МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Лабораторная работа №1**по дисциплине: Алгоритмы и структуры данных тема:  
 «Встроенные структуры данных (Pascal/С)»

Выполнил: ст. группы ВТ-201 Билык Дмитрий Анатольевич

Проверил:

Синюк Василий Григорьевич

Белгород 2021 г.

**Встроенные структуры данных (Pascal/С)**

**Вариант 1**

**Цель работы**: изучение базовых типов данных языка Pascal/C как структур данных (СД).

**Задание**

1. Для типов данных (см.:Варианты заданий втабл.1,2) определить:
   1. Абстрактный уровень представления СД:
      1. Характер организованности и изменчивости.
      2. Набор допустимых операций.
   2. Физический уровень представления СД:
      1. Схему хранения.
      2. Объем памяти, занимаемый экземпляром СД.
      3. Формат внутреннего представления СД и способ его интерпретации.
      4. Характеристику допустимых значений.
      5. Тип доступа к элементам.
   3. Логический уровень представления СД.  
      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 процедуры *PrintByte* и *PrintVar*.

Спецификация процедуры *PrintByte*:

1. Заголовок: procedure *PrintByte(a:byte)/void PrintByte(unsigned char a)*.
2. Назначение: выводит на экран монитора двоичное представление переменной *a* типа *byte/unsigned char*.
3. Входные параметры: *a*.
4. Выходные параметры: нет.

Рекомендации: использовать побитовые операции сдвига и логического умножения.

Спецификация процедуры *PrintVar*:

1. Заголовок: procedure *PrintVar(var a; size:word)/ void PrintVar(void a, unsigned int size)*.
2. Назначение: выводит на экран монитора двоичное представление переменной *a* произвольного типа размером *size* байт.
3. Входные параметры: *a* – переменная произвольного типа, значение которой выводится на экран в двоичном представлении (нетипизованный параметр);  
   *size* – объем памяти (в байтах) занимаемый переменной *a*.
4. Выходные параметры: нет.

Рекомендации: нетипизованную переменную *a* привести к типу «массив байт», значение каждого элемента которого выводить на экран в двоичном представлении процедурой *PrintByte*.

1. Обработать программой значения, полученные в результате выполнения пункта 3 задания. Сделать выводы.
2. Разработать и отладить программу, определяющую значение переменной по ее двоичному представлению по следующему алгоритму:

1. Ввести двоичный код в переменную S строкового типа.

2. Преобразовать Sв вектор B типа «массив байт».

3. Привести B к заданному типу. Вывести значение.

4. Конец.

1. Обработать программой значения, полученные в результате выполнения пункта 4 задания. Сделать выводы.

**Задание 1:**

**Тип1: int**

* 1. Абстрактный уровень представления СД:
     1. Характер организованности и изменчивости: простейший, статический;
     2. Набор допустимых операций: инициализация, доступ, присваивание, сравнения, побитовые, арифметические;
  2. Физический уровень представления СД:
     1. Схему хранения: последовательная;
     2. Объем памяти, занимаемый экземпляром СД: 4 байта;
     3. Формат внутреннего представления СД и способ его интерпретации: хранится как 4-х байтовое целое, первый бит отвечает за знак числа.
     4. Характеристику допустимых значений: -32768...32767  
        (-2147483648…2147483647);
     5. Тип доступа к элементам: прямой;
  3. Логический уровень представления СД.  
     Способ описания СД и экземпляра СД на языке программирования: int i;

**Тип2: float**

* 1. Абстрактный уровень представления СД:
     1. Характер организованности и изменчивости: простейший, статический;
     2. Набор допустимых операций: инициализация, доступ, присваивание, сравнения, побитовые, арифметические;
  2. Физический уровень представления СД:
     1. Схему хранения: последовательная;
     2. Объем памяти, занимаемый экземпляром СД: 4 байта;
     3. Формат внутреннего представления СД и способ его интерпретации: хранится как 4-х байтовое целое, первый бит отвечает за знак числа, следующие 8 бит отвечают порядок, следующие 23 бита отвечают за мантису.
     4. Характеристику допустимых значений: 1.1754943E-38…1.175494E+38;
     5. Тип доступа к элементам: прямой;
  3. Логический уровень представления СД.  
     Способ описания СД и экземпляра СД на языке программирования: float f;

**Тип3: {red, yellow, green}colors**

* 1. Абстрактный уровень представления СД:
     1. Характер организованности и изменчивости: простейший, статический;
     2. Набор допустимых операций: инициализация, доступ, присваивание, сравнения, побитовые, арифметические;
  2. Физический уровень представления СД:
     1. Схему хранения: последовательная;
     2. Объем памяти, занимаемый экземпляром СД: 6 байт;
     3. Формат внутреннего представления СД и способ его интерпретации: принимает только те значения, которые были заданы изначально. Максимальная мощность перечисляемого типа составляет 65536 значений.
     4. Характеристику допустимых значений: 0...2;
     5. Тип доступа к элементам: прямой;
  3. Логический уровень представления СД.  
     Способ описания СД и экземпляра СД на языке программирования:  
     enum colors {  
      red,  
      yellow,  
      green  
     };

