МИНОБРНАУКИ РОССИИ

–––––––——————————–––––––

Санкт-Петербургский государственный  
электротехнический университет «ЛЭТИ»

————————————————————

**Лабораторная работа №1 по теме**

**«Исследование внутреннего представления различных форматов данных»**

Санкт-Петербург

Издательство СПБГЭТУ «ЛЭТИ»

2024

**Цель работы**

Знакомство с внутренним представлением различных типов данных, используемых компьютером при их обработке.

**Задание (вариант 5)**

1. Разработать алгоритм ввода с клавиатуры int число и показать на экране его внутреннее представление в двоичной системе счисления.

2. Написать и отладить программу на языке С, реализующую разработан-ный алгоритм. Программа должна

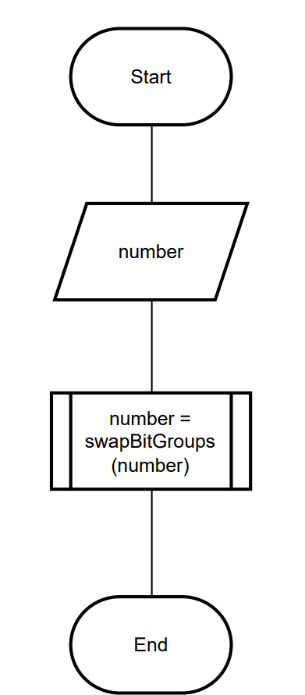
- иметь дружественный интерфейс

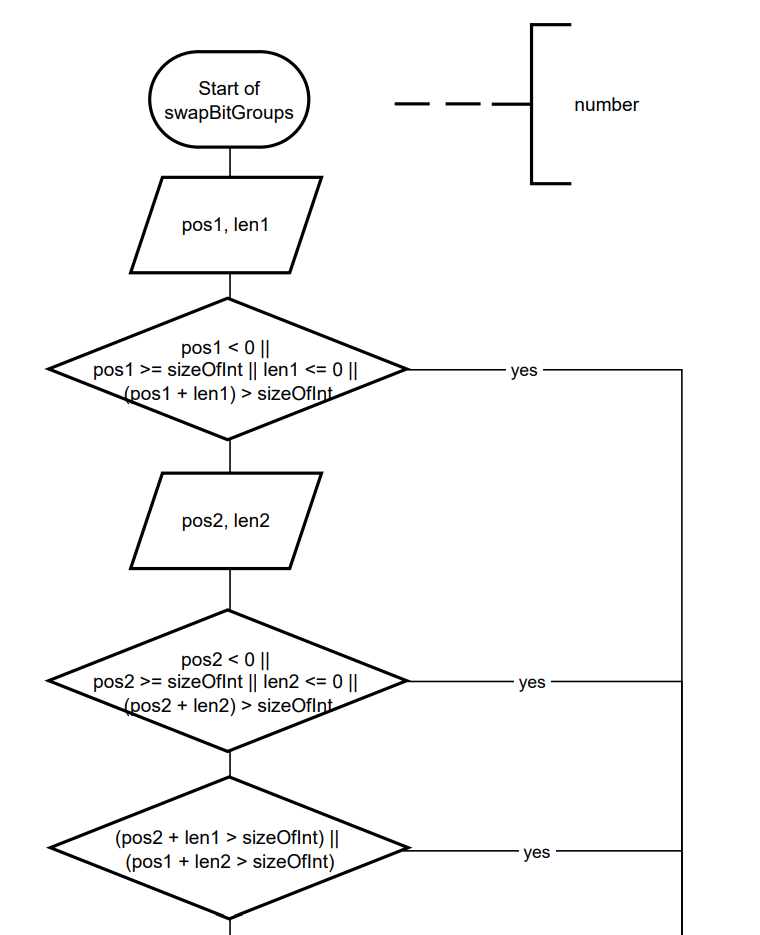
- выводить на экран информативное сообщение при вводе некорректных дан-ных

