МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра информационных систем

ОТЧЕТ

по лабораторной работе №1

по дисциплине «ОЭВМ»

Тема: ИССЛЕДОВАНИЕ ВНУТРЕННЕГО ПРЕДСТАВЛЕНИЯ РАЗЛИЧНЫХ ФОРМАТОВ ДАННЫХ

Студенты гр. 4373 _____ Шепелев Д.Н.
Дядюра Ю.С.

Преподаватель _____ Кочетков А.В.

Санкт-Петербург

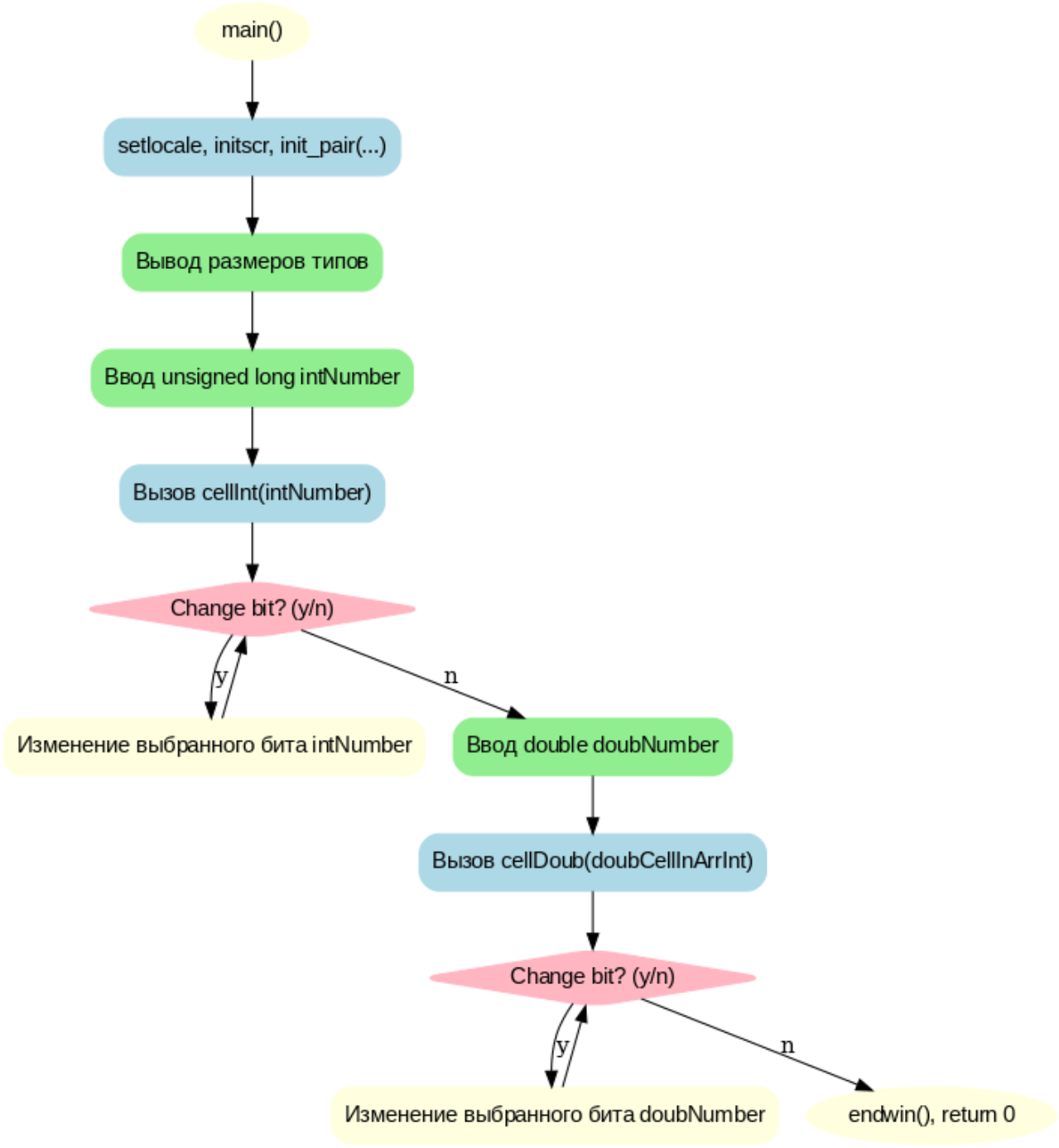
2024

Задание на лабораторную работу

1. В зависимости от номера варианта задания разработать алгоритм ввода с клавиатуры требуемых типов данных и показать на экране их внутреннее представление в двоичной системе счисления.

2. Написать и отладить программу на языке C++, реализующую разработанный алгоритм.

