ФЕДЕРАЛЬНОЕ Государственное АВТОНОМНОЕ образовательное учреждение

высшего образования

**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

Кафедра математического и программного обеспечения информационных систем

**Отчет по лабораторной работе №3**

**по дисциплине: «Структуры и алгоритмы компьютерной обработки данных»**

**Тема работы «Сравнительный анализ методов сортировки»**

студента очного отделения

2 курса 12001801 группы

Скрипниченко Илья Игоревич

Проверил(а):

Курлов Василий Васильевич

Белгород 2019

**Лабораторная работа №3**

**Цель работы:** Изучение базовых типов данных языка C++ как структур данных (СД).

Вариант 24.

1. Для типов данных (*unsigned char, float, {winter,spring,summer,autumn}*) определить:

1. Абстрактный уровень представления СД:

1.1. Характер организованности и изменчивости.

1.2. Набор допустимых операций.

2. Физический уровень представления СД:

2.1. Схему хранения.

2.2. Объём памяти, занимаемый экземпляром СД.

2.3. Формат внутреннего представления СД и способ его интерпретации.

2.4. Характеристику допустимых значений.

2.5. Тип доступа к элементам.

3. Логический уровень представления СД.

3.1. Способ описания СД и экземпляра СД на языке программирования.

2. Для заданных типов данных определить набор значений, необходимый для изучения физического уровня представления СД.

3. Преобразовать значения в двоичный код.

4. Преобразовать двоичный код в значение.

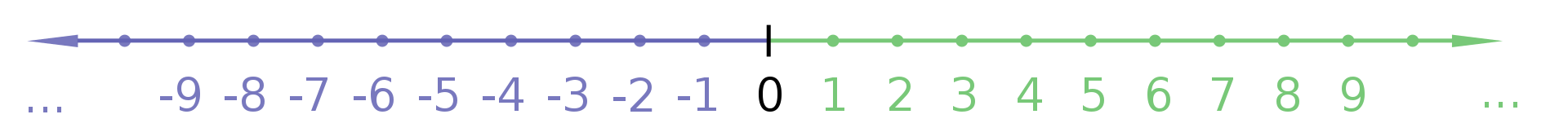
5. Разработать и отладить программу, выдающую двоичное представление значений заданных СД.В программе использовать функции BinOut и BinIn.

**Характеристика каждого заданного типа данных как СД в соответствии с п.1 задания.**

**unsigned char**

Является целочисленной переменной в беззнаковой форме.

Целые числа – это множества натуральных чисел, получаемые добавлением нуля и отрицательных чисел. Целые числа расположены на интервале от −∞ до +∞.



Целые числа на числовой прямой

Для целочисленной переменной, также как и для целого числа, определены основные арифметические операции: сложение, вычитание, умножение, деление; операции отношения =, О, <, >, >=, <= и стандартные функции.

Любая информация в компьютере, в том числе СД, хранится в виде последовательности бит (минимальная единица) в оперативной памяти компьютера. Бит в свою очередь может принимать два значения: 0 или 1. Таким образом, определить сколько различных значений может принимать СД можно по формуле *2N*, где N – количество бит, отведенных для хранения СД.

Доступ к хранимой в оперативной памяти информации производится побайтно, а саму оперативную память можно рассматривать как конечную последовательность байт. Для того чтобы получить доступ к конкретной ячейке в памяти, следует обратиться к физическому адресу этой ячейки, хранящей определенную информацию, который представляет собой конкретный номер байта из всей последовательности.

Для целочисленного беззнакового типа unsigned char выделяется 8 битов (1 байт) памяти. Основой представления этого типа является запись целого числа в двоичной система счисления, где каждый бит соответствует двоичной цифре. В отличии от знакового типа, старший бит не отводится под знак. Следовательно, этот тип может принимать 28 = 256 значений, то есть диапазон значений от 0 до 255.

