|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 ИНБО-03-22 | Алтухов. А.В. |
| 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.

# **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 №2. Условие задания:

|  |  |
| --- | --- |
| Упражнение 1 | Три старших |
| Упражнение 2 | 12-ый, 14-ый, 3-ий |
| Упражнение 3 | 4 |
| Упражнение 4 | 4 |
| Упражнение 5 | Установить n-ый бит в 1, используя маску 2 |

# **Решение**

Поразрядные операции (или операции над битами) — это операции, которые выполняются над отдельными битами в двоичном представлении чисел. Эти операции работают на уровне битов и могут использоваться для манипуляции с данными, кодирования и декодирования информации, а также для решения различных задач в программировании и компьютерных науках. Поразрядные операции широко применяются в программировании, особенно при работе с манипуляциями с данными, битовыми флагами, алгоритмами шифрования, а также в различных областях, таких как обработка изображений, аудио-обработка и многое другое, где требуется точное управление битами данных.

В моем варианте №2 текущей практической работы используются такие поразрядные операции, как: Поразрядное И, Поразрядное ИЛИ, Сдвиг влево и вправо и Поразрядное инвертирование. Теперь о каждой операции подробнее.

Поразрядное И: позволяет установить бит в 1 только в том случае, если оба операнда имеют соответствующие биты, установленные в 1. Эта операция часто используется для маскирования битов и очистки определенных флагов в регистрах. В С++ обозначается “&”.

Поразрядное ИЛИ: устанавливает бит в 1, если хотя бы один из операндов имеет соответствующий бит, установленный в 1. Это можно использовать для установки определенных флагов или объединения битов из разных источников. В С++ обозначается “ | ”.

Поразрядное инвертирование (или поразрядное отрицание): инвертирует каждый бит в числе, то есть 1 становится 0, а 0 становится 1. Эта операция полезна для инвертирования флагов или битовых масок. В С++ обозначается “~”.

Сдвиги позволяют перемещать биты числа влево или вправо. Сдвиг влево эквивалентен умножению числа на 2^n, где n - количество сдвигаемых позиций. Сдвиг вправо эквивалентен делению числа на 2^n с отбрасыванием дробной части исходного числа. Эти операции полезны при работе с битовыми полями и оптимизации вычислений. В С++ обозначаются соответственно “<<” и “>>”.

Для перевода чисел из десятичной системы счислений в двоичную была написана функция bitout. Функция принимает на вход две целочисленные переменные – число, которое будет переведено в двоичную систему счисления, и количество разрядов, которые будут отображаться, и возвращает исходное число в двоичной системе счисления в виде строки.

|  |
| --- |
| //Функция перевода числа в двоичную систему счисления из заданной  string bitout(unsigned int x, int b)  {  int xc = x;  string result;  for (int i = 0; i < b; i++)  {  if (x % 2 == 1) result = "1" + result;  else result = "0" + result;  x = x / 2;  }  return result + " " + to\_string(xc);  } |

Для решения первого упражнения была написана функция Task1, выполняющая операцию Поразрядное ИЛИ с маской, заданной в моём варианте. Функция принимает на вход маску и целочисленную переменную, с которой требуется выполнить поразрядную операцию, а возвращает результат операции.

|  |
| --- |
| //Функция, которая ставит необходимые биты в значение 1  int Task1(int ch, int mask) {  ch |= mask;  return ch;  } |

Для решения второго упражнения была написана функция Task2, выполняющая операцию Поразрядное И с маской, заданной в моём варианте. Функция принимает на вход маску и целочисленную переменную, с которой требуется выполнить поразрядную операцию, а возвращает результат операции.

|  |
| --- |
| //Функция, которая обнуляет необходимые разряды  int Task2(int ch, int mask) {  return ch & mask;  } |

Для решения третьего упражнения была написана функция Task3, которая выполняет сдвиг влево. Функция принимает на вход целочисленную переменную, с которой будет выполняться поразрядная операция, а возвращает ее результат.

|  |
| --- |
| //Функция, которая поразрядно умножает число на 2  int Task3(int ch) {  return ch << 2;  } |

