**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, определяемые пользователем. Структуры

|  |  |  |
| --- | --- | --- |
| Студент гр. 5372 |  | Листопадов И. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2025

**Цель работы.**

Изучение и организация структур; получение практических навыков работы со структурами; определение преимуществ и недостатков использования структур.

**Основные теоретические положения.**

**Встроенные (фундаментальные) типы**

* **Целочисленные**: char, short, int, long, long long
* **Вещественные**: float, double

**Целочисленные типы (**int, short, long, long long**)**

Представляют целые числа. Их размер и, следовательно, диапазон значений, **зависит** от компилятора и архитектуры процессора.

**Модификаторы знака:** Целочисленные типы могут быть дополнительно модифицированы для изменения их знаковости.

* signed (знаковый): может хранить положительные и отрицательные числа. Это значение по умолчанию для int, short, long, long long.
* unsigned (беззнаковый): может хранить только неотрицательные числа. Диапазон сдвигается в положительную сторону.
  + unsigned short: 0 ... 65 535
  + unsigned int: 0 ... 4 294 967 295
  + unsigned long long: 0 ... 2^64 - 1

Целые числа хранятся в памяти в виде двоичного кода. Для знаковых типов обычно используется дополнительный код,что позволяет одной схемой выполнять арифметические операции над числами любого знака.

**Типы с плавающей точкой (**float, double, long double**)**

Представляют вещественные (дробные) числа. Хранятся в памяти согласно стандарту IEEE 754.

Число представляется в виде: знак + экспонента + мантисса. Это позволяет оперировать очень большими и очень маленькими числами, но приводит к ошибкам округления. double является типом по умолчанию для операций с плавающей точкой в C++.

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

**Операции.**

Схема преобразования, используемая при выполнении арифметических операций, называется обычными арифметическими преобразованиями. Эта схема может быть описана следующими правилами:

1. Все данные типов char и short int преобразуются к типу int.
2. Если хотя бы один из операндов имеет тип double, то и другой операнд преобразуется к типу double (если он другого типа); результат вычисления имеет тип double.
3. Если хотя бы один из операндов имеет тип float, то и другой операнд преобразуется к типу float (если он другого типа); результат вычисления имеет тип float.
4. Если хотя бы один операнд имеет тип long, то и другой операнд преобразуется к типу long (если он другого типа); результат имеет тип long.
5. Если хотя бы один из операндов имеет тип unsigned, то и другой операнд преобразуется к типу unsigned (если его тип не unsigned); результат имеет тип unsigned.

Унарный плюс возвращает значение операнда без изменений.

Унарный минус (его иногда называют арифметическим отрицанием) меняет знак операнда на обратный.

Умножение**-** \* - бинарная операция.

Деление - **/** - бинарная операция.

Остаток от деления - **%** - бинарная операция. Операнды только целого типа. Результат операции целого типа.

Сложение (**+**) и вычитание (**-**) – бинарные операции. Операнды могут быть любых арифметических типов данных.

Логические операции используются при построении сложных логических выражений. В эту группу входят 3 операции:

·         **!**      - логическое отрицание (логическое НЕ);

·         **&& -** конъюнкция (логическое И);

·         **||**       - дизъюнкция (логическое ИЛИ).

Операции сдвига - << и >> - бинарные операции. Операнды целого типа. Результат также целого типа. Формат записи:

< Операнд 1 >   <<   < Операнд 2 > - сдвиг влево

< Операнд 1 >   >>   < Операнд 2 > **-**сдвиг вправо

Операции выполняют копирование битов двоичного представления первого операнда со сдвигом на количество разрядов, указанное во втором операнде, в соответствующем направлении.

Поразрядные логические операции

К этой группе операций относятся:

·         **~**    - побитовое отрицание (побитовое НЕ) - унарная операция;

·         **&**   - побитовая конъюнкция (побитовое И) - бинарная операция;

·         **|**     - побитовая дизъюнкция (побитовое ИЛИ) - бинарная операция;

·         **^**    - побитовое исключающее ИЛИ - бинарная операция.

?: - условная тернарная операция.