Таким образом, можно говорить о том, что в языке программирования C++ ключевые слова unsigned и signed указывают, как интерпретируется нулевой бит объявляемой переменной, то есть если указано ключевое слово unsigned, то нулевой бит интерпретируется как часть числа, в противном случае нулевой бит интерпретируется как знаковый. В случае отсутствия ключевого слова unsigned целая переменная считается знаковой.

Следует также отметить, что модификатор-типа char используется для представления символа (из массива представление символов) или для объявления строковых литералов. Значением объекта типа char является код (размером 1 байт), соответствующий представляемому символу. Для представления символов русского алфавита, модификатор типа идентификатора данных имеет вид unsigned char, так как коды русских букв превышают величину 12.

На логическом уровне структура данных типа unsigned char может быть объявлена следующим образом:

*unsigned char* **ch**;

Для доступа к целочисленному значению на логическом уровне достаточно указать имя целочисленной переменной.

**float**

Является переменной, представляющей число с плавающей точкой (вещественное число).

Вещественное число – это [математический объект](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82), возникший из потребности [измерения](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%BC%D0%B5%D1%80%D0%B5%D0%BD%D0%B8%D0%B5) [геометрических](https://ru.wikipedia.org/wiki/%D0%93%D0%B5%D0%BE%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%8F) и [физических](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D0%BA%D0%B0) величин окружающего мира, а также проведения таких вычислительных операций, как [извлечение корня](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%B5%D0%BD%D1%8C_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), вычисление [логарифмов](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%B3%D0%B0%D1%80%D0%B8%D1%84%D0%BC), решение [алгебраических уравнений](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5), исследование поведения функций.

Если [натуральные числа](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%82%D1%83%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) возникли в процессе счёта, [рациональные](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) — из потребности оперировать частями целого, то вещественные числа предназначены для измерения непрерывных величин. Таким образом, расширение запаса рассматриваемых чисел привело к множеству вещественных чисел, которое, помимо [чисел рациональных,](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE) включает элементы, называемые [иррациональными числами](https://ru.wikipedia.org/wiki/%D0%98%D1%80%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%87%D0%B8%D1%81%D0%BB%D0%BE).

Вещественные числа обычно представляются в виде чисел с плавающей запятой. Числа с плавающей запятой — один из возможных способов представления действительных чисел, который является компромиссом между точностью и диапазоном принимаемых значений, его можно считать аналогом экспоненциальной записи чисел, но только в памяти компьютера.

Число с плавающей запятой состоит из набора отдельных двоичных разрядов, условно разделенных на так называемые знак, порядок и мантиссу. В наиболее распространённом формате (стандарт IEEE 754) число с плавающей запятой представляется в виде набора битов, часть из которых кодирует собой мантиссу числа, другая часть — показатель степени, и ещё один бит используется для указания знака числа (0 — если число положительное, 1 — если число отрицательное). При этом порядок записывается как целое число в [коде со сдвигом](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D1%86%D0%B5%D0%BB%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB:_%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B9_%D0%BA%D0%BE%D0%B4,_%D0%BA%D0%BE%D0%B4_%D1%81%D0%BE_%D1%81%D0%B4%D0%B2%D0%B8%D0%B3%D0%BE%D0%BC,_%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), а мантисса — в [нормализованном виде](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB#.D0.9D.D0.BE.D1.80.D0.BC.D0.B0.D0.BB.D1.8C.D0.BD.D0.B0.D1.8F_.D0.B8_.D0.BD.D0.BE.D1.80.D0.BC.D0.B0.D0.BB.D0.B8.D0.B7.D0.BE.D0.B2.D0.B0.D0.BD.D0.BD.D0.B0.D1.8F_.D1.84.D0.BE.D1.80.D0.BC.D0.B0), своей дробной частью в двоичной системе счисления.

