(counts);

}

}

cryptstruct c;

c.return\_value.set\_of\_bits = counts;

c.return\_value.ch = b;

c.return\_value.num = i;

data\_is[i] = c.return\_value.return\_value;

cout << format("{:0>16b}", data\_is[i]) << endl;

}

}

void decrypt2() {

cout << "decrypt" << endl;

for (int i = 0; i < 128; i++) {

int sets = 0;

cout << format("{:0>16b}", data\_is[i]) << endl;

cryptstruct c;

c.return\_value.return\_value = data\_is[i];

for (int j = 0; j < 15; j++) {

if (c.return\_value.return\_value & (1 << j)) {

sets++;

}

}

if ((sets & 1) == (c.return\_value.return\_value >> 15)) {

cout << "integirity passed";

} else {

cout << "integirity failed";

continue;

}

int index = (c.return\_value.num);

char b = (c.return\_value.ch);

output[index] = b;

cout << output[index];

cout << endl;

}

}

}

Створив стуктуру, яка має всередині змінну типу юніон, всередині юніон, є стуктура в якій я відповідно вказую, біт парності, 8 бітів символа, та 7 номера рядка. Відповідно вони всі в середині обʼєднання, в якому є ще одна окрема змінна, яка і тримає всі ці змінні в свої памʼятті.

Шифр відбувається відповідно, простим записом в структуру що всередині обʼєднання. Та коли все готово забираю згадану окрему змінну та записую всередину списку виводу.

Відповідно до цього, дешифрація відбувається за допомогою простого запису змінної з вводу, в окерум змінну в обʼєднанні. Та подальша перевірка, беручи данні з структури з обʼєднання.

**4 Завдання.**

Зашифрувати рядок символів циклічним зсувом вліво його бітів на вказану величину.

char shif\_left(char a, int i) { return (a << i) | (a >> (8 - i)); }

char shif\_right(char a, int i) {

unsigned char b = a;

b = b >> i;

b |= a << (8 - i);

return b;

}

void console\_to\_array(char \*a, int size) {

cout << "array" << endl;

for (int i = 0; i < size; i++) {

cin >> a[i];

}

}

void shuffle(char \*a, int flag, int size) {

int shift = 0;

cout << "shift" << endl;

cin >> shift;

for (int i = 0; i < size; i++) {

if (flag == 0) {

cout << shif\_left(a[i], shift);

a[i] = shif\_left(a[i], shift);

} else {

cout << shif\_right(a[i], shift);

a[i] = shif\_right(a[i], shift);

cout << bitset<8>(a[i]) << endl;

}

}

}

Просто беру символ з списку, зсуваю його на відповідну кількість бітів вліво. Стерту частину записую справа відповідно.

**Висновок**

Дізнався як працюють побітові операції, та структури з обʼєднаннями. Попрацював з модулями.