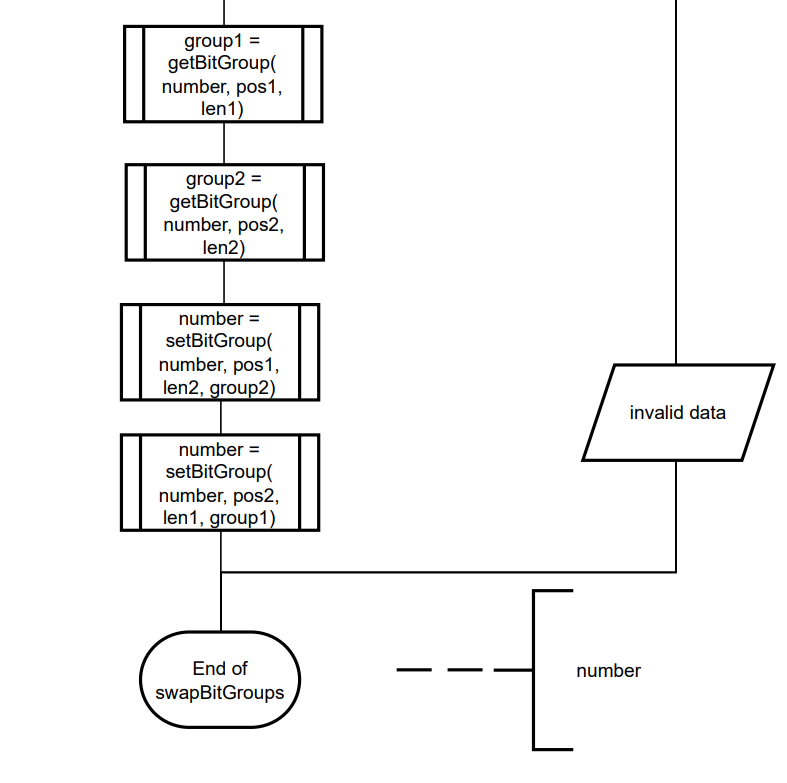
- предложить повторный ввод пока не будут введены корректные данные

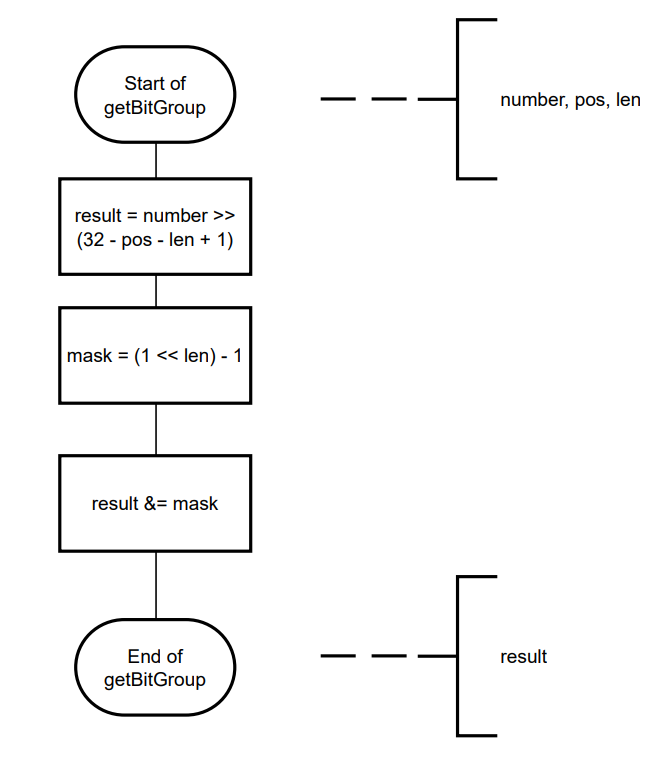
3. Поменять местами заданные пользователем группы рядом стоящих бит, номера старших разрядов этих групп и количество бит в группе, вводится с клавиатуры.