Нормальной формой числа с плавающей запятой называется такая форма, в которой мантисса (без учёта знака) в десятичной системе находится на полуинтервале [0; 1).  Такая форма записи имеет недостаток: некоторые числа записываются неоднозначно (например 0. 001 можно записать в 4 формах – 0.0001 \* 100, 0.001 \* 10-1, 0,01 \* 10-2, 0,1 \* 10-3), поэтому распространена также другая форма записи — нормализованная, в которой мантисса десятичного числа принимает значения [1; 10), а мантисса двоичного числа принимает значения [1; 2). То есть в мантиссе слева от запятой до применения порядка находится ровно один знак. В такой форме любое число (кроме 0) записывается единственным образом. Ноль же представить таким образом невозможно, поэтому стандарт предусматривает специальную последовательность битов для задания числа 0 (а заодно и некоторых других [полезных чисел](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9F%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2%D0%B5%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D1%85_%D1%87%D0%B8%D1%81%D0%B5%D0%BB#.D0.9E.D1.81.D0.BE.D0.B1.D1.8B.D0.B5_.D0.B7.D0.BD.D0.B0.D1.87.D0.B5.D0.BD.D0.B8.D1.8F_.D1.87.D0.B8.D1.81.D0.B5.D0.BB_.D1.81_.D0.BF.D0.BB.D0.B0.D0.B2.D0.B0.D1.8E.D1.89.D0.B5.D0.B9_.D1.82.D0.BE.D1.87.D0.BA.D0.BE.D0.B9), таких как −∞ и +∞). Так как старший двоичный разряд (целая часть) мантиссы вещественного числа в нормализованном виде всегда равен 1, то его можно не записывать, сэкономив таким образом один бит, что и используется в стандарте IEEE 754.

Тип float в соответствии со стандартом IEEE 754 записывается в 32-битном формате (формат с одинарной точностью). Число в данном формате занимает 4 байта (32 бита), из которых старший бит кодирует знак числа, следующие за ним 8 бит – порядок числа и оставшиеся 23 бита – мантиссу.

|  |  |  |
| --- | --- | --- |
| 1 | 8 | 23 |
| s | e (порядок) | m (мантисса) |

Следует отметить, что старший бит мантиссы в данном случае всегда равен 1, поэтому диапазон значений для типа с плавающей точкой float от -3.4 \* 1038 до 3.4 \* 1038.

Формула для вычисления значения числа из 32-битного представления:

v=(-1)s×2e-127×1.m, если 0<e<255

v=(-1)s×2126×0.m, если e=0 и m<>0

v=(-1)s×0., если e=0 и m=0

v=(-1)s×Inf, если e=255 и m=0

v=NaN, если e=255 и m<>0

Точность представления числа с плавающей точкой определяет количество значащих цифр. Точность в свою очередь зависит от размера типа данных и присваиваемого значения. Для float точность от 6 до 9 значащих цифр.

Для доступа к числу с плавающей точкой на логическом уровне достаточно указать имя переменной типа float.

На логическом уровне СД типа float может быть объявлена следующим образом:

float **v**;

**{winter, spring, summer, autumn}**

Перечисляемый тип состоит из конечного числа упорядоченных именованных значений. В классическом варианте, свойственном, например, языкам линии Паскаль, определение типа состоит из перечисления имен значений (поэтому справедливо называть такой тип перечисляемым), эти имена в дальнейшем играют роль имен литеральных констант этого типа и должны отличаться от литерального изображения констант любого другого типа. Поскольку значения типа задаются путем перечисления, каждому значению можно однозначно сопоставить натуральное число от 1 до n, где n – число значений перечисляемого типа.

Обычно для любого перечисляемого типа предопределяются операции получения значения по его номеру и получения номера по значению. Кроме того, для перечисляемого типа предопределяются операции сравнения и получения следующего и предыдущего значения. По причине однозначного сопоставления значению перечисляемого типа натурального числа, возможно неявное преобразование этих значений к значению любого числового типа данных.