Библиотека <iostream> позволяет пользователю выводить на экран различные данные. Библиотека подключается в начале файла через #include, а ее название записывается в скобках <…>

Для вывода данных пользователь может использовать структуру потока

std::cout << команда << std::endl;

Но для более простого использования и повышения скорости написания кода, в начале файла можно обозначить пространство, с которым будет работать пользователь:

using namespace std;

теперь для вывода данных достаточно напечатать cout << команда << endl;

В ходе работы используются объединения и массивы.

Объединение представляет собой совокупность нескольких типов данных, которые хранятся в одной ячейке памяти, что позволяет производить некоторые операции, до этого нам не доступные.

Объединение union имеет такую структуру:

union {

int numOne;

float numTwo;

} myUnion;

Таким образом, пользователь может быстро переключаться между типами float и int.

**Циклы.**

Условные операторы:

if - выполнить блок кода, если условие истинно.

else - выполнить блок кода, если условие в if ложно.

else if - проверить дополнительное условие, если предыдущее условие ложно.

Циклы:

while - выполнять блок кода пока условие истинно. Проверка условия происходит перед каждой итерацией.

for - цикл с инициализацией переменной-счетчика, условием продолжения и выражением для изменения счетчика после каждой итерации.

Общие особенности:

* Условия указываются в круглых скобках
* Тело операторов обрамляется фигурными скобками
* Можно использовать операторы break и continue для управления циклом

Массив представляет собой череду ячеек, количество которых пользователь может установить сам, либо оно устанавливается автоматически, исходя из количества элементов массива

int A [2] – массив из двух элементов типа int.

Индексация начинается с нуля, поэтому, чтобы обратиться к первому элементу следует написать A [0], а A [1] для второго элемента соответственно.

Чтобы вывести элементы массива следует продумать два вложенных цикла for, первый из которых проходится по количеству массивов, а второй по длине разрядов числа, над которым совершаются логические операции.

Оператор switch() позволяет продумать несколько исходов работы функции.

Он принимает целое число, и на его основе выполняет определенный алгоритм.

int num;

cin >> num;

switch(num) {

case 1:

блок команд 1;

break;

case 2:

блок команд 2;

break;

**Постановка задачи.**

В ходе практической работы необходимо решить пять задач:

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

2) Необходимо 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) Необходимо 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) Необходимо 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.

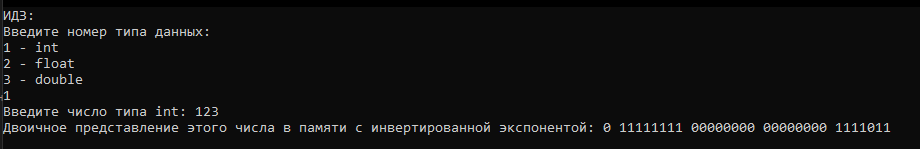
5) Необходимо реализовать возможность произвольного изменения любого бита в введенном числе (для всех типов данных) с использованием логических операций.

ИДЗ №29) Необходимо инвертировать все биты порядка, не трогая знаковый бит и мантиссу.

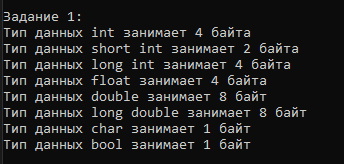
**Выполнение работы.**

Код программы представлен в приложении А.

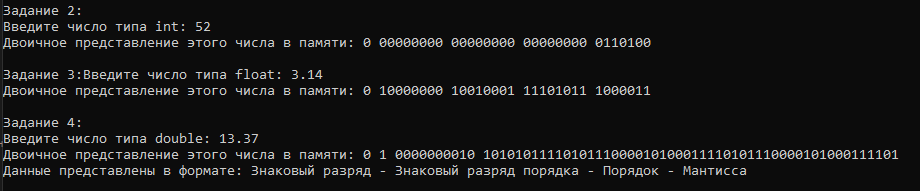
Выполнение ИДЗ на примере целого числа:



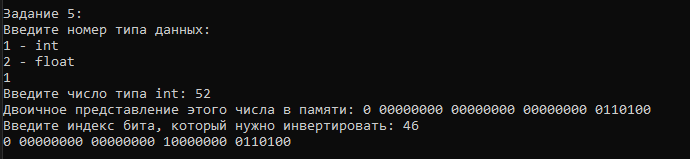
Вывод количества памяти, занимаемой разными типами данных:



Ввод с клавиатуры целого, вещественного чисел и числа типа double и вывод их двоичных представлений на экран:



Выбор типа данных, ввод числа и номера бита и выборочное изменения бита на противоположный:



**Выводы.**

В ходе работы я ознакомился с организацией типов данных в памяти компьютера, получил практические навыки работы с различными типами данных, их двоичными и десятичными представлениями.