**Задание 2:**

**int:**1. 178  
2. -7895

**float:**1. 56.48  
2. -126.234

**{red, yellow, green}colors**1. red  
2. green

**Задание 3:**

**int:**

1. ;

;

;

;

;

;

;

17810 = 0000 0000 0000 0000 0000 0000 1011 00102;

2. ;

;

;

;

;

;

;

;

;

;

;

;

Помним, что отрицательные числа хранятся в дополнительном коде, для этого в полученном двоичном представлении числа меняем единицы на нули и наоборот. Затем прибавляем единицу. Не забываем про знаковый бит в самом начале:

1000 0000 0000 0000 0001 1110 1101 0111;

1111 1111 1111 1111 1110 0001 0010 1000;

-789510 = 1111 1111 1111 1111 1110 0001 0010 10012;

**float:**1. Чтобы перевести вещественное число, необходимо вначале перевести его целую, затем дробную части (возьмём точность в 6 знаков):

; ;

; ;

; ;

; ;

; ;

;

;

Полученное двоично число приведём к виду , где целая часть равна 12:

111000.01111 = 1.1100001111\*25;

Получаем смещённый порядок:

5+127 = 13210 = 1000 0100;

Запишем полученное число в виде: 1й бит – знаковый, 8 бит – порядок, остальные биты – число, учитываем, что целая единица не записывается.

56.4810 = 0 1000 0100 110 0001 1110 0000 0000 00002;

2. ; ;

; ;

; 0|936;

; 1|872;

; 1|744;

; 1|488;

1111110.00111 = 1.11111000111\*26;

6+127 = 13310 = 1000 01012;

-126.23410 = 1 1000 0101 111 1100 0111 0111 1100 00002;

**{red, yellow, green}colors**1. 0000 0000 0000 0000;  
2. 0000 0000 0000 0010;

**Задание 4:**

**int:**1. Для перевода из десятичной в двоичную целого числа необходимо установить позиции битов справа налево начиная с нулевой. Необходимое десятичное число получится в результате суммы двоек в степени позиций значимых битов.

;

2. Для перевода отрицательного двоичного числа в десятичное, необходимо перевести дополнительный код в прямой, для этого отнимем один бит от исходного числа и инвертируем полученное, помним, что первый бит знаковый.

**1**110 0001 0010 10012(ДК) = **1**110 0001 0010 10002(ОК) = **1**001 1110 1101 01112(ПК)

Далее по схеме из пункта 1, не забываем про минус.

;

**float:**1. Чтобы перевести вещественное число из двоичной записи в десятичную, нужно вспомнить, что храниться в каждом из битов. В нашем случае (тип float) первый бит знаковый, последующие 8 – порядок, остальные 27 – мантиса. Для начала вычислим порядок k, помним что он хранится в двоичном представлении как 127+k.

1000 01002 = 132 = 127+5; k=5;

Теперь переведём вещественное число из нормализованного вида в стандартный, помним что мантиса хранится памяти как вещественное число с одним целым. Получим:

1.1100001111\*25=111000.01111

Знаем, как перевести целую часть:

1110002 = 8+16+32 = 5610;

Дробная часть переводится путём сложения числа 0.5 (т.е. ) в степени позиции значащего бита слева направо, начиная с первой.

0.01112 = 0.46875;

В итоге получим

0 1000 0100 110 0001 1110 0000 0000 00002 = 56.46875

2. Отрицательное десятичное число переводится абсолютно так же, необходимо лишь помнить про знаковый бит.

В нашем случае:

Порядок: 1000 01012 = 133 = 127+6;

1.11111000111011111\*26 = 1111110.00111011111;

Снизим точность до 6 знаков после запятой.

Как нетрудно было догадаться , поэтому получим:

1111110.001112 = 26\*1 + 25\*1 + 24\*1 + 23\*1 + 22\*1 + 21\*1 + 20\*0 + 2-1\*0 + 2-2\*0 + 2-3\*1 +  
+ 2-4\*1 + 2-5\*1 = 64+32+16+8+4+2+0.125+0.0625+0.03125+0.015625=126.23437510;

Не забываем про знаковый бит. Получим:

1 1000 0101 111 1100 0111 0111 1100 00002 = -126.23410;

**{red, yellow, green}colors**1. red находится на нулевой позиции, следовательно:

0000 0000 0000 0000 = red;

2. Green находится на второй позиции, следовательно:

0000 0000 0000 0010 = green;

**Задание 5:**

#include <stdio.h>

#include <stdlib.h>

enum colors {

red,

yellow,

green

};

void PrintByte(unsigned char a) {

unsigned int m = 128, i = 0;

unsigned char t = a;

for (i = 0; i < 8; i++) {

printf("%d", (t & m) ? 1 : 0);

t <<= 1;

}

printf(" ");

}

void PrintVar(void\* a, unsigned int size) {

char i;

unsigned char\* byte = (unsigned int\*)a;

for (i = size - 1; i >= 0; i--) PrintByte(byte[i]);

byte = NULL;

printf("\n");