В языках линии Си под тем же термином "перечисляемый тип" понимается нечто другое, поскольку при определении такого типа можно явно сопоставить имени значения некоторое целое (не обязательно положительное) число. При отсутствии явного задания целого первому элементу перечисляемого типа неявно соответствует 0, а каждому следующему – целое значение, на единицу большее целого значения предыдущего элемента.

При этом:

a) использование имени перечисляемого типа для объявления переменной эквивалентно использованию типа int, и такая переменная может содержать любое целое значение;

b) имена значений перечисляемого типа на самом деле понимаются как имена целых констант, и к этим значениям применимы все операции над

целыми числами, даже если они выводят за пределы множества целых значений элементов перечисляемого типа.

На логическом уровне перечисляемый тип (или просто перечисление) определяются с помощью ключевого слова enum, которое указывает на начало перечисляемого типа. Стандартный вид определения перечислений следующий:

enum ярлык {список перечислений} список переменных;

Как имя перечисления – ярлык, так и список переменных необязательны, но один из них должен присутствовать. Список перечислений – это разделенный запятыми список идентификаторов (допустимых значений перечисления). Имя перечисления используется для объявления переменных данного типа.

Значением перечисления на физическом уровне является целое число – порядковый номер идентификатора значения логического уровня в списке. Нумерация начинается с 0.

В С++ имеется возможность определить значения одного или нескольких символов, используя инициализатор. Это делается путем помещения за символом знака равенства и целочисленного значения. При использовании инициализатора, символы, следующие за инициализационным значением, получают значение большее, чем указанное перед этим.

Стандарт C++ 11 не говорит, какой размер в памяти компьютера должен занимать переменная перечисляемого типа; только то, что он должен быть достаточно большим, чтобы соответствовать любому перечислителю.

В данном случае, оператор sizeof() возвращает длину в размере 4-х байт.

**Набор значений заданных типов, порядок их преобразования в двоичное представление, двоичное представление значений.**

unsigned char p = '!';

float v = 68.5;

enum seasons { winter = 1, spring = 10, summer = 20, autumn = 30 };

seasons snow = winter;

seasons snow\_melting = spring;

seasons heat = summer;

seasons rain = autumn;

1. **unsigned char**

По таблице ASKII находим значение символа ‘!’ в десятичной форме. Оно будет равняться 33-м.

Переводим 33 из десятичной формы в двоичную:

33/2 = 16 (остаток **1**), 16/2 = 8 (остаток **0**), 8/2 = 4 (остаток **0**), 4/2 = 2 (остаток **0**), 2/2 = 1 (остаток **0**), **1**.

3310 = 1000012

Т.к. переменная типа unsigned char занимает в памяти компьютера 1 байт, следовательно ее двоичная запись будет равна **00100001**.

2. **float**

Находим двоичную форму вещественного числа 68.5:

68/2 = 34 (остаток **0**), 34/2 = 17 (остаток **0**), 17/2 = 8 (остаток **1**), 8/2 = 4 (остаток **0**), 4/2 = 2 (остаток **0**), 2/2 = 1 (остаток **0**), **1**.

0,5 \* 2 = **1**.0

68,510 = 1000100.12

Приводим двоичную форму записи вещественного числа к виду 1. \_\_\_ \* 2n

1000100.12 = 1.00010012 \* 26

Т.к. переменная типа float занимает 4 байта, воспользуемся формулой для вычисления значения из 32 битной формулы числа:

*v=(-1)s×2e-127×1.m, если 0<e<255.*

Т.к. число положительное, то s = 0.

E = 6 + 127 = 133;

133/2 = 66 (ост. **1**), 66/2 = 33 (ост. **0**), 33/2 = 16 (ост. **1**), 16/2 = 8 (ост. **0**), 8/2 = 4 (ост. **0**), 4/2 = 2 (ост. **0**), 2/2 = 1 (ост. **0**), **1**.

e (8 бит) = 13310 = 100001012

m (23 бит) = 000100100000000000000002

Т.о., вещественное число 68.5 имеет физическое представление **0 10000101 00010010000000000000000**.

