|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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-33-22 | Шило Ю.С. |
| Принял преподаватель | Филатов А.С. |

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

Москва 2023

# **Цель работы**

Получение навыков применения поразрядных операций в алгоритмах.

1. **Постановка задачи**
2. 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 №10.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вариант | Задание 1 | Задание 2 | Задание 3 | Задание 4 | Задание 5 |
| 10 | Четыре младших | 3-ий, 11-ый, 5-ый | 16 | 16 | Установить n-ый бит в 1, используя маску 2 |

# **Решение**

Побитовые операции, также известные как побитовые операторы, представляют собой способ манипуляции с отдельными битами чисел в их двоичном представлении. Они предоставляют дополнительный уровень контроля над структурой чисел, позволяя выполнять различные операции над их битами.

В языке программирования C++, вы можете использовать следующие побитовые операции:

* 1. Побитовое И (&) - выполняет побитовую конъюнкцию (логическое И) между соответствующими битами двух операндов. Результат будет установлен в 1 только в случае, если оба бита равны 1.
  2. Побитовое ИЛИ (|) - выполняет побитовую дизъюнкцию (логическое ИЛИ) между соответствующими битами двух операндов. Результат будет установлен в 1, если хотя бы один из битов равен 1.
  3. Побитовое исключающее ИЛИ (^) - выполняет побитовую исключающую дизъюнкцию (логическое исключающее ИЛИ) между соответствующими битами двух операндов. Результат будет установлен в 1 только в случае, если биты операндов различны.
  4. Побитовый сдвиг влево (<<) - выполняет сдвиг всех битов числа влево на определенное количество позиций. При этом, свободные места справа заполняются нулями.
  5. Побитовый сдвиг вправо (>>) - выполняет сдвиг всех битов числа вправо на определенное количество позиций. При этом, свободные места слева заполняются нулями.
  6. Побитовая инверсия (~) - выполняет инверсию всех битов числа, заменяя все нули на единицы и наоборот.

Эти побитовые операции в языке программирования C++ позволяют вам более гибко управлять битами чисел и использовать их для решения различных задач в программировании.

1. Функция cout\_2sc была создана для вывода всех заданий. Она преобразует беззнаковое целое число в строку, представляя его значение в двоичной системе счисления с определенным количеством бит. Возвращаемое значение функции - полученная строка.

|  |
| --- |
| std::string cout\_2sc(unsigned int x){  std::string str;  int n = sizeof (int) \* 8;  unsigned mask = (1<<(n-1));  for(int i = 1; i <= n; i++){  str += std::to\_string(((x & mask)>>(n-i)));  mask = mask>>1;  }  return str;  } |

1. Функция Number\_1 была создана для решения первого упражнения. Она устанавливает биты в заданном числе x, используя битовую маску mask. Для этого она выполняет операцию побитового ИЛИ между x и mask, чтобы установить все биты, которые установлены в mask, в x.

|  |
| --- |
| int Number\_1(int x){  int mask = 0x000F;  return x | mask; } |

1. Функция Number\_2 была создана для решения второго упражнения. Она сбрасывает биты в заданном числе x, используя битовую инвертированную маску mask. Для этого она выполняет операцию побитового И между mask и x, чтобы сбросить все биты, которые установлены в mask, в x.

|  |
| --- |
| int Number\_2(int x){  int mask = ~0x0828;  return x & mask; } |

1. Функция Number\_3 была создана для решения третьего упражнения. Она умножает заданное число на 16, сдвигая его биты влево на 4 позиции. Это эквивалентно умножению на 2 в степени 4, так как каждый сдвиг влево на 1 позицию удваивает число.

|  |
| --- |
| int Number\_3(int x){  return x << 4;  } |

1. Функция Number\_4 была создана для решения четвертого упражнения. Она делит заданное число на 16, сдвигая его биты вправо на 4 позиции. Это эквивалентно делению на 2 в степени 4, так как каждый сдвиг вправо на 1 позицию уменьшает число вдвое.

|  |
| --- |
| int Number\_4(int x){  return x >> 4;  } |

1. Функция Number\_5 была создана для решения пятого упражнения. Она устанавливает 1 в n-й бит в заданном числе. Для этого функция создает битовую маску mask\_2, а затем выполняет операцию побитового ИЛИ между mask\_2 и x, чтобы установить 1 в n-й бит в x.

|  |
| --- |
| int Number\_5(int x, int n){  unsigned int mask\_2 = 0x80000000 >> 31 - n;  return mask\_2 | x;  } |

При запуске программы пользователь может ввести два числа одно число, над которым мы будем выполнять операции, второе за номер бита в пятом задании.

