

1. **Цель работы**

Ознакомление с поразрядными операциями и сдвигами.

1. **Постановка задачи**

Вариант 1: В длинном целом числе N поменять местами нулевой разряд с первым разрядом, второй разряд с третьим разрядом и т.д.

1. **Формализация задачи**

N является длинным целым числом. Число N вводится пользователем в шестнадцатеричном виде. Для наглядности введённое число отображается на экране в двоичном виде. Задаются маски чётных и нечётных битов. Число N, считанное по маске нечётных разрядов, смещается на бит влево, а число N, считанное по маске чётных разрядов смещается на бит вправо. Полученные промежуточные числа логически складываются и заносятся в число N как итоговый результат, который для наглядности вначале выводится в двоичном, а затем в шестнадцатеричном виде. По завершении программы, пользователю предлагается повторить вычисления либо завершить работу программы.

1. **Набор тестовых примеров**

|  |  |
| --- | --- |
| Входные значения | Вывод программы |
| a | 5 |
| f63c | f93c |
| 3 | 3 |

1. **Схема алгоритма**

Схема алгоритма программы представлена на рисунке 1.

Начало

A

В

endFlag = 1; N;

mask = 0xaaaaaaaa;

mask2 = 0x55555555;

tmp, tmp2;

Вывод N

Ввод N

1 - Count again.

2 - Exit.

1

2

tmp = N & mask;

tmp = tmp >> 1;

tmp2 = N & mask2;

tmp2 = tmp2 << 1;

Конец

A

В

*Рисунок 1 – Схема алгоритма программы.*

1. **Листинг программы**

#if defined(\_WIN32) || defined(\_\_WIN32\_\_) || defined(WIN32)

#define \_CRT\_SECURE\_NO\_WARNINGS

#pragma warning(disable:4996)

#pragma warning(disable:6031)

#endif

#include <stdio.h>

#include <stdlib.h> //contains functions for memory allocation, control of the program execution process, type conversion, etc.

int main() {

int endFlag = 1; //a flag that stops the application

//a global cycle that prevents the program from ending without the users desire

do {

long int N; //the number to be operated on

unsigned long int mask = 0xaaaaaaaa; //mask of all odd bits

unsigned long int mask2 = 0x55555555; //mask of all even bits

int tmp, tmp2;

//prompt for user input

printf("Please input N in hex format: ");

scanf("%lx", &N);

//output in binary for clarity

printf("\nYour number in binary : ");

for (int i = CHAR\_BIT \* sizeof(long) - 1; i >= 0 ; i--)

printf("%d", (N & (1 << i)) ? 1 : 0);

tmp = N & mask; //extracting odd bits

tmp = tmp >> 1; //offset odd to the left

tmp2 = N & mask2; //extracting even bits

tmp2 = tmp2 << 1; //offset even to the right

N = tmp | tmp2; //combining the results of offsets

printf("\nYour number in binary : ");

for (int i = CHAR\_BIT \* sizeof(long) - 1; i >= 0; i--)

printf("%d", (N & (1 << i)) ? 1 : 0);

printf("\n\nNew N is: %lx", N);

//suggesting to user to run program again or exit.

printf("\n\nPlease choose from the following:\n\t1 - Count again.\n\t2 - Exit.\n");

scanf("%d", &endFlag);

switch (endFlag) {

case 1: endFlag = 1; break;

case 2: endFlag = 0; break;

default: printf("Incorrect answer. The program will be closed."); endFlag = 0; break;

}

} while (endFlag); //if flag is 0, then exiting out of cycle

return 0;

}

1. **Выводы**

Поразрядные операции допустимы только над целыми типами данных.

Приведите таблицы истинности для известных в языке Си поразрядных логических операций.

Дополнение

|  |  |
| --- | --- |
| x | ~ |
| 0 | 1 |
| 1 | 0 |

Поразрядное логическое умножение

|  |  |  |
| --- | --- | --- |
| x | y | & |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Поразрядное логическое сложение

|  |  |  |
| --- | --- | --- |
| x | y | | |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Исключающее ИЛИ

|  |  |  |
| --- | --- | --- |
| x | y | ^ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

В общем случае нельзя сказать, что сдвиг вправо на один разряд эквивалентен делению на 2, потому что для нечётных отрицательных чисел такая эквивалентность работает некорректно.

В общем случае можно сказать, что сдвиг влево на один разряд эквивалентен умножению на 2, потому что такая эквивалентность для всех чисел, в частности для беззнаковых и знаковых, работает одинаково

В языке Си существует 4 поразрядных унарных операций:

~ Дополнение -- инвертирует каждый разряд независимо

& Поразрядное логическое умножение

| Поразрядное логическое сложение

^ Исключающее ИЛИ

Также существуют 2 сдвиговые операции:

>> Сдвиг влево

<< Сдвиг вправо