3. В соответствии с заданием дополнить разработанный ранее алгоритм блоками для выполнения преобразования двоичного полученного кода исходного типа данных и последующего вывода преобразованного кода в двоичной системе счисления и в формате исходного данного. Задание: установить в заданное пользователем состояние определённое количество бит, номера которых, как и всё остальное, вводится с клавиатуры.



Блок-схемы алгоритмов

Рис 1 Блок-схема функции main

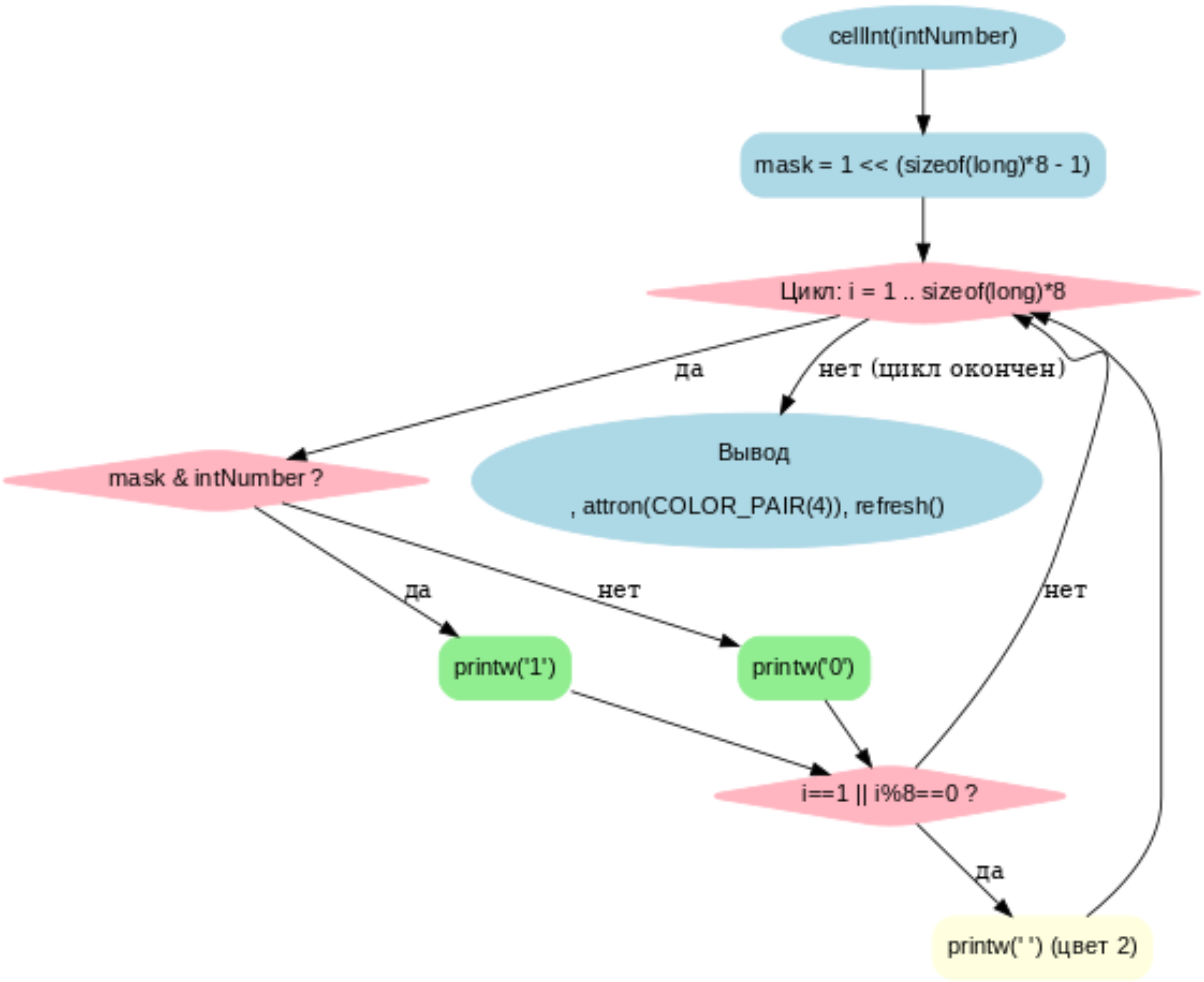


Рис 2 Блок-схема функции cellint

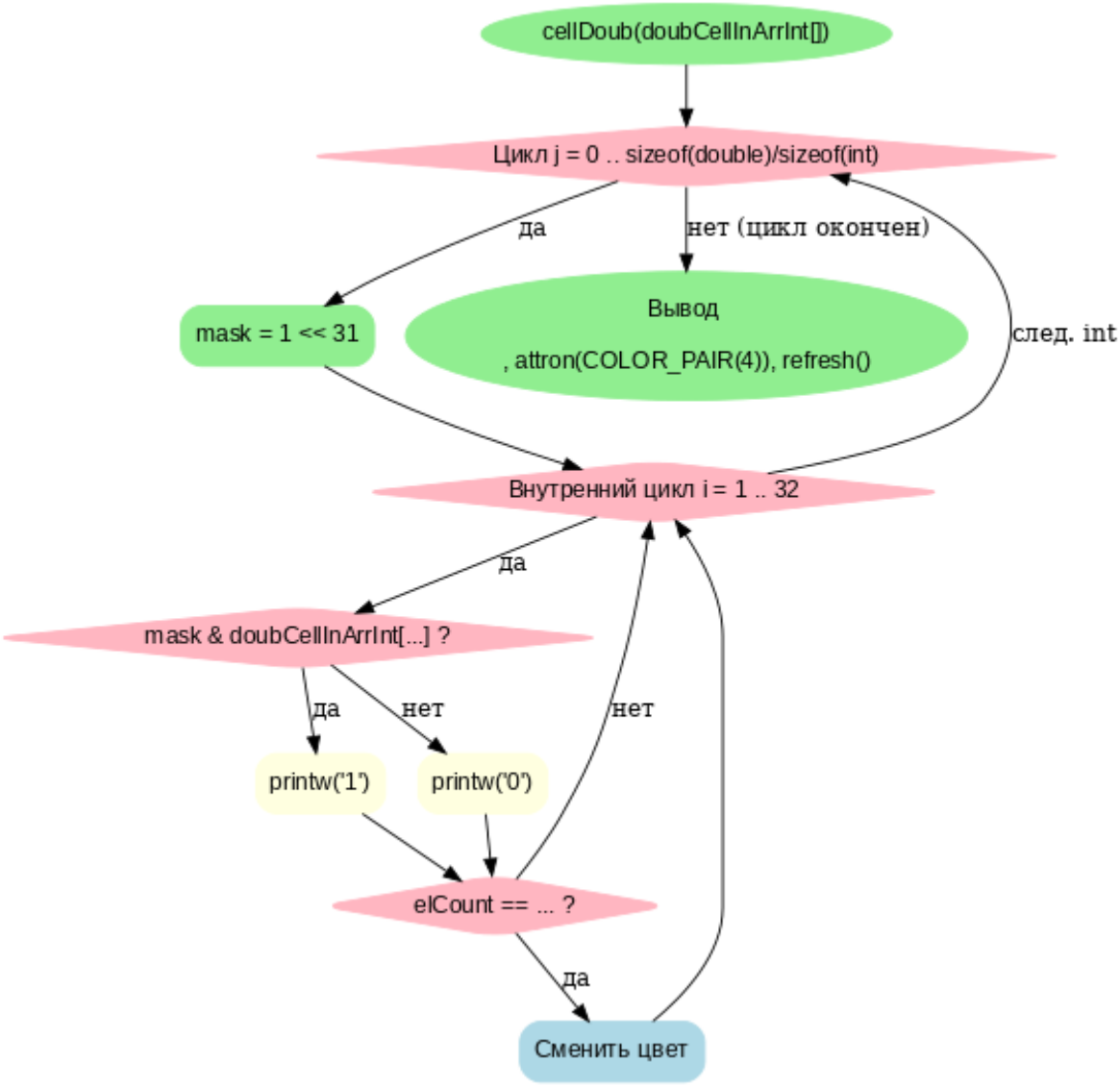


Рис 3 Блок-схема функции и cellDoub

Код программы:

```
#include <iostream>
#include <ncurses.h>
```

```
using namespace std;

void cellInt(unsigned long intNumber) {
unsigned long mask = 1;
mask <<= sizeof(long) * 8 - 1;
attron(COLOR_PAIR(1));
for (long i = 1; i < sizeof(long) * 8 + 1; ++i, mask >>= 1) {
mask & intNumber ? printw("1") : printw("0");
if (i == 1 || i % 8 == 0) {
attron(COLOR_PAIR(2));
printw(" ");
}
}
printw( "\n\n" );
attron(COLOR_PAIR(4));
refresh();
}

void cellDoub(int doubCellInArrInt[]) {

unsigned int mask, elCount = 0;
attron(COLOR_PAIR(1));
for (int j = 0; j < sizeof(double) / sizeof(int); ++j) {
mask = 1;
mask <<= 31;
for (int i = 1; i < sizeof(int) * 8 + 1; ++i, mask >>= 1) {
mask& doubCellInArrInt[sizeof(double) / sizeof(int) - 1 - j] ? printw("1") : printw("0");
elCount++;
if (elCount == (sizeof(double) / 8) * 12) {
attron(COLOR_PAIR(3));
printw(" ");
}
else if (elCount == 1) {
attron(COLOR_PAIR(2));
printw(" ");
}
}
}
printw( "\n\n");
attron(COLOR_PAIR(4));
refresh();
}

int main()
{
setlocale(LC_ALL, "Russian");