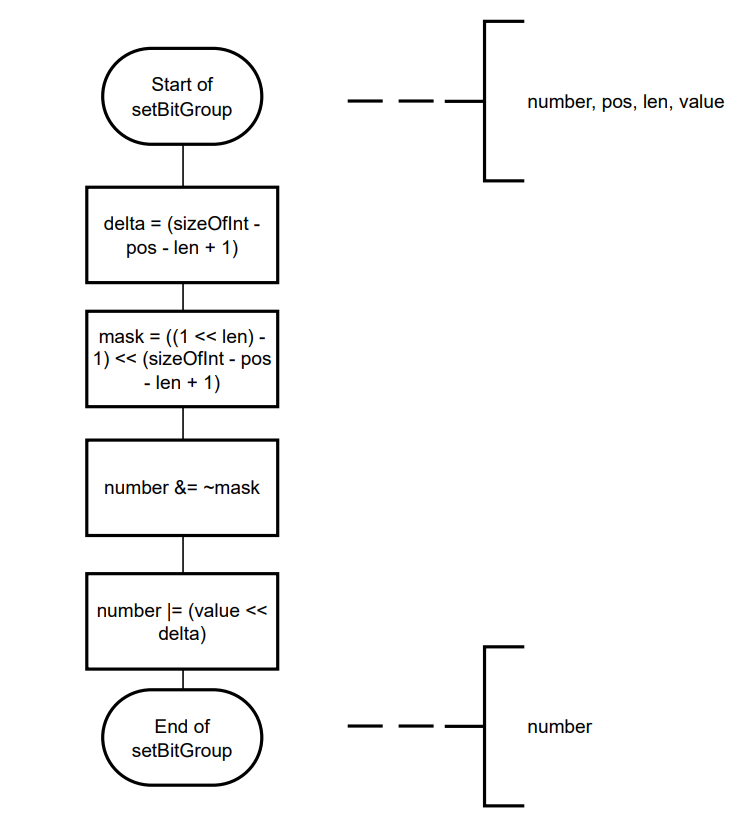
**Блок-схема алгоритма**

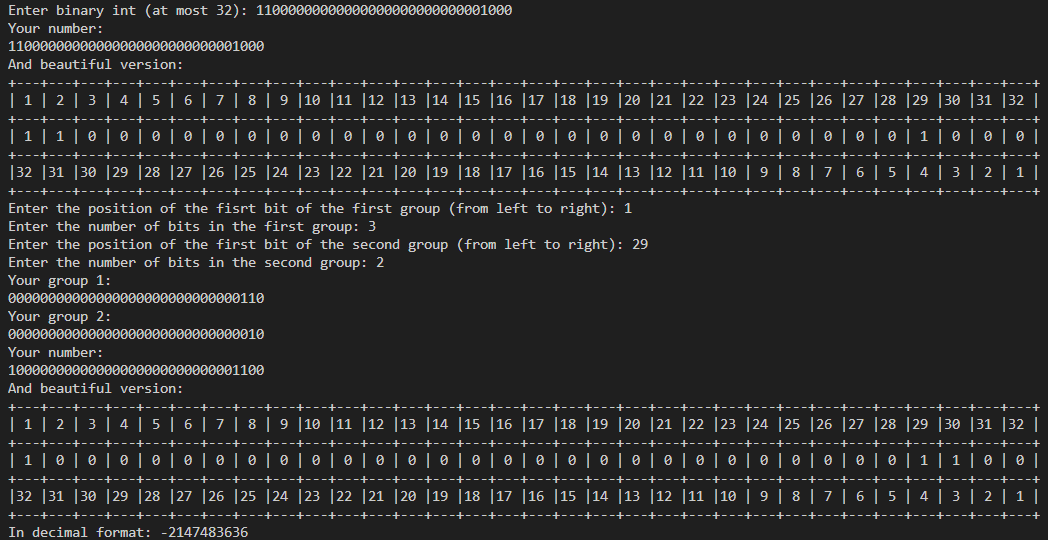


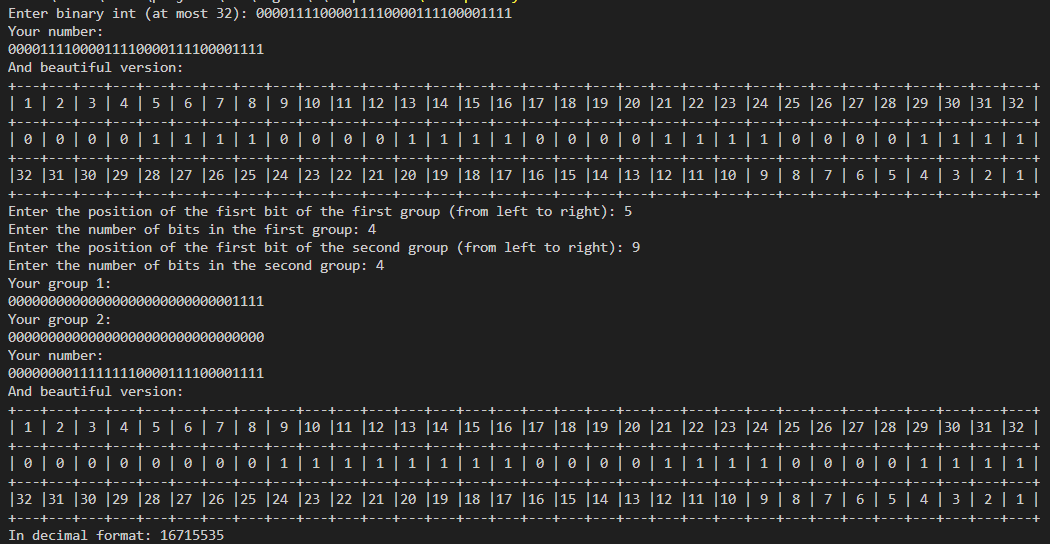


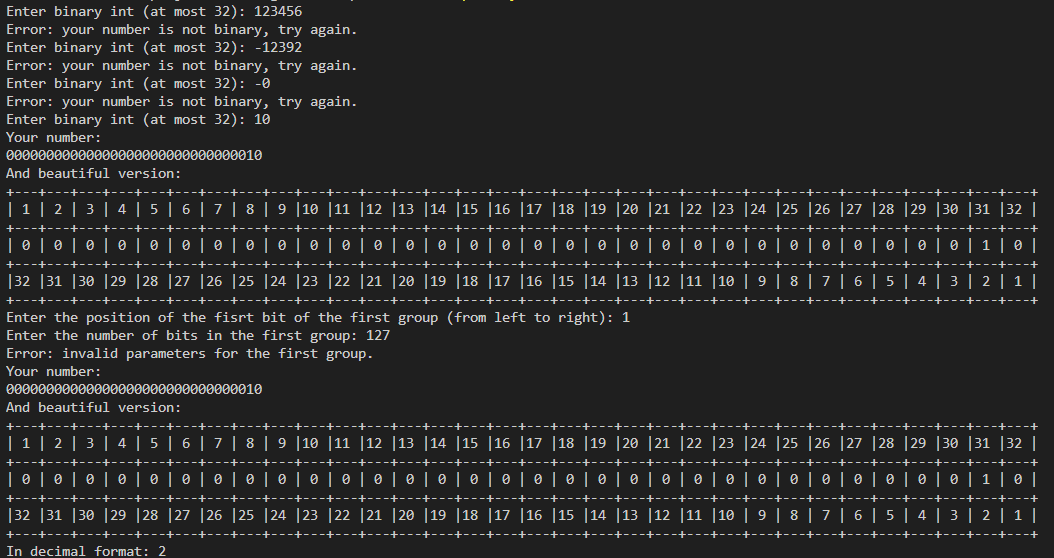


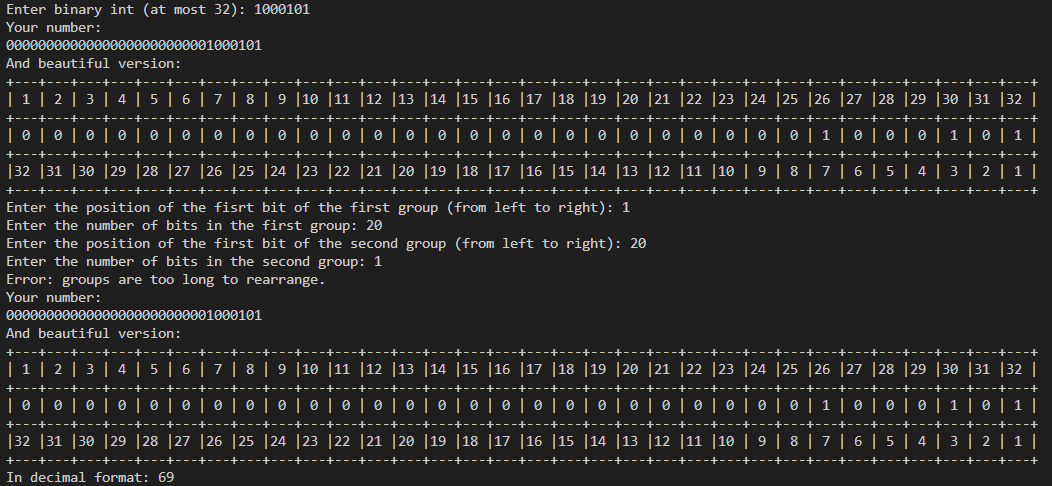




**Тестовые примеры выполнения программы**

****





**Текст программы**

#include <stdio.h>

#include <string.h>

const int sizeOfInt = sizeof(int) \* 8;

/\* io \*/

void printBinaryInteger(int n) {

int size = sizeof(int) \* 8;

for (int i = size - 1; i >= 0; i--) {

int bit = (n >> i) & 1;

printf("%d", bit);

}

printf("\n");

}

void printLine() {

printf("+");

for (int i = 1; i <= sizeOfInt; i++) {

printf("---");

if (i != sizeOfInt) {