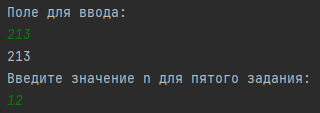


Рисунок 1. Интерфейс программы

# **Тестирование**

Протестируем работу написанной нами программы. Для этого введем любое целочисленное значение, например, 213 и число отвечающие за n, например 12, в бедующем оно будет использовано в пятом задании. В результате отработки программы должны получить следующие данные: первое задание - 00000000000000000000000011011111, второе - 00000000000000000000000011010101, третье - 00000000000000000000110101010000, четвертое - 00000000000000000000000000001101 и пятое - 00000000000000000001000011010101. Вывод, продемонстрированный на рисунке 2, подтверждает корректность работы нашей программы.

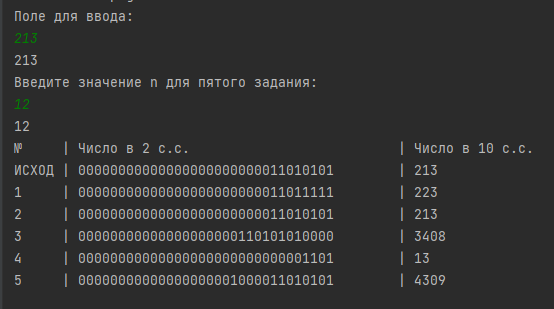


Рисунок 2. Первое тестирование работы программы

Для того чтобы убедиться в корректности работы программы введем другие целочисленные значения, например, 32 и 16. В результате отработки программы должны получить следующие данные: первое задание - 00000000000000000000000000101111, второе - 00000000000000000000000000000000, третье - 00000000000000000000001000000000, четвертое – 00000000000000000000000000000010 и пятое - 00000000000000010000000000100000. Вывод, продемонстрированный на рисунке 3, подтверждает корректность работы нашей программы.

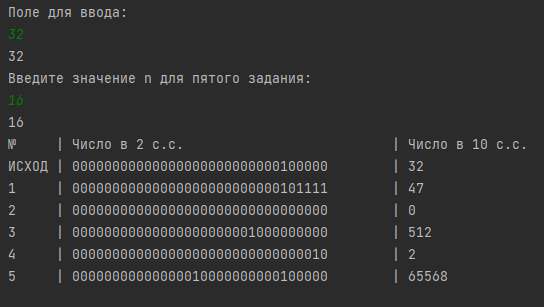


Рисунок 3. Второе тестирование программы

# **Вывод**

В результате выполнения работы я освоил алгоритмы работы с поразрядными операциями и их реализацию на языке программирования C++.

# **Исходный код программы**

|  |
| --- |
| #include <iostream> #include "string" std::string cout\_2sc(unsigned int x){  std::string str;  int n = sizeof (int) \* 8;  unsigned mask = (1<<(n-1));  for(int i = 1; i <= n; i++){  str += std::to\_string(((x & mask)>>(n-i)));  mask = mask>>1;  }  return str; } // Задание №1 // Четыре младших разряда int Number\_1(int x){  int mask = 0x000F; // => 1111 в 2с.с. => F 16с.с.  return x | mask; }  // Задание №2 // 3-ий 5-ый 11-ый int Number\_2(int x){  int mask = ~0x0828;  return x & mask; } // Задание №3 // 16 => 2^4 int Number\_3(int x){  return x << 4; } // Задание №4 // 16 => 2^4 int Number\_4(int x){  return x >> 4; } // Задание №5 // Установить n-ый бит в 1, используя маску 2 int Number\_5(int x, int n){  unsigned int mask\_2 = 0x80000000 >> 31 - n;  return mask\_2 | x; } int main() {  int x, n;  setlocale(LC\_ALL, "ru");  system("chcp 65001");  std::cout << "Поле для ввода:" << std::endl;  std::cin >> x;  std::cout << "Введите значение n для пятого задания:" << std::endl;  std::cin >> n;  std::cout << "№ | " << "Число в 2 с.с." << "\t\t\t\t| " << "Число в 10 с.с." << std::endl  << "ИСХОД | " << cout\_2sc(x) << "\t| " << x << std::endl  << "1 | " << cout\_2sc(Number\_1(x)) << "\t| " << Number\_1(x) << std::endl  << "2 | " << cout\_2sc(Number\_2(x)) << "\t| " << Number\_2(x) << std::endl  << "3 | " << cout\_2sc(Number\_3(x)) << "\t| " << Number\_3(x) << std::endl  << "4 | " << cout\_2sc(Number\_4(x)) << "\t| " << Number\_4(x) << std::endl  << "5 | " << cout\_2sc(Number\_5(x, n)) << "\t| " << Number\_5(x, n) << std::endl; } |