}

void main() {

int a = 178, b = -7895;

PrintVar(&a, sizeof(int));

PrintVar(&b, sizeof(int));

float x = 56.48, y = -126.234;

PrintVar(&x, sizeof(float));

PrintVar(&y, sizeof(float));

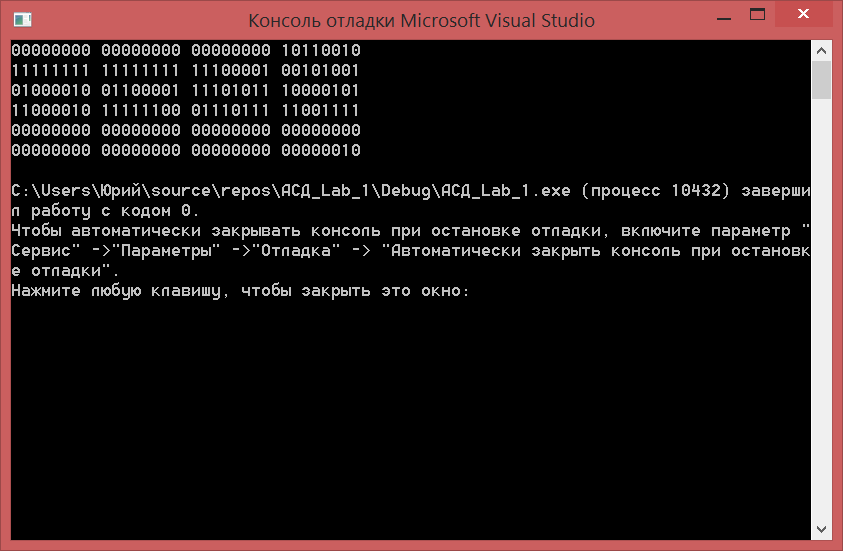
enum colors n = red, m = green;

PrintVar(&n, sizeof(enum colors));

PrintVar(&m, sizeof(enum colors));

}

**Задание 6:**



**Задание 7:**

#include <stdio.h>

#include <string.h>

int conv\_i(char\* s)

{

char len = strlen(s);

char size = len / 8;

char a[4];

int\* b;

int mask = 1, i = 0, j = 0, st = 0;

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

{

for (j = 0; j < 8; j++)

{

a[i] <<= 1;

if (s[st++] == '1')

a[i] |= mask;

}

}

b = (int\*)a;

return \*b;

}

float conv\_f(char\* s)

{

char len = strlen(s);

char size = len / 8;

char a[4];

float\* b;

int mask = 1, i = 0, j = 0, st = 0;

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

{

for (j = 0; j < 8; j++)

{

a[i] <<= 1;

if (s[st++] == '1')

a[i] |= mask;

}

}

b = (float\*)a;

return \*b;

}

void main()

{

char\* s1 = "00000000000000000000000010110010";

char\* s2 = "11111111111111111110000100101001";

char\* s3 = "01000010011000011110101110000000";

char\* s4 = "11000010111111000111011111000000";

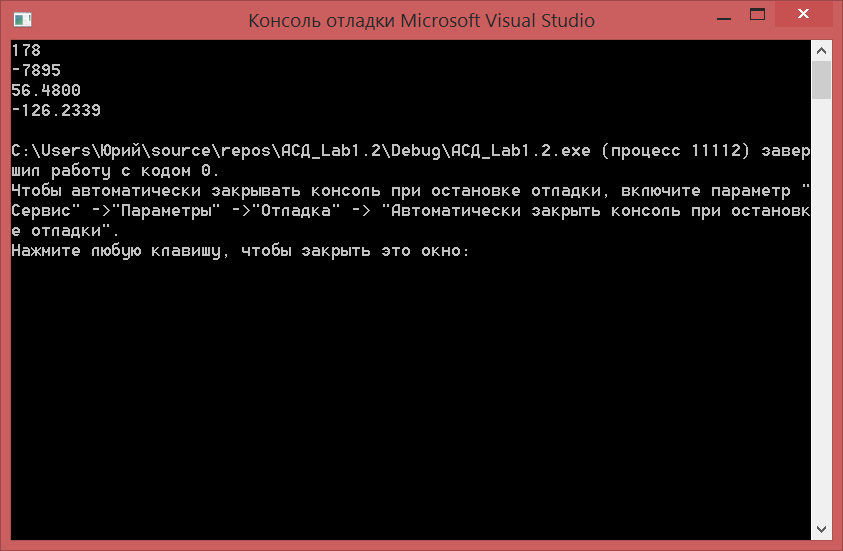
printf("%i\n", conv\_i(s1));

printf("%i\n", conv\_i(s2));

printf("%.4f\n", conv\_f(s3));

printf("%.4f\n", conv\_f(s4));

}



**Вывод:** В ходе выполнения лабораторной работы мы определили абстрактный, физический и логический уровни представления для различных типов данных, преобразовали значения типов в двоичный код и двоичный код в значение. Результаты работы программы по преобразованию совпали с результатами ручной проверки.