**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(ка) гр. |  | Цветов И.С. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2021

**Цель работы.**

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

**Основные теоретические положения.**

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.

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.

Логические операции:

В ходе работы используется побитовое логическое умножение &, которое возвращает произведение двоичного кода первого числа на второе

Логическое сложение | позволяет пользователю сложить двоичные коды двух чисел.

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 компьютере занимают идентификаторы int, short, long, float, double, char и bool.

2) Необходимо вывести на экран двоичное представление целого числа и выделить значащие разряды.

3) Необходимо вывести на экран двоичное представление вещественного числа и выделить значащие разряды.

4) Необходимо вывести на экран двоичное представление числа типа double и выделить значащие разряды.

5) Необходимо реализовать возможность произвольного изменения любого бита в введенном числе.

**Выполнение работы.**

Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов

Для удобного использования программы был использован оператор switch(), который позволяет пользователю выбрать задачу и типы данных, а исходя из его выбора выводится определенный блок кода.

Для решения первого задания используется команда sizeof(), которая возвращает пользователю объём зарезервированной памяти.

Пользователь может выводить данные поочерёдно или все типы разом:

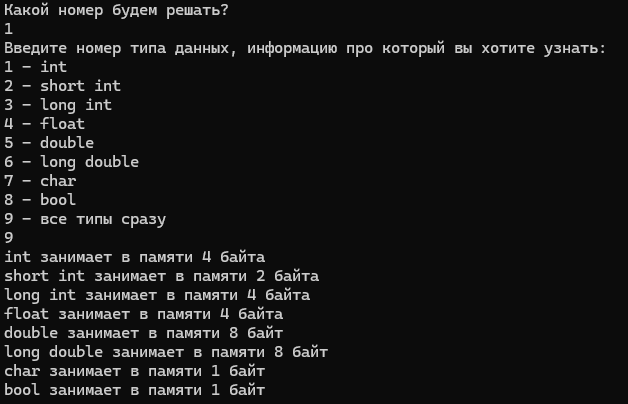


Рисунок 1. Пример работы кода для задания 1

Для решения второго задания через побитовый сдвиг единицы влево создается маска, через которую путем логического умножения (&) получается представить число, вводимое пользователем, в двоичном виде.

Важно отметить, что для удобства чтения цветом выделены знаковый разряд и значащие разряды (Рисунок 2). Библиотека <iostream> позволяет установить соответствующие цвета:

32- зеленый

33- желтый

34- синий

35- пурпурный

36- голубой

37- серый

Цвет текста устанавливается через поток вывода cout по схеме:

**“\033[**номер\_цвета**m**текст”, где номер\_цвета - кодовый номер цвета, который хочет установить пользователь.

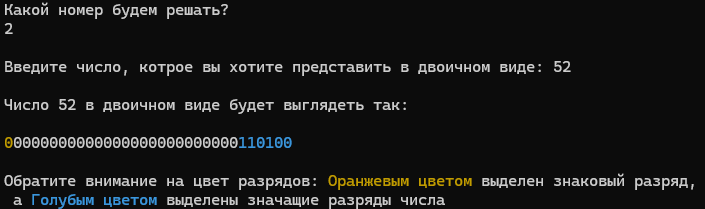


Рисунок 2. Результат работы кода для второго задания

Для решения номера три используется код, практически идентичный коду второго задания. Отличие заключается в объединении union, так как у числа float и целочисленной маски разные типы данных, которые нельзя сравнивать. Union позволяет обращаться к одной и той же ячейке памяти через разные типы данных (в нашем случае float и int). Аналогично пункту два мы используем маску unsigned mask, через которую сдвигом битов получаем итоговый результат. При выводе цветом выделены знак числа, экспонента и мантисса (Рисунок 3).

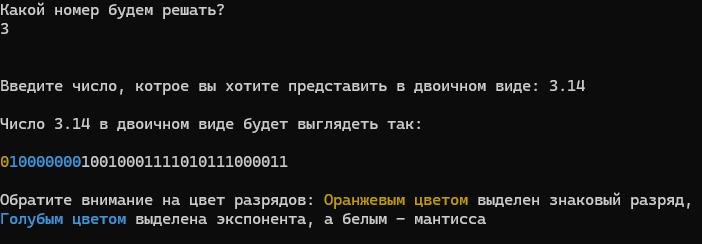


Рисунок 3. Результат работы кода для третьего задания

При решении четвертого пункта лабораторной работы потребовалось использовать массив из двух типов int, так как идентификатор double занимает в памяти 8 байт, а int всего 4. При выводе цветом выделены знак числа, знак порядка, знак мантиссы, порядок и мантисса (Рисунок 4).

