**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: **ТИПЫ ДАННЫХ И ИХ ВНУТРЕННЕЕ ПРЕДСТАВЛЕНИЕ В ПАМЯТИ**

|  |  |  |
| --- | --- | --- |
| Студент гр. 5372 |  | Телушков А.В |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2025

**Цель работы.**

Изучение и организация типов данных; получение практических навыков работы с типами данных; определение преимуществ и недостатков использования определенных типов данных.

**Основные теоретические положения.**

Внутреннее представление величин целого типа – целое число в двоичном коде. При использовании спецификатора signed старший бит числа интерпретируется как знаковый (0 – положительное число, 1 – отрицательное). Для кодирования целых чисел со знаком применяется прямой, обратный и дополнительный коды.

Представление положительных и отрицательных чисел в прямом, обратном и дополнительном кодах отличается. В прямом коде в знаковый разряд помещается цифра 1, а в разряды цифровой части числа – двоичный код его абсолютной величины. Прямой код числа −3 (для 16- разрядного процессора):

Изображение выглядит как текст, Шрифт, число, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Обратный код получается инвертированием всех цифр двоичного кода абсолютной величины, включая разряд знака: нули заменяются единицами, единицы – нулями. Прямой код можно преобразовать в обратный, инвертировав все значения всех битов (кроме знакового). Обратный код числа −3:

Изображение выглядит как текст, Шрифт, число, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Дополнительный код получается образованием обратного кода с последующим прибавлением единицы к его младшему разряду. Дополнительный код числа −3:

Изображение выглядит как текст, Шрифт, число, мерная рейка

Содержимое, созданное искусственным интеллектом, может быть неверным.

Увидеть, каким образом тип данных представляется на компьютере, можно при помощи логических операций: побитового сдвига (<<) и поразрядной конъюнкции (&).

putchar(value & mask ? '1' : '0'); // если 1, то возвращается 1, иначе 0

value <<= 1; // побитовый сдвиг влево на 1 бит

Putchar возвращает один символ в консоль. Альтернатива - cout. В представленном способе, маска - то, с чем сравнивается значение. И побитовый сдвиг применяется для value. Таким образом 1 бит будет сравниваться с каждым битом числа. Альтернатива - побитовый сдвиг вправо, но при этом нужно проводить данную операцию не над значением(единицей), а над маской (исходым числом, битовое представление которого нужно получить).

При сдвиге вправо для чисел без знака позиции битов, освобожденные при операции сдвига, заполняются нулями. Для чисел со знаком бит знака используется для заполнения освобожденных позиций битов. Другими словами, если число 25 является положительным, используется 0, если число является отрицательным, используется 1. При сдвиге влево позиции битов, освобожденных при операции сдвига, заполняются нулями. Сдвиг влево является логическим сдвигом (биты, сдвигаемые с конца, отбрасываются, включая бит знака).

Вещественные типы данных хранятся в памяти компьютера иначе, чем целочисленные. Внутреннее представление вещественного числа состоит из двух частей – мантиссы и порядка.

Для 32-разрядного процессора для float под мантиссу отводится 23 бита, под экспоненту – 8, под знак – 1. Для double под мантиссу отводится 52 бита, под экспоненту – 11, под знак – 1:

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Увидеть, каким образом вещественные типы данных представляются в компьютере немного сложнее. Логические операции, которые использовались с int, для вещественных типов данных не подходят. Но это ограничение можно легко обойти, использовав объединения.

Объединения – это две или более переменных расположенных по одному адресу (они разделяют одну и ту же память). Объединения определяются с использованием ключевого слова union. Объединения не могут хранить одновременно несколько различных значений, они позволяют интерпретировать несколькими различными способами содержимое одной и той же области памяти.

**Постановка задачи.**

Разработать алгоритм и написать программу, которая позволяет:

ИДЗ) Реализовать циклический сдвиг всех битов числа на 3 позиции влево.

1) Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.

2) Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.

3) Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

4) Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок. (\*)

5) Реализовать возможность произвольного изменения любого бита в введенном числе (для всех типов данных) с использованием логических операций.

**Выполнение работы.**

**ИДЗ)**

Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов

\*/

/\*

Блок скриншотов работы программы

\*/

/\*

Блок таблицы с тестовыми данными

\*/

Условный пример представлен в приложении (данный пример необходимо удалить при написании отчёта). Пример – ориентир на содержание, а не на стиль оформления.

