**Титульный лист материалов по дисциплине**

|  |  |
| --- | --- |
| ДИСЦИПЛИНА | **Структуры и алгоритмы обработки данных (ч. 2)** |
|  | (полное наименование дисциплины без сокращений) |
| ИНСТИТУТ | **ИТ** |
| КАФЕДРА | **Математического обеспечения и стандартизации информационных технологий** |
|  | (полное наименование кафедры) |
| ВИД УЧЕБНОГО | **Практические работы** |
| МАТЕРИАЛА | (в соответствии с пп.1-11) |
| ПРЕПОДАВАТЕЛЬ | **Муравьёва Екатерина Андреевна** |
|  | (фамилия, имя, отчество) |
| СЕМЕСТР | **3 семестр, 2023-2024 уч. год** |
|  | (указать семестр обучения, учебный год) |

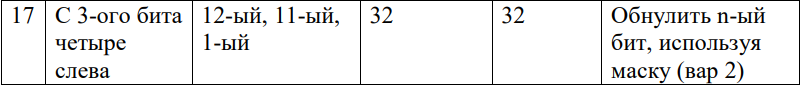
|  |  |
| --- | --- |
| Выполнил | **Козлов К.И.** |
|  | (фамилия, имя, отчество) |

**Практическая работа № 1**

**Цель работы**

Освоить приёмы работы с битовым представлением беззнаковых целых чисел, реализовать эффективный алгоритм сортировки на основе битового массива.

**Ход работы**

**Вариант 17** 

**Задание 1**

**Формулировка задачи:**

1) Определить переменную целого типа, присвоить ей значение, используя константу в шестнадцатеричной системе счисления. Разработать оператор присваивания и его выражение, которое установит заданные в задании биты исходного значения переменной в значение 1, используя соответствующую маску и поразрядную операцию.

2) Определить переменную целого типа. Разработать оператор присваивания и его выражение, которое обнуляет заданные в задании биты исходного значения переменной, используя соответствующую маску и поразрядную операцию. Значение в переменную вводится с клавиатуры.

3) Определить переменную целого типа. Разработать оператор присваивания и выражение, которое умножает значение переменной на число, указанное в третьем столбце варианта, используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

4) Определить переменную целого типа. Разработать оператор присваивания и выражение, которое делит значение переменной на число, указанное в четвертом столбце варианта, используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

5) Определить переменную целого типа. Разработать оператор присваивания и выражение, в котором используются только поразрядные операции. В выражении используется маска – переменная. Маска может быть инициализирована единицей в младшем разряде (вар 1) или единицей в старшем разряде (вар 2). Изменяемое число вводится с клавиатуры.

**Решение:**

Для решения данных подзадач первого задания были разработаны следующие функции, приведенные в листинге 1.

Первая функция побитово складывает введённое пользователем число с маской (0078 в 16-ичной системе счисления), чтоб установить 4 бита слева от третьего в значение 1.

Вторая функция побитово умножает введенное пользователем число на маску (7fd в 16-ичной системе счисления или 2045 в десятичной системе счисления), чтоб обнулить 12, 11 и 1 биты введенного пользователем числа.

Третья функция побитово сдвигает введенное пользователем число на 5 битов влево, тем самым умножая введенное пользователем число на 32.

Четвертая функция побитово сдвигает введенное пользователем число на 5 битов вправо, тем самым производя деление введенного пользователем числа на 32.

Пятая функция, в зависимости от нужного разряда или позиции, обнуляет бит введенного пользователем числа на этой позиции путем применения поразрядной операции исключающего или.

|  |
| --- |
| unsigned short SubTaskOne(unsigned short UserNum)  {  return(0x0078 | UserNum);  }  unsigned short SubTaskTwo(unsigned short UserNum)  {  return(0xe7fd & UserNum);//2045//  }  unsigned short SubTaskThree(unsigned short UserNum)  {  return(UserNum << 5);  }  unsigned short SubTaskFour(unsigned short UserNum)  {  return(UserNum >> 5);  }  unsigned short SubTaskFive(unsigned short UserNum, unsigned short PositionOfBit)  {  unsigned short bitMask = 1;  for (int i = 0; i < PositionOfBit; i++)  {  bitMask \*= 2;  }  return(bitMask ^ UserNum);  } |

