|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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-03-21 | Хречко С.В. |
| 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 №11. Условие задания:

|  |  |
| --- | --- |
| Упражнение 1 | С 5-ого четыре слева |
| Упражнение 2 | 5-ый, 7-ой справа |
| Упражнение 3 | 32 |
| Упражнение 4 | 32 |
| Упражнение 5 | Обнулить n-ый бит, используя маску 1 |

# **Решение**

Побитовые операции – это операции, работающие напрямую с битами данных.

|  |  |  |
| --- | --- | --- |
| x<<n | Сдвиг влево двоичного кода (умножение на 2n) | unsigned int x=7; x=x<<2;  результат 0х0000001С |
| x>>n | Сдвиг вправо двоичного кода (деление на 2n) | unsigned int x=28; x=x>>2;  результат =0х00000007 |
| x & maska | Поразрядное И (применяется для записи в указанный разряд 0) | Правило выполнения операции  111  & 100  = 100  Установить в двоичном коде переменной ***х*** только 9-ый справа бит в 0  unsigned short int x=0xAEFF;  unsigned short int maska=0xFDFF;  x=x & maska результат 0xACFF |
| X | maska | Поразрядное ИЛИ (применяется для записи в указанный разряд 1) | Правило выполнения операции  111  | 100  111  Установить в двоичном коде переменной х 9-ый справа бит в 1  unsigned short int x=0xACFF;  unsigned short int maska=0x0200;  x=x | maska результат 0xAEFF; |
| X ^ maska | Исключающее ИЛИ для поразрядных операций.  Используется для проверки соответствующих битов двух переменных, если они имеют разные значения, то результат 1, а если равны, то 0. | Правило выполнения операции  1111  ^ 0001  = 1110  unsigned int x=0xF, a=1;  a=x^a;  Результат: в переменной ***а*** значение  0х0000000Е |
| ~ | Инверсия (0 заменяет на 1, а 1 на 0) | x=0x0F;  ~x;  результат 0хF0 |

В первом упражнении требовалось установить в 1 с пятого бита четыре слева в числе определенном разработчиком, поэтому используется маска, в которой четыре бита, начиная с пятого, установлены в единицы, остальные в ноль, далее применяется поразрядная дизъюнкция, как следует из краткой теоретической справки.

|  |
| --- |
| unsigned int firstTask(unsigned int x)  {      unsigned int mask = 0b111100000;      return(x | mask);  } |

Во втором упражнении требовалось установить в 0 5-й и 7-й биты, поэтому используется поразрядная конъюнкция с маской, в которой 5-й и 7-й биты установлены в ноль, остальные в единицу. Для получения такой маски используется поразрядная инверсия маски, в которой все биты кроме 5-ого и 7-ого установлены в 0.

|  |
| --- |
| unsigned int secondTask(unsigned int x)  {      unsigned int mask = 0b10100000;      mask = ~mask;      return(x & mask);  } |

В третьем упражнении требовалось умножить число на 32. Это можно сделать побитовым сдвигом числа налево на 5 бит, так как логарифм по основанию 2 из 32 равен 5.

|  |
| --- |
| unsigned int thirdTask(unsigned int x)  {      return(x << 5);  } |

В четвертом упражнении требовалось разделить число на 32. Это можно сделать побитовым сдвигом числа направо на 5 бит, так как логарифм по основанию 2 из 32 равен 5.

|  |
| --- |
| unsigned int fourthTask(unsigned int x)  {      return(x >> 5);  } |

В пятом упражнении требуется обнулить n-ый бит, задание выполняется по аналогии со вторым, но бит, который требуется обнулить передается в функцию, поэтому маску требуется создать, по заданию начиная с 1 в нулевом бите. Совершая n побитовых сдвигов влево, мы получаем маску, которая после поразрядной инверсии представляет число, все биты которого кроме n-ого равны единице. Поразрядная конъюнкция с которым позволяет обнулить n-ый бит.

|  |
| --- |
| unsigned int fifthTask(unsigned int x, int n)  {      unsigned int mask = 1;      mask = mask << n;      return(x & (~mask));  } |

Функция coutp выводит на экран переданное число в битовом виде, количество выводимых битов определяется n, которое в данном случае задано количеством битов в int. Далее в цикле с помощью маски полностью состоящей из единиц и побитовых сдвигов, для получения значения каждого отдельного бита, побитово выводится число на печать.

|  |
| --- |
| void coutp(unsigned int x)  {      int n=sizeof(int)\*8;      unsigned maska=(1<<(n-1));      for(int i=1; i<=n;i++)      {          cout<<((x&maska)>>(n-i));          maska=maska>>1;      }      cout << endl;  } |

Пользовательский интерфейс предлагает выбрать какое упражнение выполнять (рис. 1).