initscr();
if (has_colors() == FALSE){
endwin();
cout << "Your terminal does not support color\n)";
```

```
return 1;
}
start_color();
init_pair(1,COLOR_RED, COLOR_BLACK);
init_pair(2,COLOR_GREEN, COLOR_BLACK);
init_pair(3,COLOR_BLUE, COLOR_BLACK);
init_pair(4,COLOR_WHITE, COLOR_BLACK);
int bitID, bitValue;
unsigned long intNumber;
unsigned long maskLong;
unsigned int maskBitID[2] = {0,0};
char input;
union {
float floatNumber;
int floatCellInInt;
};
union {
double doubNumber;
int doubCellInArrInt[sizeof(double) / sizeof(int)];
};
setlocale(0, "");

printw("int: %ld bytes\n", sizeof(int));
printw("short int: %ld bytes\n" , sizeof(short int));
printw("long int: %ld bytes\n", sizeof(long int));
printw("float: %ld bytes\n" , sizeof(float));
printw("double: %ld bytes\n" , sizeof(double));
printw("long double: %ld bytes\n" , sizeof(long double));
printw("char : %ld bytes\n", sizeof(char));
printw("bool: %ld bytes\n", sizeof(bool));

printw("\nEnter variable unsigned long: ");
refresh();
scanw("%ld",&intNumber);
printw("\n");
cellInt(intNumber);

do{
printw( "Change bit? (y/n): ");
refresh();
scanw("%c",&input);
if (input != 'n'){
printw( "\nBit place (right to left): ");
refresh();
scanw("%i",&bitID);
printw( "\nBit value: ");
refresh();
scanw("%i",&bitValue);
maskLong = 1;
maskLong <<= bitID;
```

```
if (bitValue == 1) {
if ((intNumber ^ maskLong) > intNumber) {
intNumber ^= maskLong;
}
}
else {
if ((intNumber ^ maskLong) < intNumber) {
intNumber ^= maskLong;
}
}
printw("\n");
cellInt(intNumber);
printw("Result number: %ld \n\n", intNumber);
refresh();
}
} while (input != 'n');

printw( "\nEnter variable double: ");