Для решения четвертого упражнения была написана функция Task4, которая выполняет сдвиг вправо. Функция принимает на вход целочисленную переменную, с которой будет выполняться поразрядная операция, а возвращает ее результат.

|  |
| --- |
| //Функция, которая поразрядно делит число на 2  int Task4(int ch) {  return ch >> 2;  } |

Для решения пятого упражнения была написана функция Task5, которая устанавливает заданный разряд в 1, используя маску 2. Функция принимает на вход 2 целочисленные переменные – исходное число, с которой будет выполняться поразрядная операция, и номер разряда, который будет установлен в 1, а возвращает результат поразрядной операции.

|  |
| --- |
| //Функция, которая устанавливает в 1 n-ый разряд с помощью маски 2  int Task5(int ch, int n) {  unsigned int mask = 0x1 << 31;  mask = mask >> (31 - n);  ch |= mask;  return ch;  } |

При запуске программы пользователю необходимо поочередно вводить данные для четырех упражнений, начиная со второго.

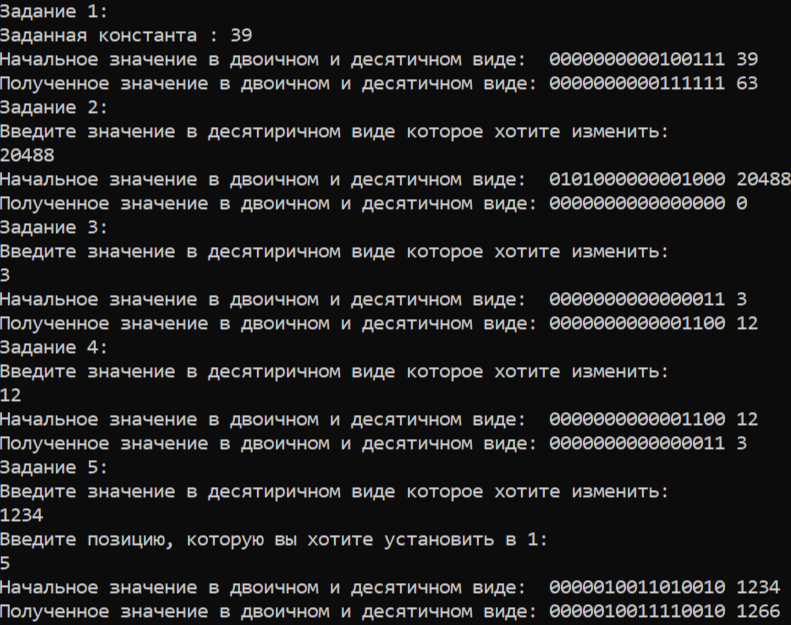


Рисунок 1. Интерфейс программы

# **Тестирование**

Протестируем работу программы. Для тестирования первого упражнения нам не требуется вводить данных. Выбран наглядный случай, который демонстрирует корректность выполнения функции, что мы можем видеть на рисунке 2.

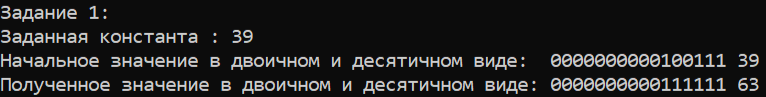


Рисунок 2. Результат работы функции Task1

Для тестирования второго упражнения нам понадобится рассмотреть несколько случаев. Введем число 20488, у которого единицы только в разрядах 3, 12 и 14, которые и требуется обнулить. Посмотрим на рисунок 3 и убедимся, что программа выведет число 0.

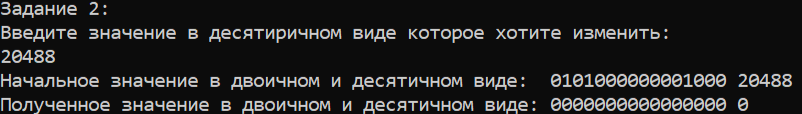


Рисунок 3. Результат работы функции Task2

Попробуем ввести число 21499 и убедимся, глядя на рисунок 4, что функция не затрагивает лишние разряды.