3. **перечисляемый тип**

Значения переменных перечисляемого типа занимают 4 байта в памяти компьютера и их двоичное представление получается тем же способом, что и представление целочисленных переменных.

110 = 12

1010 = 10102

2010 = 101002

3010 = 111102

*Seasons snow = winter* – **00000000000000000000000000000001**

*Seasons snow\_melt = spring* – **00000000000000000000000000001010**

*Seasons heat = summer* – **00000000000000000000000000010100**

*Seasons rain = autumn* – **000000000000000000000000000111**

**Набор двоичных векторов, порядок их преобразования в значения заданных типов, значения заданных типов.**

1. **unsigned char**

Переводим двоичное представление 00100001 (можем отбросить незначимые нули):

1000012 = 1 \* 20 + 0 \* 21 + 0 \* 22 + 0 \* 23 + 0 \* 24 + 1 \* 25 = 1 + 32 = 33.

По таблице ASKII находим соответствующий десятичной кодировке элемент – ‘**!**’.

2. **float**

Переведем двоичное представление 01000010100010010000000000000000 по той же формуле 32-битного представления:

s = 0

e = 100001012 = 1 + 4 + 128 = 13310

m = 00010010000000000000000

*v=(-1)s×2e-127×1.m, если 0<e<255.*

v = (-1)0 \* 2133-127 \* 1.00010010000000000000000 = 1.0001001 \* 26 = 1000100.12 = 1 \* 2-1 + 0 \* 20 + 0 \* 21 + 1 \* 22 + 0 \* 23 + 0 \* 24+ 0 \* 25 + 1 \* 26 = 0,5 + 4 + 64 = **68.510**

3. **перечисляемый тип**

Переведем двоичные значения в десятичную форму:

000000000000000000000000000000012 = 110

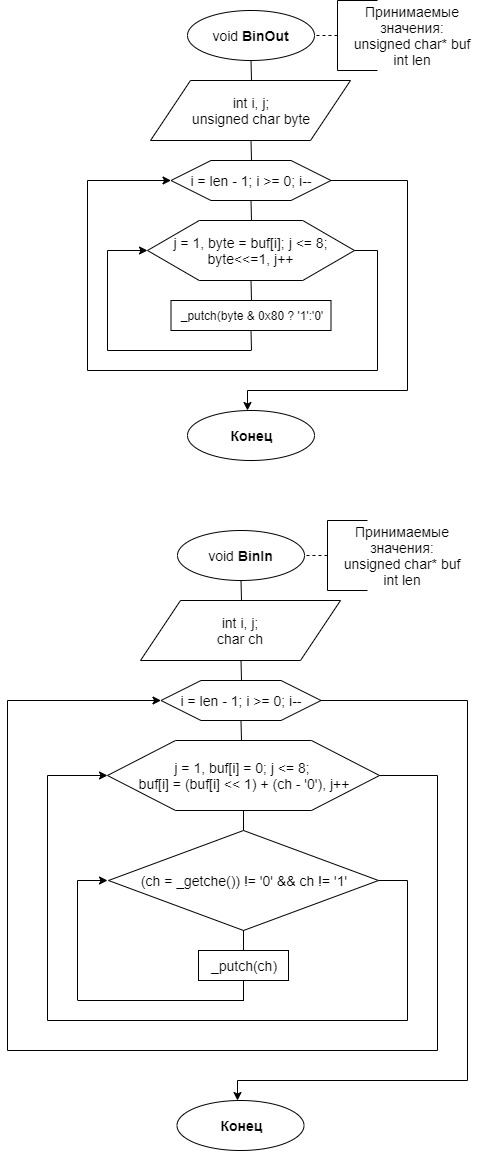
000000000000000000000000000010102 = 1010

000000000000000000000000000101002 = 2010

000000000000000000000000000101002 = 3010

**Спецификация алгоритма, текст программы, результаты работы про-граммы, выводы.**

**Блок- схема**

****

**Листинг**

#include "pch.h"