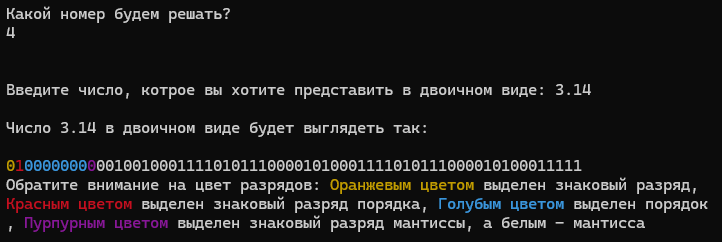


Рисунок 4. Результат работы кода для четвертого задания

Для решения пятого задания понадобилось использование логической операции исключающее или. Пользователь вводит число, над которым хочет провести некие операции, затем компилятор принимает индекс разряда, с которым работает код (нумерация с нуля), затем на основе введенного разряда создается специальная маска. Исключающее или позволяет заменить нужный нам бит числа на противоположный. (Рисунок 5)

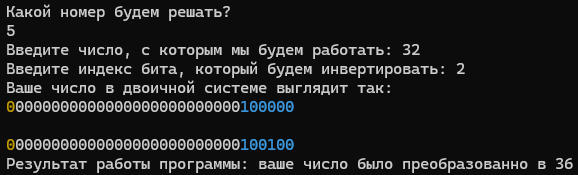


Рисунок 5. Результат работы кода для пятого задания

**Выводы.**

При работе с кодом крайне важно обращать внимание на типы данных и их совместимость, так как операции над разными типами могут привести к неприятным последствиям и ошибкам.

Для оптимизации кода не стоит забывать о типах данных и объёме памяти, который они резервируют. Не стоит перегружать код тяжеловесными операциями, если их можно избежать.

Приложение А

рабочий код

#include <iostream>

#include <locale.h>

using namespace std;

int main() {

setlocale(LC\_ALL, "Russian");

int numOfTask;

cout << "Какой номер будем решать?\n";

cin >> numOfTask;

switch (numOfTask)

{

case 1:

/\*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.\*/

{

int nameOfData;

cout << "Введите номер типа данных, информацию про который вы хотите узнать:\n";

cout << "1 - int\n";

cout << "2 - short int \n";

cout << "3 - long int\n";

cout << "4 - float\n";

cout << "5 - double\n";

cout << "6 - long double\n";

cout << "7 - char\n";

cout << "8 - bool\n";

cout << "9 - все типы сразу\n";

cin >> nameOfData;

switch (nameOfData)

{

case 1: cout << "int занимает в памяти " << sizeof(int) << " байта";

break;

case 2: cout << "short int занимает в памяти " << sizeof(short int) << " байта";

break;

case 3: cout << "long int занимает в памяти " << sizeof(long int) << " байта";

break;

case 4: cout << "float занимает в памяти " << sizeof(float) << " байта";

break;

case 5: cout << "double занимает в памяти " << sizeof(double) << " байт";

break;

case 6: cout << "long double занимает в памяти " << sizeof(long double) << " байт";

break;

case 7: cout << "char занимает в памяти " << sizeof(char) << " байт";

break;

case 8: cout << "bool занимает в памяти " << sizeof(bool) << " байт";

break;

case 9: cout << "int занимает в памяти " << sizeof(int) << " байта\n";

cout << "short int занимает в памяти " << sizeof(short int) << " байта\n";

cout << "long int занимает в памяти " << sizeof(long int) << " байта\n";

cout << "float занимает в памяти " << sizeof(float) << " байта\n";

cout << "double занимает в памяти " << sizeof(double) << " байт\n";

cout << "long double занимает в памяти " << sizeof(long double) << " байт\n";

cout << "char занимает в памяти " << sizeof(char) << " байт\n";

cout << "bool занимает в памяти " << sizeof(bool) << " байт\n";

break;

default:

cout << "К сожалению, вы указали неправильное число\n";

}

}

cout << endl;

break;

case 2:

/\*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.\*/

{

int ourValue;

int order = sizeof(int) \* 8;

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

cout << "\nВведите число, котрое вы хотите представить в двоичном виде: ";

cin >> ourValue;

cout << "\nЧисло " << ourValue << " в двоичном виде будет выглядеть так:\n\n";

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

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

if (i == 0) {

cout << "\033[33m1\033[0m";

}

else {

cout << "\033[36m1";

}

}

else {

if (i == 0) {

cout << "\033[33m0\033[0m";

}

else {

cout << "0";

