|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Поразрядные операции и их применение»** | |
|  | |
| Выполнил студент группы ИКБО-02-21 | Семянников Н.С. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2022

# **Цель работы**

Получение навыков применения поразрядных операций в алгоритмах.

# **Постановка задачи**

1. Разработать программу, которая продемонстрирует выполнение упражнений варианта. Результаты выполнения упражнения выводить на монитор.

Требования к упражнениям:

1. Определить переменную целого типа, присвоить ей значение, используя константу в шестнадцатеричной системе счисления. Разработать функцию, которое установит заданные в задании биты исходного значения переменной в значение 1, используя соответствующую маску и поразрядную операцию.
2. Разработать функцию, которая обнуляет заданные в задании биты исходного значения целочисленной переменной, введенной пользователем, используя соответствующую маску и поразрядную операцию.
3. Разработать функцию, которая умножает значение целочисленной переменной, введенной пользователем, на множитель, используя соответствующую поразрядную операцию.
4. Разработать функцию, которая делит значение целочисленной переменной, введенной пользователем, на делитель, используя соответствующую поразрядную операцию.
5. Разработать функцию, реализующую задание, в которой используются только поразрядные операции. В выражении используется маска – переменная. Маска инициализируется единицей в младшем разряде (маска 1) или единицей в старшем разряде (маска 2). Изменяемое число и n вводится с клавиатуры.
6. Провести тестирование программы на небольших объемах данных, введенных вручную. Разработанные тесты должны покрывать все случаи входных данных (средний, лучший, худший). Результаты тестирования свести в сводные таблицы.
7. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Вариант №5. Условие задания:

|  |  |
| --- | --- |
| Упражнение 1 | 17-ый, 15-ый, 1-ый |
| Упражнение 2 | С 5-ого бита три справа |
| Упражнение 3 | 64 |
| Упражнение 4 | 64 |
| Упражнение 5 | Обнулить n-ый бит в 1, используя маску пункта 2 |

# **Решение**

Поразрядные операции - операции над цепочками битов. Как правило, в этот класс включаются логические побитовые операции и битовые сдвиги. Применяются в языках программирования и цифровой технике, изучаются в дискретной математике.

Битовые операции лежат в основе обработки цифровых сигналов: посредством их из одного или нескольких сигналов на входе получается новый сигнал, который в свою очередь может быть подан на вход одной или нескольким таким операциям. Именно битовые операции в сочетании с запоминающими элементами (например, триггерами) реализуют всё богатство возможностей современной цифровой техники.

Побитовое отрицание (~) – унарная операция. Преобразовывает поданный бит в противоположный, то есть 1 в 0, а 0 в 1.

Таблица 1. Таблица истинности операции Побитовое отрицание

|  |  |  |
| --- | --- | --- |
| **НЕ** | **0** | **1** |
|  | 1 | 0 |

Побитовое И (&) – бинарная операция. В случае если оба бита будут равны 1, то результат будет 1, в противном случае будет 0.

Таблица 2. Таблица истинности операции Побитовое И

|  |  |  |
| --- | --- | --- |
| **И** | **0** | **1** |
| **0** | 0 | 0 |
| **1** | 0 | 1 |

Побитовое ИЛИ (|) – бинарная операция. В случае если хотя бы один бит будет равен 1, то результат будет 1. В противном случае 0.

Таблица 3. Таблица истинности операции Побитовое ИЛИ

|  |  |  |
| --- | --- | --- |
| **ИЛИ** | **0** | **1** |
| **0** | 0 | 1 |
| **1** | 1 | 1 |

Побитовое исключающее ИЛИ (^) – бинарная операция. В случае, если оба бита одинаковые, результат будет 0, в противном случае будет 1.

Таблица 4. Таблица истинности операции Побитовое исключающее ИЛИ

|  |  |  |
| --- | --- | --- |
| **Искл. ИЛИ** | **0** | **1** |
| **0** | 1 | 0 |
| **1** | 0 | 1 |

Битовый сдвиг (<< - умножение, >> - деление) – бинарная операция. Все биты в двоичной системе счисления сдвигаются влево (умножение на 2) или вправо (деление на 2).

Упражнение 1. В статичном числе 17-й, 15-й и 1-й биты сделать 1.

Для реализации задания была разработана маска в 16-чной системе счисления 0x28002. Данная маска и статичное число передаются в разработанный метод logOr. Метод возвращает результат операции, путем сложения числа и маски.

|  |
| --- |
| //Функция сложения с помощью поразрядных операций (упр.1)  int logOr(int x, int mask)  {  return x | mask;  } |

Упражнение 2. В статичном числе 17-й, 15-й и 1-й биты сделать 0.

Для реализации задания была разработана маска в 16-чной системе счисления 0xFFFFFFC7. Данная маска и введенное пользователем число передаются в разработанный метод logAnd. Метод возвращает результат операции, путем умножения числа на маску.

|  |
| --- |
| //Функция умножения с помощью поразрядных операций (упр.2)  int logAnd(int x, int mask)  {  return x & mask;  } |

Упражнение 3. Умножение числа на 64.