**Выводы.**

Приложение А

рабочий код

/\*int value = 124;

int order = sizeof(int) \* 8;

unsigned int mask = 1 << (order - 1);

int shiftVal = (value << 3) | (value >> (order - 3));

cout << "десятичное представление: " << value << endl;

cout << "двоичное представление: " << " ";

for (int i = 0; i < order; i++) {

if (value & (mask >> i)) {

cout << "1";

}

else {

cout << "0";

}

}

cout << endl;

cout << "после циклического сдвига на 3 позиции влево:" << endl;

cout << "десятичное представление: " << shiftVal << endl;

cout << "двоичное представление:" << " ";

for (int i = 0; i < order; i++) {

if (shiftVal & (mask >> i)) {

cout << "1";

}

else {

cout << "0";

}

}\*/

Условный пример содержания пункта «Выполеннеие работы»

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| Меню | |
| При запуске программы перед пользователем появляется окно с меню, где он может выбрать тип будущей вводимой последовательности | Меню:    Проверка на ввод символов, которые не входят в диапазон выбора: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Создание новой записи | |
| При вводе пользователем корректного значения пункта меню и выбора создания новой записи, пользователь может добавить данные о студенте, но только на русском языке. | Для начала идёт проверка на правильный ввод пользователем:    Как только все данные заданы, пользователь может еще раз просмотреть все данные о студенте и в случае, если видит ошибку, сразу же ее исправить без дополнительного вызова меню: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Внесение изменений в имеющуюся запись | |
| Если пользователь допустил ошибку в создании карточки студента или данные изменились, он может внести изменения по пунктам, которые были созданы ранее. Это второе меню, с которым встречается пользователь. | При необходимости внесения изменений в запись о студенте, перед пользователем появляется меню:    Как только пользователь вводит корректное значение (в данном случае от 1 до 10), его ответ обрабатывается и предлагается изменить существующие данные на новые. После пользователь возвращается (при необходимости) в главное меню. |

Продолжение Таблицы

|  |  |
| --- | --- |
| Информация о студентах в количестве | |
| Если пользователь хочет классифицировать студентов по полу, по группам, по номеру в списке, то он может вывести список о всех студентах | Пример, когда пользователь выбрал список данных по конкретной группе, ввел номер группы, а студентов в этой группе пока что не имеется:    Пример, когда пользователь выбрал студентов по их номеру в списке соответственно для двух групп: |

Продолжение Таблицы

|  |  |
| --- | --- |
| Рейтинг студентов | |
| Пользователь в главном меню может выбрать такую опцию как рейтинг студентов, чтобы наглядно увидеть список студентов с минимальными данными и рейтингу их оценок. | Выводится номер в списке по группам, ФИО, сама группа и рейтинг. Рейтинг складывается из всех оценок за сессию (диф зачеты экзамены):    В данном примере студентов в базе было занесено один, то есть, информация и средний бал вывелся только по одному человеку. |

Продолжение Таблицы

|  |  |
| --- | --- |
| Стипендия | |
| Стипендию пользователь может узнать в двух случаях: он может выбрать непосредственно пункт в меню, где ему выйдет список студентов, которые будут получать стипендию. Или пользователь может выбрать список с данными о студентах, которые могли бы получать стипендию, то есть они учатся не на очной форме обучения при оценках не ниже хорошо. | Пример вывода данных о студенте, который не будет получать стипендию при положительных оценках: |

Окончание Таблицы

|  |  |
| --- | --- |
| Очередь в общежитие | |
| Пользователь может узнать очередь в общежитие, занося данные о место происхождении студента и о его заработке. | Пример очереди из студентов, которые получают разных доход (в тыс. руб.), являются и иногородними, и местными:  ../Desktop/очередь.png |
| Полный список о студентах | |
| Если пользователь не собирается получать данные о студентах по конкретному критерию, он может вывести полный список студентов в консоли или же получить отдельный файл. | Вывод в консоли:  ../Desktop/Снимок%20экрана%202020-02-29%20в%2010.22.33.png  Вывод в файл:  ../Desktop/Снимок%20экрана%202020-02-29%20в%2010.22.07.png |