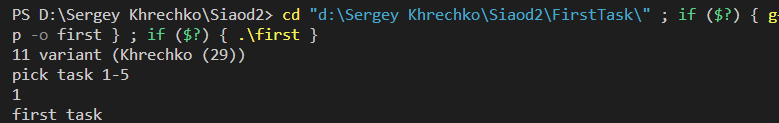


Рисунок 1. Интерфейс программы

# **Тестирование**

Тестирование первого упражнения. Специально подобранный x = A7 = 167 среди четырех битов, начиная с 5-ого, налево, как единицы так и нули, и мы можем наблюдать, что программа устанавливает в единицу все четыре и не меняет остальные биты (рис. 2).

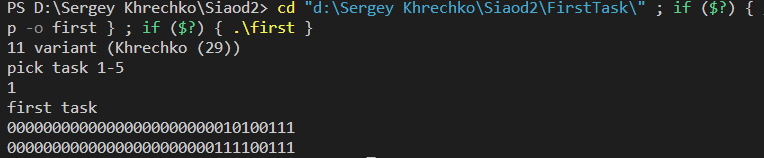


Рисунок 2. Тестирование 1 упражнения 1

Для второго теста было использовано число 7. В нем нет единиц на позициях, которые программа должна установить в 1, а значит, мы можем проверить, что все позиции действительно устанавливаются (рис. 3).

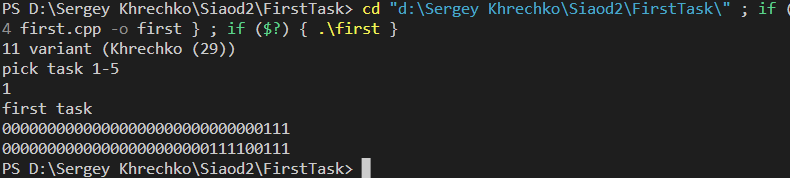


Рисунок 3. Тестирование 2 упражнения 1

Тестирование второго упражнения. В числе 294 7-й бит равен 0, а 5-й равен 1, мы можем наблюдать, что для обоих значений программа устанавливает биты в ноль (рис. 4).

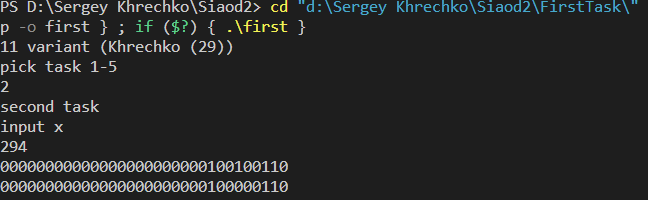


Рисунок 4. Тестирование 1 упражнения 2

Для следующего теста было взято число 160, его 5-й и 7-й биты равны 1, а значит, мы можем удостовериться, что оба бита устанавливаются в 0 (рис. 5).

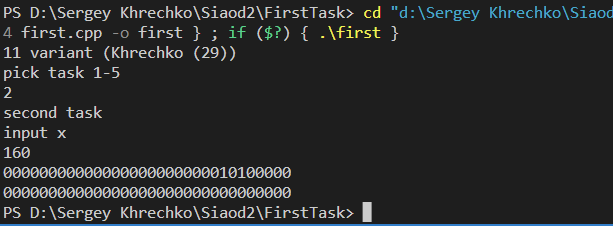


Рисунок 5. Тестирование 2 упражнения 2

Тестирование третьего упражнения. Число 4 взятое для проверки, правильно умножается на 32 (рис. 6).

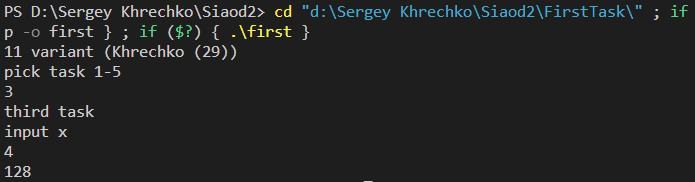


Рисунок 6. Тестирование 1 упражнения 3

Умножение нуля на 32 также дает верный ответ (рис. 7).

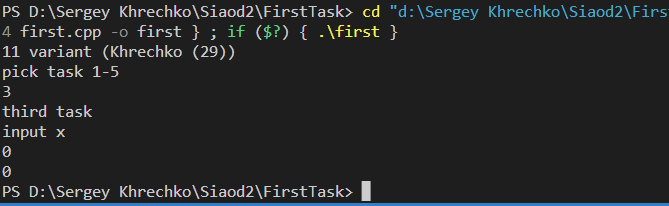


Рисунок 7. Тестирование 2 упражнения 3

Тестирование четвертого упражнения. Для проверки было взято число 130, которое не делится нацело на 32, однако в соответствии с ожиданиями целочисленных вычислений мы получаем ответ 4 (рис. 8).