Листинг 1 – код задач 1.1-1.5

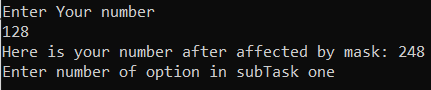


Рисунок 1 - результат тестирования 1.1

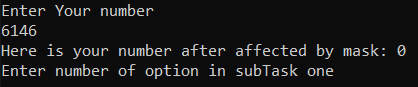


Рисунок 2 – результат тестирования 1.2

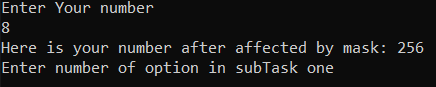


Рисунок 3 – результат тестирования 1.3

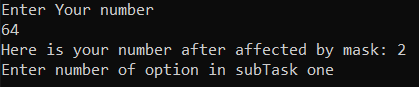


Рисунок 4 – результат тестирования 1.4

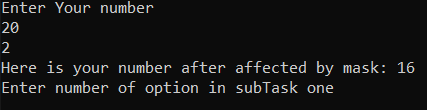


Рисунок 5 – результат тестирования 1.5

Далее представлен листинг всего первого задания.

|  |
| --- |
| case 1:  {  int subTastOneControl;  bool flagSubTaskOne = true;  unsigned short UserNum;  unsigned short bitPlace;  while (flagSubTaskOne)  {  std::cout << "Enter number of option in subTask one" << std::endl;  std::cin >> subTastOneControl;  switch (subTastOneControl)  {  case 1:  {  std::cout << "Enter Your number" << std::endl;  std::cin >> UserNum;  std::cout << "Here is your number after affected by mask: " << SubTaskOne(UserNum) << std::endl;  break;  }  case 2:  {  std::cout << "Enter Your number" << std::endl;  std::cin >> UserNum;  std::cout << "Here is your number after affected by mask: " << SubTaskTwo(UserNum) << std::endl;  break;  }  case 3:  {  std::cout << "Enter Your number" << std::endl;  std::cin >> UserNum;  std::cout << "Here is your number after affected by mask: " << SubTaskThree(UserNum) << std::endl;  break;  }  case 4:  {  std::cout << "Enter Your number" << std::endl;  std::cin >> UserNum;  std::cout << "Here is your number after affected by mask: " << SubTaskFour(UserNum) << std::endl;  break;  }  case 5:  {  std::cout << "Enter Your number" << std::endl;  std::cin >> UserNum;  std::cin >> bitPlace;  std::cout << "Here is your number after affected by mask: " << SubTaskFive(UserNum, bitPlace) << std::endl;  break;  }  }  }  } |

Рисунок 6 – полный листинг кода к заданию 1.

**Задание 2**

Формулировка задачи:

1. Реализовать пример с вводом произвольного набора до 8-ми чисел (со значениями от 0 до 7) и его сортировкой битовым массивом в виде числа типа unsigned char.
2. Исправьте программу задания, чтобы для сортировки набора из 64-х чисел использовалось не одно число типа unsigned long long, а линейный массив чисел типа unsigned char.

**Решение:**

При решении данного задания был создан динамический массив, чтоб пользователь смог сам выбирать количество вводимых им чисел. Также были созданы две переменные unsigned char и массив unsigned char, чтоб отсортировать введенные пользователем числа.

