Для демонстрации данного алгоритма предоставлен следующий код.

В данном коде присутствуют новые функции, расширяющие функционал предыдущих лабораторных, для передачи через COM-порт. В этих функциях используется пользовательский класс Heming для реализации алгоритма Хемминга.

Новые функции для передачи закодированного сообщения в классе COM.

void COM::WriteHeming(const uint8\_t\* data)

{

int i;

for (i = 0; \*(data + i) != '\0'; ++i)

{

char temp = \*(data + i);

Heming heming(8, temp);

uint16\_t toSend = heming.code();

Write(&toSend, 2);

}

Heming heming(8, '\0');

uint16\_t toSend = heming.code();

Write(&toSend, 2);

}

void COM::ReadHeming(uint8\_t\* data) {

int i;

uint16\_t temp = 'a';

for (i = 0; temp != '\0'; ++i)

{

Read(&temp, 2);

Heming heming(8, temp);

\*(data + i) = heming.decode();

}

\*(data + i) = '\0';

}

Файл Heming.h.

#pragma once

#include <iostream>

#include <bitset>

using namespace std;

class Heming

{

public:

Heming(int bit);

Heming(int bit, char number);

Heming(int bit, uint16\_t number);

void setNumber(char number);

void setNumber(uint16\_t number);

char getNumber();

void flipNumberAt(int);

uint16\_t code();

char decode();

private:

int numberBits;

bitset<16> number;

bitset<16> result;

int realWord;

int allWord;

int controlWord;

void countWords();

void writeNumberToCode();

void generateControlBits();

void generateControlBitsToCheck();

void checkForMistakes();

void readNumberFromCode();

};

Файл Heming.cpp.

#include "Heming.h"

Heming::Heming(int bit)

{

this->numberBits = bit;

countWords();

}

Heming::Heming(int bit, char number)

{

this->numberBits = bit;

countWords();

this->number = number;}

Heming::Heming(int bit, uint16\_t number)

{

this->numberBits = bit;

countWords();

this->number = number;

}

void Heming::setNumber(char number)

{

this->number = number;

}

void Heming::setNumber(uint16\_t number)

{

this->number = number;

}

char Heming::getNumber()

{

return number.to\_ulong();

}

void Heming::flipNumberAt(int i)

{

this->number.flip(i);

}

void Heming::countWords()

{

realWord = numberBits;

{

//подсчёт контрольной части

allWord = 0;

while ((pow(2, allWord) / (allWord + 1)) < (pow(2, realWord)))

{

allWord++;

}

controlWord = allWord - realWord;

}

void Heming::writeNumberToCode()

{

//запись числа в код защищенный

int k = realWord - 1;

for (int i = 1; i <= allWord; ++i)

{

bool isControlSlot = false;

for (int j = 0; j < controlWord; j++)

{

if (i == pow(2, j))

{

isControlSlot = true;

break;

}

}

if (!isControlSlot)

{

result[i] = number[k--];

}

}

}

void Heming::generateControlBits()

{

//формирование контрольных битов

for (int j = 0; j < controlWord; ++j)

{

int controlSlot = pow(2, j);

int controlBitValue = 0;

bitset<8> check;

int numberOfOne = 0;

for (int i = controlSlot; i <= allWord; i++)

{

check = i;

if (check[j] == 1)

{

numberOfOne += (result[i] == 1) ? 1 : 0;

}

}

result[controlSlot] = (numberOfOne % 2 == 0) ? 0 : 1;

}

cout << "Before sending: "<<result << endl;

}

uint16\_t Heming::code()

{

writeNumberToCode();

generateControlBits();

return result.to\_ulong();

}

void Heming::generateControlBitsToCheck()

{

result = number;

//формирование контрольных битов

for (int j = 0; j < controlWord; ++j)

{

int nControlSlot = pow(2, j);

//cout << nControlSlot<<endl;

int nControlBitValue = 0;

bitset<8> check;

int numberOfOne = 0;

for (int i = nControlSlot + 1; i <= allWord; i++)

{

check = i;

if (check[j] == 1)

{

numberOfOne += (result[i] == 1) ? 1 : 0;

}

}

//cout << endl;

result[nControlSlot] = (numberOfOne % 2 == 0) ? 0 : 1;

}

cout << "After sending: "<< result << endl;

}

void Heming::checkForMistakes()

{

//подсчёт несовпадений

int numberOfDiffrentBits = 0;

int positionMistake = 0;

for (int j = 0; j < controlWord; ++j)

{

int nControlSlot = pow(2, j);

//numberOfDiffrentBits += (nResult[nControlSlot] != nNumber[nControlSlot]) ? 1: 0;

if (result[nControlSlot] != number[nControlSlot])

{

numberOfDiffrentBits++;

positionMistake += nControlSlot;

}

}

switch (numberOfDiffrentBits)

{

case 0:

{

cout << "No mistakes" << endl;

break;

}

case 1:

{

cout << "No mistakes in real number(only in control word)" << endl;

break;

}

default:

{

cout << "There is a mistake at " << positionMistake << " position " << endl;

number.flip(positionMistake);

cout << "Corrected number: " << number << endl;

break;

}

}

}

void Heming::readNumberFromCode()

{

result.reset();

int nk = realWord - 1;

for (int i = 1; i <= allWord; ++i)

{

bool isControlSlot = false;

for (int j = 0; j < controlWord; j++)

{

if (i == pow(2, j))

{

isControlSlot = true;

break;

}

}

if (!isControlSlot)

{

result[nk--] = number[i];

}

}

}

char Heming::decode()

{

countWords();

generateControlBitsToCheck();

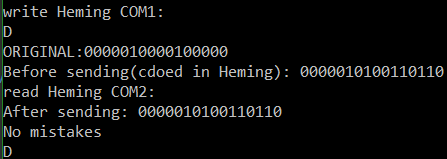
checkForMistakes();

readNumberFromCode();

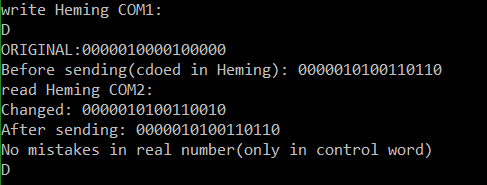
return result.to\_ulong();

}

Пример работы кода без ошибок:



Пример работы кода c ошибкой в контрольном бите (создана вручную после приёма сообщения):



Пример работы кода c ошибкой (создана вручную после приёма сообщения):