**ПРИЛОЖЕНИЕ А**

**РАБОЧИЙ КОД**

#include <iostream>

using namespace std;

int main() {

setlocale(0, "");

//ИДЗ) Инверсия битов порядка. Инвертируйте все биты порядка, не трогая знаковый бит и мантиссу.

cout << "ИДЗ:" << endl;

int numOfTask0;

cout << "Введите номер типа данных:\n";

cout << "1 - int\n";

cout << "2 - float\n";

cout << "3 - double\n";

cin >> numOfTask0;

switch (numOfTask0) {

case 1: {

int zeroValue;

unsigned int zeroMask = 1 << sizeof(int) \* 8 - 1;

cout << "Введите число типа int: ";

cin >> zeroValue;

cout << "Двоичное представление этого числа в памяти с инвертированной экспонентой: ";

for (int i = 0; i < sizeof(int) \* 8; i++) {

if (i > 0 && i < 9) {

putchar(zeroValue & (zeroMask >> i) ? '0' : '1');

}

else {

putchar(zeroValue & (zeroMask >> i) ? '1' : '0');

}

if (i % 8 == 0) {

putchar(' ');

}

}

cout << endl;

}

break;

case 2: {

union {

int zeroToolF;

float fNumber0;

} intFloatUnion0;

unsigned int zeroMaskF = 1 << sizeof(float) \* 8 - 1;

cout << "Введите число типа float: ";

cin >> intFloatUnion0.fNumber0;

int zeroValueF = intFloatUnion0.zeroToolF;

cout << "Двоичное представление этого числа в памяти с инвертированной экспонентой: ";

for (int i = 0; i < sizeof(int) \* 8; i++) {

if (i > 0 && i < 9) {

putchar(zeroValueF & (zeroMaskF >> i) ? '0' : '1');

}

else {

putchar(zeroValueF & (zeroMaskF >> i) ? '1' : '0');

}

if (i % 8 == 0) {

putchar(' ');

}

}

cout << endl;

}

break;

case 3: {

union {

int zeroToolD[2];

double dNumber0;

} intDoubleUnion0;

cout << "\nВведите число типа double: ";

cin >> intDoubleUnion0.dNumber0;

cout << "Двоичное представление этого числа в памяти с инвертированной экспонентой: ";

for (int counter(1); counter > -1; counter--) {

unsigned int doubleMask = 1 << (sizeof(int) \* 8 - 1);

for (int i(0); i < sizeof(int) \* 8; i++) {

if (i > 0 && i < 12) {

putchar(intDoubleUnion0.zeroToolD[counter] & (doubleMask >> i) ? '0' : '1');

}

else {

putchar(intDoubleUnion0.zeroToolD[counter] & (doubleMask >> i) ? '1' : '0');

}

if ((i == 0 || i == 1 || i == 11) && counter == 1) {

putchar(' ');

}

}

}

}

break;

default: {

cout << "Неверно указан номер";

}

}

cout << endl;

//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.

cout << "\nЗадание 1:" << endl;

cout << "Тип данных int занимает " << sizeof(int) << " байта" << endl;

cout << "Тип данных short int занимает " << sizeof(short int) << " байта" << endl;

cout << "Тип данных long int занимает " << sizeof(long int) << " байта" << endl;

cout << "Тип данных float занимает " << sizeof(float) << " байта" << endl;

cout << "Тип данных double занимает " << sizeof(double) << " байт" << endl;

cout << "Тип данных long double занимает " << sizeof(long double) << " байт" << endl;

cout << "Тип данных char занимает " << sizeof(char) << " байт" << endl;

cout << "Тип данных bool занимает " << sizeof(bool) << " байт" << endl;

//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.