printf("+");

}

}

printf("+\n");

}

void beautifulPrint(int n) {

printLine();

printf("|");

for (int i = 1; i <= sizeOfInt; i++) {

printf("%2d |", i);

}

printf("\n");

printLine();

printf("|");

for (int i = sizeOfInt - 1; i >= 0; i--) {

int bit = (n >> i) & 1;

printf("%2d |", bit);

}

printf("\n");

printLine();

printf("|");

for (int i = sizeOfInt; i >= 1; i--) {

printf("%2d |", i);

}

printf("\n");

printLine();

}

int inputBinaryAndConvert() {

char binaryStr[35] = {0};

int isValid;

int result;

do {

isValid = 1;

result = 0;

printf("Enter binary int (at most %d): ", sizeOfInt);

if (fgets(binaryStr, sizeof(binaryStr), stdin) == NULL) {

printf("Error reading input, try again.\n");

isValid = 0;

continue;

}

int len = strlen(binaryStr);

if (binaryStr[len - 1] == '\n') {

binaryStr[len - 1] = '\0';

len--;

}

if (len > sizeOfInt) {

printf("Error: input too long, try again.\n");

isValid = 0;

continue;

}

for (int i = 0; binaryStr[i] != '\0'; i++) {

if (binaryStr[i] != '0' && binaryStr[i] != '1') {

isValid = 0;

printf("Error: your number is not binary, try again.\n");

break;

}

result = result \* 2 + (binaryStr[i] - '0');

}

} while (!isValid);

return result;

}

/\* logic \*/

int getBitGroup(int number, int pos, int len) {

int result = number >> (sizeOfInt - pos - len + 1);

int mask = (1 << len) - 1;

result &= mask;

return result;