Для реализации задания был разработан метод logMult, в который передается введенное пользователем число. Метод возвращает результат операции, путем побитового сдвига числа.

|  |
| --- |
| //Функция поразрядного умножения с помощью побитового умножения(упр.3)  int logMult(int x)  {  return x << 6;  } |

Упражнение 4. Деление числа на 64.

Для реализации задания был разработан метод logDiv, в который передается введенное пользователем число. Метод возвращает результат операции, путем побитового сдвига числа.

|  |
| --- |
| //Функция поразрядного деления с помощью побитового деления(упр.4)  int logDiv(int x)  {  return x >> 6;  } |

Упражнение 5. Установить введенный пользователем бит в 0, используя маску 0x80000000

Для реализации задания был использован метод logAnd, в который передается введенное пользователем число и маска. Предварительно старшая единица сдвигается на определённое количество разрядов вправо, и вся маска инвертируется. Метод возвращает результат операции, путем сложения числа и новой маски.

|  |
| --- |
| //Функция умножения с помощью поразрядных операций (упр.2)  int logAnd(int x, int mask)  {  return x & mask;  } |

После запуска программы, пользователю предлагается выбрать номер упражнения в меню. После выбора, пользователю выводится информация о том, какие переменные ему нужно ввести. В случае правильного ввода всех данных программа выдает результат упражнения. Для повторного выполнения упражнения, программу необходимо закрыть и запустить заново.

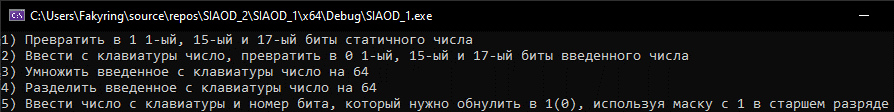


Рисунок - Меню программы

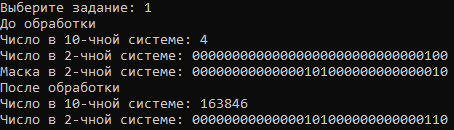


Рисунок – Без ввода

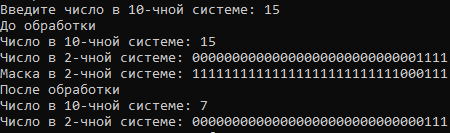


Рисунок – С вводом

# **Тестирование**

Для тестирования программы были выбраны следующие числа в 10-чной системе счисления.

|  |  |  |
| --- | --- | --- |
| Порядковый номер тестирования | Номер упражнения | Число |
| 1 | 1 | 4 |
| 2 | 2 | 60 |
| 3 | 2 | 100 |
| 4 | 3 | 3 |
| 5 | 4 | 128 |
| 6 | 4 | 60 |
| 7 | 5 | 173, 3 |

В первом тесте видно, что число увеличивается на 163842 (рисунок 4)

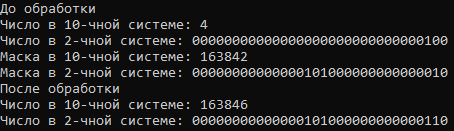


Рисунок - Тест 1

Во втором и третьем тестах видно, что число уменьшается на 56, если в нужных позициях (1, 15, 17) стоит 1 (рисунок 5 и рисунок 6)

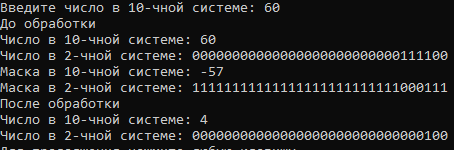


Рисунок 5 – Тест 2

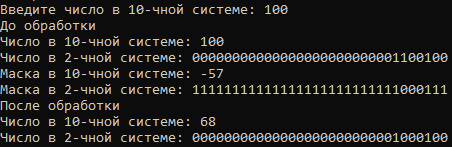


Рисунок - Тест 3

В четвёртом тесте видно, что число умножается на 64 (рисунок 7)

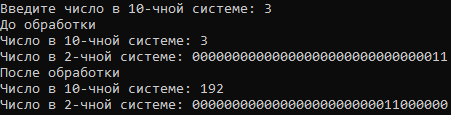


Рисунок - Тест 4

В пятом тесте видно, что число делится на 64 (рисунок 8)

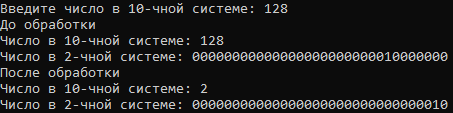


Рисунок - Тест 5

В шестом тесте видно, что если число меньше 64, то результат будет 0 (рисунок 9)

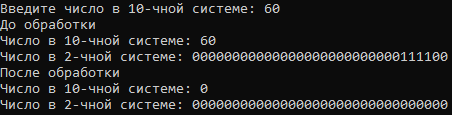


Рисунок - Тест 6

В седьмом тесте видно, что число уменьшается на 8, если выбранный бит – 3 (рисунок 10)

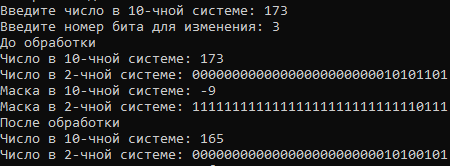


Рисунок - Тест 7

Из результатов выполнения программы видно:

1. Программа работает корректно.