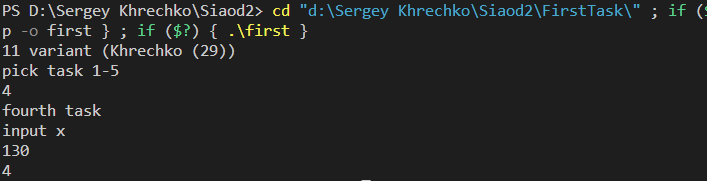


Рисунок 8. Тестирование 1 упражнения 4

Деление числа делящегося нацело также дает верный результат (рис. 9).

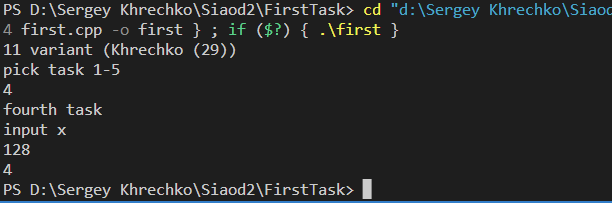


Рисунок 9. Тестирование 2 упражнения 4

Использовав первое пришедшее в голову число и видя его побитовое представление, можно выбрать бит, который обнулить; биты нумеруются с нуля. Из (рис. 10) видно, что заданный бит обнуляется.

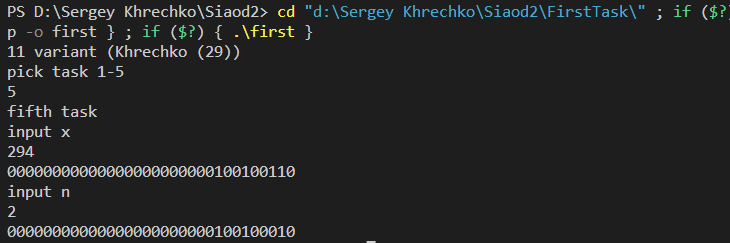


Рисунок 10. Тестирование 1 упражнения 5

Если нужно обнулить бит, в котором уже ноль, то ноль и останется (рис. 11).

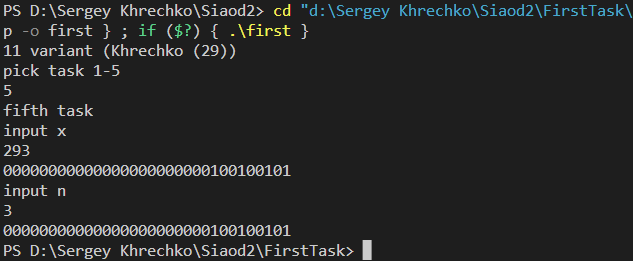


Рисунок 11. Тестирование 2 упражнения 5

Из результатов выполнения программы видно:

1. Программа работает правильно и выполнены все упражнения.

# **Вывод**

В результате выполнения работы я:

1. Освоил алгоритмы работы с поразрядными операциями и их реализацию на языке программирования C++

# **Исходный код программы**

|  |
| --- |
| #include <iostream>  using namespace std;  void coutp(unsigned int x);  unsigned int firstTask(unsigned int x);  unsigned int secondTask(unsigned int x);  unsigned int thirdTask(unsigned int x);  unsigned int fourthTask(unsigned int x);  unsigned int fifthTask(unsigned int x, int n);  int main()  {      cout << "11 variant (Khrechko (29))" << endl;      cout << "pick task 1-5" << endl;      //I don't do protection against wront input      int pick;      cin >> pick;      unsigned int x;      switch (pick)      {      case 1:          cout << "first task" << endl;          x = 0x7;          coutp(x);          coutp(firstTask(x));          break;      case 2:          cout << "second task" << endl;          cout << "input x" << endl;          cin >> x;          coutp(x);          coutp(secondTask(x));          break;      case 3:          cout << "third task" << endl;          cout << "input x" << endl;          cin >> x;          cout << thirdTask(x);          break;      case 4:          cout << "fourth task" << endl;          cout << "input x" << endl;          cin >> x;          cout << fourthTask(x);          break;      case 5:          cout << "fifth task" << endl;          cout << "input x" << endl;          cin >> x;          coutp(x);          cout << "input n" << endl;          int n;          cin >> n;          coutp(fifthTask(x ,n));          break;      default:          cout << "wrong input" << endl;          break;      }      return 0;  }  unsigned int firstTask(unsigned int x)  {      unsigned int mask = 0b111100000;      return(x | mask);  }  unsigned int secondTask(unsigned int x)  {      unsigned int mask = 0b10100000;      mask = ~mask;      return(x & mask);  }  unsigned int thirdTask(unsigned int x)  {      return(x << 5);  }  unsigned int fourthTask(unsigned int x)  {      return(x >> 5);  }  unsigned int fifthTask(unsigned int x, int n)  {      unsigned int mask = 1;      mask = mask << n;      return(x & (~mask));  }  void coutp(unsigned int x)  {      int n=sizeof(int)\*8;      unsigned maska=(1<<(n-1));      for(int i=1; i<=n;i++)      {          cout<<((x&maska)>>(n-i));          maska=maska>>1;      }      cout << endl;  } |