refresh();
scanw("%lf",&doubNumber);
printw("\n");
cellDoub(doubCellInArrInt);
do {
printw("Change bit? (y/n): ");
refresh();
scanw("%c",&input);
if (input != 'n') {
printw( "\nnBit place (right to left): ");
refresh();
scanw("%i",&bitID);
printw( "\nBit value: ");
refresh();
scanw("%i",&bitValue);
if (bitID < 32) {
maskBitID[0] = 0;
maskBitID[1] = 1;
maskBitID[1] <<= bitID;
}
else {
maskBitID[0] = 1;
maskBitID[1] = 0;
maskBitID[0] <<= bitID - 32;
}
if (bitValue == 1) {
if (((doubCellInArrInt[1] ^ maskBitID[0]) + (doubCellInArrInt[0] ^ maskBitID[1])) > doubCellInArrInt[0]+
doubCellInArrInt[1]) {
doubCellInArrInt[1] ^= maskBitID[0];
doubCellInArrInt[0] ^= maskBitID[1];
}
}
```

```
else {
if (((doubCellInArrInt[1] ^ maskBitID[0]) + (doubCellInArrInt[0] ^ maskBitID[1])) < doubCellInArrInt[0] +
doubCellInArrInt[1]) {
doubCellInArrInt[1] ^= maskBitID[0];
doubCellInArrInt[0] ^= maskBitID[1];
}
}
printw("\n");
cellDoub(doubCellInArrInt);
printw("\nResult number: %f \n\n", doubNumber);
refresh();
}
} while (input != 'n');

endwin();
return 0;
}
```

Примеры запуска:

```
int: 4 bytes
short int: 2 bytes
long int: 8 bytes
float: 4 bytes
double: 8 bytes
long double: 16 bytes
char : 1 bytes
bool: 1 bytes

Enter variable unsigned long: 7

0 0000000 00000000 00000000 00000000 00000000 00000000 00000000 00000111

Change bit? (y/n): n

Enter variable double: 0

0 00000000000 0000000000000000000000000000000000000000000000000000

Change bit? (y/n): y

nBit place (right to left): 0

Bit value: 1

0 00000000000 0000000000000000000000000000000000000000000000000001


Result number: 0.000000

Change bit? (y/n):
```