}

}

}

cout << "\n\033[0m";

cout << "\nОбратите внимание на цвет разрядов: \033[33mОранжевым цветом\033[0m выделен знаковый разряд, а \033[36mГолубым цветом\033[0m выделены значащие разряды числа";

}

break;

case 3:

/\*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.\*/

{

union {

int tool;

float numbF;

} intFloatUnion;

cout << "\n\nВведите число, котрое вы хотите представить в двоичном виде: ";

cin >> intFloatUnion.numbF;

int ourValue = intFloatUnion.tool;

cout << "\nЧисло " << intFloatUnion.numbF << " в двоичном виде будет выглядеть так:\n\n";

int order = sizeof(int) \* 8;

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

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

{

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

if (i == 0) {

cout << "\033[33m1\033[0m";

}

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

cout << "\033[36m1\033[0m";

}

else {

cout << "\033[0m1";

}

}

else {

if (i == 0) {

cout << "\033[33m0\033[0m";

}

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

cout << "\033[36m0\033[0m";

}

else {

cout << "\033[0m0";

}

}

}

cout << "\n\033[0m";

cout << "\nОбратите внимание на цвет разрядов: \033[33mОранжевым цветом\033[0m выделен знаковый разряд, \033[36mГолубым цветом\033[0m выделена экспонента, а белым - мантисса\n";

}

cout << "\n";

break;

case 4:

/\*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. (\*)\*/

{

union {

int tool[2];

double num;

} intDoubleUnion;

int numOfSigns = 0; //счётчик разрядов для выделения важных байтов цветом

cout << "\n\nВведите число, котрое вы хотите представить в двоичном виде: ";

cin >> intDoubleUnion.num;

cout << "\nЧисло " << intDoubleUnion.num << " в двоичном виде будет выглядеть так:\n\n";

int order = sizeof(int) \* 8;

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

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

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

if (intDoubleUnion.tool[counter] & (mask >> i)) {

if (numOfSigns == 0) {

cout << "\033[33m1\033[0m";

numOfSigns += 1;

}

else if (numOfSigns == 9) {

cout << "\033[35m1\033[0m";

numOfSigns += 1;

}

else if (numOfSigns == 1) {

cout << "\033[31m1\033[0m";

numOfSigns += 1;

}

else if (numOfSigns > 1 && numOfSigns < 9) {

cout << "\033[36m1\033[0m";

numOfSigns += 1;

}

else if (numOfSigns > 9) {

cout << "\033[0m1\033[0m";

numOfSigns += 1;

}

}

else {

if (numOfSigns == 0) {

cout << "\033[33m0\033[0m";

numOfSigns += 1;

}

else if (numOfSigns == 1) {

cout << "\033[31m0\033[0m";

numOfSigns += 1;

}

else if (numOfSigns == 9) {

cout << "\033[35m0\033[0m";

numOfSigns += 1;

}

else if (numOfSigns > 1 && numOfSigns < 9) {

cout << "\033[36m0\033[0m";

numOfSigns += 1;

}

else if (numOfSigns > 9) {

cout << "0";

numOfSigns += 1;

}

}

}

}

cout << "\nОбратите внимание на цвет разрядов: \033[33mОранжевым цветом\033[0m выделен знаковый разряд, \033[31mКрасным цветом\033[0m выделен знаковый разряд порядка, \033[36mГолубым цветом\033[0m выделен порядок, \033[35mПурпурным цветом\033[0m выделен знаковый разряд мантиссы, а белым - мантисса\n";

}

break;

case 5:

int num;

int numOfBit;

cout << "Введите число, с которым мы будем работать: ";

cin >> num;

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

cin >> numOfBit;

cout << "Ваше число в двоичной системе выглядит так:\n";

int order = sizeof(int) \* 8;

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

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

if (num & (zeroMask >> i)) {

if (i == 0) {

cout << "\033[33m1\033[0m";

}

else {

cout << "\033[36m1";

}

}

else {

if (i == 0) {

cout << "\033[33m0\033[0m";

}

else {

cout << "0";

}

}

}

cout << "\033[0m";

int mask = 1 << numOfBit;

int newNum = num ^ mask;

int newOrder = sizeof(int) \* 8;

unsigned int newMask = 1 << (newOrder - 1);

cout << "\n\n";

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

if (newNum & (newMask >> i)) {

if (i == 0) {

cout << "\033[33m1\033[0m";

}

else {

cout << "\033[36m1";

}

}

else {

if (i == 0) {

cout << "\033[33m0\033[0m";

}

else {

cout << "0";

}

}

}

cout << "\n\033[0mРезультат работы программы: ваше число было преобразованно в " << newNum;

break;

}

}