#include <iostream>

#include <stdio.h>

#include <conio.h>

void BinOut(unsigned char\*, int);

void BinIn(unsigned char\*, int);

int main()

{

using namespace std;

setlocale(0, "");

unsigned char n;

cout << "Введите значение целочисленной переменной типа unsigned char: ";

cin >> n;

cout << "Ее представление в двоичном коде будет следующим: \n";

BinOut((unsigned char\*)&n, sizeof(n));

cout << "Введите представление целочисленной переменной в двоичном коде: \n";

BinIn((unsigned char\*)&n, sizeof(n));

cout << "Ее значение: " << n << endl;

float v;

cout << "Введите значение переменной с плавающей точкой типа float: ";

cin >> v;

cout << "Ее представление в двоичном коде будет следующим: \n";

BinOut((unsigned char\*)&v, sizeof(v));

cout << "Введите представление переменной с плавающей точкой в двоичном коде: \n";

BinIn((unsigned char\*)&v, sizeof(v));

cout << "Ее значение: " << v << endl << endl << endl;

enum seasons { winter = 1, spring = 10, summer = 20, autumn = 30 };

seasons snow = winter;

seasons snow\_melting = spring;

seasons heat = summer;

seasons rain = autumn;

cout << "Представление символьной константы winter в двоичном коде: \n";

BinOut((unsigned char\*)&snow, sizeof(snow));

cout << "Представление символьной константы spring в двоичном коде: \n";

BinOut((unsigned char\*)&snow\_melting, sizeof(snow\_melting));

cout << "Представление символьной константы summer в двоичном коде: \n";

BinOut((unsigned char\*)&heat, sizeof(heat));

cout << "Представление символьной константы autumn в двоичном коде: \n";

BinOut((unsigned char\*)&rain, sizeof(rain));

cout << "Введите представление символьной константы winter в двоичном коде: \n";

BinIn((unsigned char\*)&snow, sizeof(snow));

cout << "Ее значение: " << snow << endl;

cout << "Введите представление символьной константы spring в двоичном коде: \n";

BinIn((unsigned char\*)&snow\_melting, sizeof(snow\_melting));

cout << "Ее значение: " << snow\_melting << endl;

cout << "Введите представление символьной константы summer в двоичном коде: \n";

BinIn((unsigned char\*)&heat, sizeof(heat));

cout << "Ее значение: " << heat << endl;

cout << "Введите представление символьной константы autumn в двоичном коде: \n";

BinIn((unsigned char\*)&rain, sizeof(rain));

cout << "Ее значение: " << rain << endl;

return 0;

}

void BinOut(unsigned char\* buf, int len)

{

int i, j;

unsigned char byte;

for (i = len - 1; i >= 0; i--)

{

for (j = 1, byte = buf[i]; j <= 8; byte<<=1, j++)

{

\_putch(byte & 0x80 ? '1':'0');

}

}

printf("\n");

}

void BinIn(unsigned char\* buf, int len)

{

int i, j;

char ch;

for (i = len - 1; i >= 0; i--)

{

for (j = 1, buf[i] = 0; j <= 8; buf[i] = (buf[i] << 1) + (ch - '0'), j++)

{

while ((ch = \_getche()) != '0' && ch != '1')

\_putch(ch);

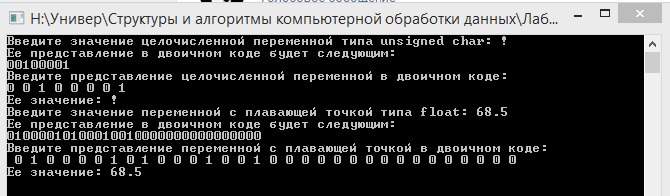
}

}

printf("\n");

}

**Скриншоты работы программы**



Скриншот 1. Работа программы с переменными типа unsigned char и float



Скриншот 2. Работа программы с перечисляемым типом