2. Программа обрабатывает все введенные данные

# **Вывод**

В результате выполнения работы я:

1. Освоил алгоритмы работы с поразрядными операциями и их реализацию на языке программирования C++
2. Научился программировать автоматическое тестирование простых программ

# **Исходный код программы**

|  |
| --- |
| #include <iostream>  using namespace std;  void toBin(unsigned int x)  {  int n = sizeof(int) \* 8;  unsigned mask = (1 << (n - 1));  for (int i = 1; i <= n; i++)  {  cout << ((x & mask) >> (n - i));  mask = mask >> 1;  }  }  int logOr(int x, int mask)  {  return x | mask;  }  int logAnd(int x, int mask)  {  return x & mask;  }  int logMult(int x)  {  return x << 6;  }  int logDiv(int x)  {  return x >> 6;  }  int main()  {  setlocale(0, "");  cout << "1) Превратить в 1 1-ый, 15-ый и 17-ый биты статичного числа" << endl;  cout << "2) Ввести с клавиатуры число, превратить в 0 1-ый, 15-ый и 17-ый биты введенного числа" << endl;  cout << "3) Умножить введенное с клавиатуры число на 64" << endl;  cout << "4) Разделить введенное с клавиатуры число на 64" << endl;  cout << "5) Ввести число с клавиатуры и номер бита, который нужно обнулить в 1(0), используя маску с 1 в старшем разряде" << endl;  cout << "Выберите задание: ";  int a;  int n, mask, b;  cin >> a;  switch (a)  {  case 1: {  n = 4;  mask = 0x28002;  cout << "До обработки" << endl;  cout << "Число в 10-чной системе: " << n << endl;  cout << "Число в 2-чной системе: ";  toBin(n);  cout << endl;  cout << "Маска в 10-чной системе: " << mask << endl;  cout << "Маска в 2-чной системе: ";  toBin(mask);  cout << endl;  cout << "После обработки" << endl;  cout << "Число в 10-чной системе: " << logOr(n, mask) << endl;  cout << "Число в 2-чной системе: ";  toBin(logOr(n, mask));  cout << endl;  break;  }  case 2: {  cout << "Введите число в 10-чной системе: ";  cin >> n;  mask = 0xFFFFFFC7;  cout << "До обработки" << endl;  cout << "Число в 10-чной системе: " << n << endl;  cout << "Число в 2-чной системе: ";  toBin(n);  cout << endl;  cout << "Маска в 10-чной системе: " << mask << endl;  cout << "Маска в 2-чной системе: ";  toBin(mask);  cout << endl;  cout << "После обработки" << endl;  cout << "Число в 10-чной системе: " << logAnd(n, mask) << endl;  cout << "Число в 2-чной системе: ";  toBin(logAnd(n, mask));  cout << endl;  break;  }  case 3: {  cout << "Введите число в 10-чной системе: ";  cin >> n;  cout << "До обработки" << endl;  cout << "Число в 10-чной системе: " << n << endl;  cout << "Число в 2-чной системе: ";  toBin(n);  cout << endl;  cout << "После обработки" << endl;  cout << "Число в 10-чной системе: " << logMult(n) << endl;  cout << "Число в 2-чной системе: ";  toBin(logMult(n));  cout << endl;  break;  }  case 4: {  cout << "Введите число в 10-чной системе: ";  cin >> n;  cout << "До обработки" << endl;  cout << "Число в 10-чной системе: " << n << endl;  cout << "Число в 2-чной системе: ";  toBin(n);  cout << endl;  cout << "После обработки" << endl;  cout << "Число в 10-чной системе: " << logDiv(n) << endl;  cout << "Число в 2-чной системе: ";  toBin(logDiv(n));  cout << endl;  break;  }  case 5: {  cout << "Введите число в 10-чной системе: ";  cin >> n;  cout << "Введите номер бита для изменения: ";  cin >> b;  mask = 0x80000000;  mask = ~abs(mask >> 31 - b);  cout << "До обработки" << endl;  cout << "Число в 10-чной системе: " << n << endl;  cout << "Число в 2-чной системе: ";  toBin(n);  cout << endl;  cout << "Маска в 10-чной системе: " << abs(mask) << endl;  cout << "Маска в 2-чной системе: ";  toBin(mask);  cout << endl;  cout << "После обработки" << endl;  cout << "Число в 10-чной системе: " << logAnd(n, mask) << endl;  cout << "Число в 2-чной системе: ";  toBin(logAnd(n, mask));  cout << endl;  break;  }  default: {  cout << "Неправильный номер" << endl;  system("pause");  exit(0);  break;  }  }  system("pause");  } |