}

int setBitGroup(int number, int pos, int len, int value) {

int delta = (sizeOfInt - pos - len + 1);

int mask = ((1 << len) - 1) << delta;

number &= ~mask;

number |= (value << delta);

return number;

}

int swapBitGroups(int number) {

int pos1, len1, pos2, len2;

printf("Enter the position of the fisrt bit of the first group (from left to right): ");

scanf("%d", &pos1);

printf("Enter the number of bits in the first group: ");

scanf("%d", &len1);

if (pos1 < 0 || pos1 >= sizeOfInt || len1 <= 0 || (pos1 + len1) > sizeOfInt) {

printf("Error: invalid parameters for the first group.\n");

return number;

}

printf("Enter the position of the first bit of the second group (from left to right): ");

scanf("%d", &pos2);

printf("Enter the number of bits in the second group: ");

scanf("%d", &len2);

if (pos2 < 0 || pos2 >= sizeOfInt || len2 <= 0 || (pos2 + len2) > sizeOfInt) {

printf("Error: invalid parameters for the second group.\n");

return number;

}

if ((pos2 + len1 > sizeOfInt) || (pos1 + len2 > sizeOfInt)) {

printf("Error: groups are too long to rearrange.\n");

return number;

}

int group1 = getBitGroup(number, pos1, len1);

printf("Your group 1:\n");

printBinaryInteger(group1);

int group2 = getBitGroup(number, pos2, len2);

printf("Your group 2:\n");

printBinaryInteger(group2);

number = setBitGroup(number, pos1, len2, group2);

number = setBitGroup(number, pos2, len1, group1);

return number;

}

int main() {

int number = inputBinaryAndConvert();

printf("Your number:\n");

printBinaryInteger(number);

printf("And beautiful version:\n");

beautifulPrint(number);

number = swapBitGroups(number);

printf("Your number:\n");

printBinaryInteger(number);

printf("And beautiful version:\n");

beautifulPrint(number);

printf("In decimal format: %d", number);

return 0;

}

**Выводы**

В ходе выполнения работы мы изучили, как компьютер хранит и обрабатывает различные типы данных, а также подробно рассмотрели двоичное представление целых чисел и операции сдвига битов. Мы узнали и применили следующие ключевые моменты:

Мы изучили, как целые числа хранятся в памяти компьютера в виде двоичных кодов. Для этого было реализовано отображение чисел в двоичном виде, что позволило визуализировать, как каждый бит числа используется для представления данных.

Рассмотрели особенности хранения как положительных, так и отрицательных чисел с использованием дополнительного кода. Этот метод позволяет эффективно представлять отрицательные числа и выполнять арифметические операции.

Мы рассмотрели три основных метода представления чисел со знаком: прямой код, обратный код и дополнительный код. Узнали, что прямой и обратный коды имеют недостатки (например, два представления для нуля), и что дополнительный код широко используется благодаря удобству выполнения арифметических операций.

Мы научились работать с побитовыми сдвигами чисел, как влево, так и вправо. Рассмотрели разницу между арифметическим и логическим сдвигом, особенно в контексте обработки отрицательных чисел, где важно сохранять или изменять знаковый бит.

Применили эти знания для обработки групп битов, извлекая их из числа и меняя их местами.

Также в процессе работы мы коснулись темы представления чисел с плавающей запятой, где изучили структуру числа: знак, экспонента и мантисса. Это дало понимание, как числа с плавающей запятой хранятся в двоичной системе.

Был разработан и реализован алгоритм, который позволяет пользователю задавать две группы битов, а затем менять их местами. Для этого мы научились извлекать группы битов, обрабатывать их и устанавливать на новые позиции с помощью побитовых операций.

Таким образом, мы научились эффективно работать с двоичными представлениями данных, узнали, как компьютер хранит целые числа и числа с плавающей запятой, и реализовали операции сдвига и перестановки битовых групп с учётом внутренней структуры данных.