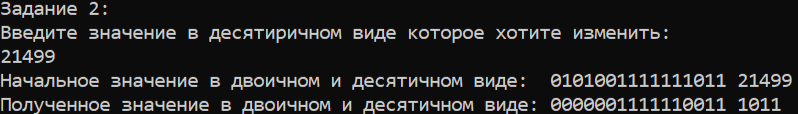


Рисунок 4. Результат работы функции Task2

Теперь введем число 101, которое не содержит единиц в заданных разрядах и убедимся по рисунку 5, что функция не изменит число.

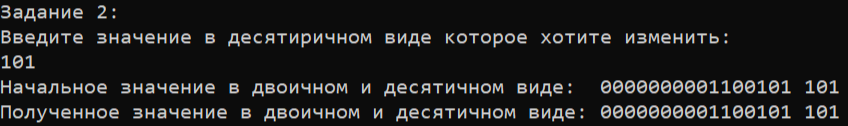


Рисунок 5. Результат работы функции Task2

Для тестирования функции, предназначенной для третьего упражнения, достаточно будет двух примеров. Рассмотрим ситуацию, когда поступает любое натуральное число, в нашем случае число 3. Посмотрим на рисунок 6 и убедимся, что функция корректно умножило число.

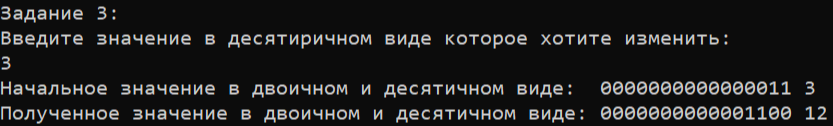


Рисунок 6. Результат работы функции Task3

Для второго случая рассмотрим умножение на ноль. Как мы видим по рисунку 7, функция не берет единицы из ниоткуда, на результате так же получается ноль.

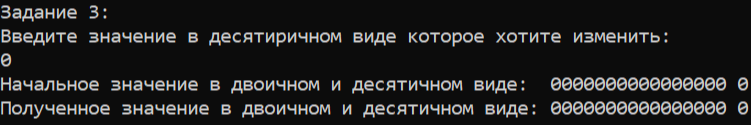


Рисунок 7. Результат работы функции Task3

Для тестирования функции, предназначенной для четвертого упражнения, нам потребуется 3 случая. Рассмотрим ситуацию, когда число кратно 2-ум, например, число 384. Посмотрим на рисунок 8 и убедимся, что данный случай обработан корректно.

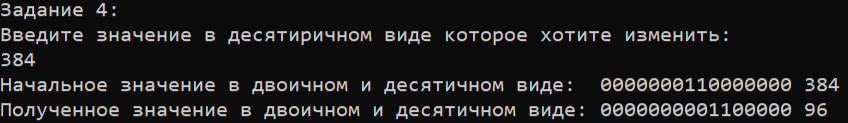


Рисунок 8. Результат работы функции Task4

Теперь рассмотрим ситуацию, когда число не кратно 2. Возьмем число 403. По рисунку 9 видно, что функция верно выполняет целочисленное деление.

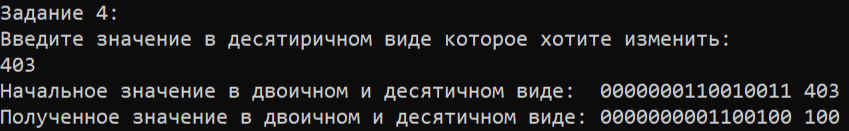


Рисунок 9. Результат работы функции Task4

Убедимся, что единицы не появляются из ниоткуда, и попробуем на 2 поделить 0. По рисунку 10 можем убедиться, что все верно.

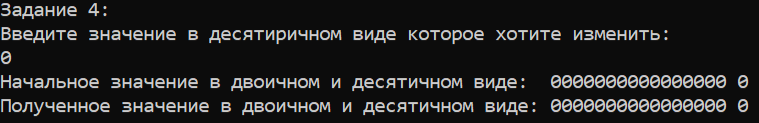


Рисунок 10. Результат работы функции Task4

Для тестирования функции, предназначенной для пятого упражнения, рассмотрим несколько случаев. Попробуем в числе 1234 поставить 5 разряд в единицу. Результат представлен на рисунке 11.