|  |
| --- |
| case 2:  {  bool subTaskFlag = true;  int subTaskTwoControl;  int length;  while (subTaskFlag)  {  std::cout << "1 - Array" << '\n';  std::cout << "2 - Array 64" << '\n';  std::cin >> subTaskTwoControl;  switch (subTaskTwoControl)  {  case 1:  {  std::cout << "Enter length of array" << '\n';  std::cin >> length;  int\* userArray = new int[length];  unsigned char userSort = 0;  for (int i = 0; i < length; i++)  {  std::cin >> userArray[i];  userSort |= (1 << userArray[i]);  }  for (int i = 0; i < 8; i++)  {  if (userSort & (1 << i))  {  std::cout << i << ' ';  }  }  std::cout << '\n';  break;  }  case 2:  {  std::cout << "Enter length of array" << '\n';  std::cin >> length;  int\* userArray = new int[length];  unsigned char userSort[8] = { 0,0,0,0,0,0,0,0 };  for (int i = 0; i < length; i++)  {  std::cin >> userArray[i];  userSort[(userArray[i] / 8)] |= (1 << userArray[i] % 8);  }  for (int i = 0; i < 8; i++)  {  for (int j = 0; j < 8; j++)  {  if (userSort[i] & (1 << j))  {  std::cout << i \* 8 + j << ' ';  }  }  }  std::cout << '\n';  break;  }  case 3:  {  subTaskFlag = false;  break;  }  }  }  break;  } |

Рисунок 7 – полный листинг кода задания 2

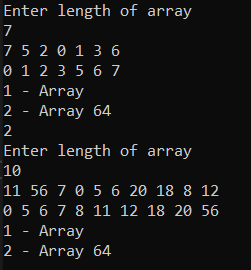


Рисунок 6 – результат тестирования 2

**Задание 3**

Формулировка задачи:

Входные данные: файл, содержащий не более n=107 неотрицательных целых чисел, среди них нет повторяющихся. Результат: упорядоченная по возрастанию последовательность исходных чисел в выходном файле. Время работы программы: ~10 с (до 1 мин. для систем малой вычислительной мощности). Максимально допустимый объём ОЗУ для хранения данных: 1 МБ. Очевидно, что размер входных данных гарантированно превысит 1МБ (это, к примеру, максимально допустимый объём стека вызовов, используемого для статических массивов). Требование по времени накладывает ограничение на количество чтений исходного файла. Реализуйте тестовый пример, демонстрирующий входные данные и заполненный битовый массив (не более 20 чисел). Реализуйте задачу сортировки числового файла для входных данных объемом 100 и 1000 чисел. Показать время выполнения сортировки для каждого объема. Реализуйте задачу сортировки заданного числового файла. В отчёт внесите результаты тестирования для наибольшего количества входных чисел, соответствующего битовому массиву длиной 1МБ и программно определить объём оперативной памяти, занимаемый битовым массивом и время выполнения работы программы (сортировки) для каждого случая.

**Решение:**

Логика выполнения программы и сортировки битовым массивом идентична подзаданию в задании 2. Однако теперь чтение и запись(вывод) производятся из файла и в файл соответственно.

|  |
| --- |
| case 3:  {  char sortedBits[1000000];  int anotherInt;  initializedBits(sortedBits);  std::ifstream fin;  fin.open("unsortedNumsByString.txt");  if (fin.is\_open())  {  std::cout << "file opend" << '\n';  }  while(fin >> anotherInt)  {  sortedBits[(anotherInt / 8)] |= (1 << anotherInt % 8);  }  fin.close();  std::ofstream fout("sortedNumsByString.txt");  auto start = std::chrono::steady\_clock::now();  for (int i = 0; i < 1000000; i++)  {  for (int j = 0; j < 8; j++)  {  if (sortedBits[i] & (1 << j))  {  fout << i \* 8 + j << '\n';  }  }  }  auto end = std::chrono::steady\_clock::now();  std::cout << "Time: " << std::chrono::duration\_cast<std::chrono::milliseconds>(end - start).count() << " miliseconds" << '\n';  fout.close();  break;  } |

Рисунок 8 – полный листинг задания 3.



**Вывод**

Были изучены поразрядные операции, а также сортировка числового файла с помощью битового массива. Был практически понят механизм битовой сортировки. Данная сортировка подходит для создания упорядоченных массивов уникальных элементов, так как она не учитывает повторения.