cout << "\nЗадание 2:" << endl;

int firstValue;

unsigned int intMask = 1 << sizeof(int) \* 8 - 1;

cout << "Введите число типа int: ";

cin >> firstValue;

cout << "Двоичное представление этого числа в памяти: ";

for (int i = 0; i < sizeof(int) \* 8; i++) {

putchar(firstValue & (intMask >> i) ? '1' : '0');

if (i % 8 == 0) {

putchar(' ');

}

}

cout << endl;

//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.

cout << "\nЗадание 3:";

union {

int firstTool;

float fNumber;

} intFloatUnion;

unsigned int floatMask = 1 << sizeof(float) \* 8 - 1;

cout << "Введите число типа float: ";

cin >> intFloatUnion.fNumber;

int secondValue = intFloatUnion.firstTool;

cout << "Двоичное представление этого числа в памяти: ";

for (int i = 0; i < sizeof(int) \* 8; i++) {

putchar(secondValue & (floatMask >> i) ? '1' : '0');

if (i % 8 == 0) {

putchar(' ');

}

}

cout << endl;

//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.

cout << "\nЗадание 4:";

union {

int secondTool[2];

double dNumber;

} intDoubleUnion;

cout << "\nВведите число типа double: ";

cin >> intDoubleUnion.dNumber;

cout << "Двоичное представление этого числа в памяти: ";

for (int counter(1); counter > -1; counter--) {

unsigned int doubleMask = 1 << (sizeof(int) \* 8 - 1);

for (int i(0); i < sizeof(int) \* 8; i++) {

putchar(intDoubleUnion.secondTool[counter] & (doubleMask >> i) ? '1' : '0');

if ((i == 0 || i == 1 || i == 11) && counter == 1) {

putchar(' ');

}

}

}

cout << "\nДанные представлены в формате: Знаковый разряд - Знаковый разряд порядка - Порядок - Мантисса" << endl;

//5) Реализовать возможность произвольного изменения любого бита в введенном чиcле с использованием логических операций.

cout << "\nЗадание 5:" << endl;

int numOfTask5;

cout << "Введите номер типа данных:\n";

cout << "1 - int\n";

cout << "2 - float\n";

cin >> numOfTask5;

switch (numOfTask5) {

case 1: {

int fourthValue, numBitI;

unsigned int intMask2 = 1 << sizeof(int) \* 8 - 1;

cout << "Введите число типа int: ";

cin >> fourthValue;

cout << "Двоичное представление этого числа в памяти: ";

for (int i = 0; i < sizeof(int) \* 8; i++) {

putchar(fourthValue & (intMask2 >> i) ? '1' : '0');

if (i % 8 == 0) {

putchar(' ');

}

}

cout << endl;

cout << "Введите индекс бита, который нужно инвертировать: ";

cin >> numBitI;

int newFourthValue;

newFourthValue = fourthValue ^ (1 << numBitI);

for (int i = 0; i < sizeof(int) \* 8; i++) {

putchar(newFourthValue & (intMask2 >> i) ? '1' : '0');

if (i % 8 == 0) {

putchar(' ');

}

}

cout << endl;

}

break;

case 2: {

int numBitF;

union {

int firstTool5;

float fNumber5;

} intFloatUnion;

unsigned int floatMask5 = 1 << sizeof(float) \* 8 - 1;

cout << "Введите число типа float: ";

cin >> intFloatUnion.fNumber5;

int fourthValue = intFloatUnion.firstTool5;

cout << "Двоичное представление этого числа в памяти: ";

for (int i = 0; i < sizeof(int) \* 8; i++) {

putchar(fourthValue & (floatMask5 >> i) ? '1' : '0');

if (i % 8 == 0) {

putchar(' ');

}

}

cout << endl;

cout << "Введите индекс бита, который нужно инвертировать: ";

cin >> numBitF;

int newFourthValue;

newFourthValue = fourthValue ^ (1 << numBitF);

for (int i = 0; i < sizeof(int) \* 8; i++) {

putchar(newFourthValue & (floatMask5 >> i) ? '1' : '0');

if (i % 8 == 0) {

putchar(' ');

}

}

cout << endl;

}

break;

default: {

cout << "Неверно указан номер";

}

return 0;

}

}