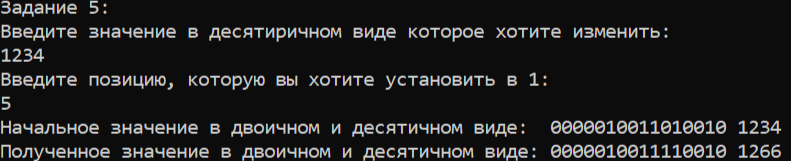


Рисунок 11. Результат работы функции Task5

Теперь попробуем в числе 3 поставить в единицу 1-й разряд, который и так равен 1. Это будет число 3. Результат представлен на рисунке 12.

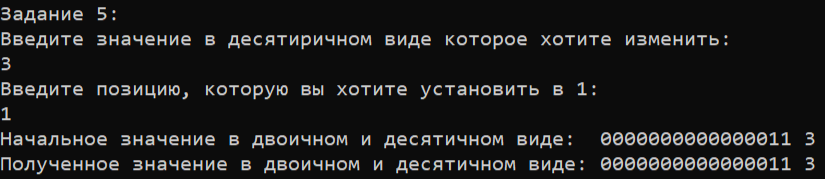


Рисунок 12. Результат работы функции Task5

# **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 <iomanip>  #include <iostream>  #include <format>  #include <string>  using namespace std;  string bitout(unsigned int x, int b)  {  int xc = x;  string result;  for (int i = 0; i < b; i++)  {  if (x % 2 == 1) result = "1" + result;  else result = "0" + result;  x = x / 2;  }  return result + " " + to\_string(xc);  }  int Task1(int ch, int mask) {  ch |= mask;  return ch;  }  int Task2(int ch, int mask) {  return ch & mask;  }  int Task3(int ch) {  return ch << 2;  }  int Task4(int ch) {  return ch >> 2;  }  int Task5(int ch, int n) {  unsigned int mask = 0x1 << 31;  mask = mask >> (31 - n);  ch |= mask;  return ch;  }  int main()  {  setlocale(LC\_ALL, "Russian");  cout << "Задание 1: " << endl;  int ch = 0x27;  cout << "Заданная константа : " << ch << endl;  int mask2 = 0b00111000;  cout << "Начальное значение в двоичном и десятичном виде: " << bitout(ch, 16) << endl;  cout << "Полученное значение в двоичном и десятичном виде: " << bitout(Task1(ch, mask2), 16) << endl;  cout << "Задание 2: " << endl;  int ch2;  cout << "Введите значение в десятиричном виде которое хотите изменить: " << endl;  cin >> ch2;  int mask = 0xAFF7;  cout << "Начальное значение в двоичном и десятичном виде: " << bitout(ch2, 16) << endl;  cout << "Полученное значение в двоичном и десятичном виде: " << bitout(Task2(ch2, mask), 16) << endl;  cout << "Задание 3: " << endl;  int ch3;  cout << "Введите значение в десятиричном виде которое хотите изменить: " << endl;  cin >> ch3;  cout << "Начальное значение в двоичном и десятичном виде: " << bitout(ch3, 16) << endl;  cout << "Полученное значение в двоичном и десятичном виде: " << bitout(Task3(ch3), 16) << endl;    cout << "Задание 4: " << endl;  int ch4;  cout << "Введите значение в десятиричном виде которое хотите изменить: " << endl;  cin >> ch4;  cout << "Начальное значение в двоичном и десятичном виде: " << bitout(ch4, 16) << endl;  cout << "Полученное значение в двоичном и десятичном виде: " << bitout(Task4(ch4), 16) << endl;  cout << "Задание 5: " << endl;  int ch5;  cout << "Введите значение в десятиричном виде которое хотите изменить: " << endl;  cin >> ch5;  cout << "Введите позицию, которую вы хотите установить в 1: " << endl;  int n;  cin >> n;  cout << "Начальное значение в двоичном и десятичном виде: " << bitout(ch5, 16) << endl;  cout << "Полученное значение в двоичном и десятичном виде: " << bitout(Task5(ch5, n), 16) << endl;  } |