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/C)»

Выполнил: ст. группы ВТ-201
Билык Дмитрий Анатольевич

Проверил:
Синюк Василий Григорьевич

Белгород 2021 г.

Встроенные структуры данных (Pascal/C)

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:
   1.1. Абстрактный уровень представления СД:
      1.1.1. Характер организованности и изменчивости.
      1.1.2. Набор допустимых операций.
   1.2. Физический уровень представления СД:
      1.2.1. Схему хранения.
      1.2.2. Объем памяти, занимаемый экземпляром СД.
      1.2.3. Формат внутреннего представления СД и способ его интерпретации.
      1.2.4. Характеристику допустимых значений.
      1.2.5. Тип доступа к элементам.
   1.3. 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*.

6. Обработать программой значения, полученные в результате выполнения пункта 3 задания. Сделать выводы.

7. 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.

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

Задание 1:

Тип1: int

1.1. Абстрактный уровень представления СД:
- 1.1.1. Характер организованности и изменчивости: простейший, статический;
- 1.1.2. Набор допустимых операций: инициализация, доступ, присваивание, сравнения, побитовые, арифметические;

1.2. Физический уровень представления СД:
- 1.2.1. Схему хранения: последовательная;
- 1.2.2. Объем памяти, занимаемый экземпляром СД: 4 байта;
- 1.2.3. Формат внутреннего представления СД и способ его интерпретации: хранится как 4-х байтовое целое, первый бит отвечает за знак числа.
- 1.2.4. Характеристику допустимых значений: -32768...32767 (-2147483648…2147483647);
- 1.2.5. Тип доступа к элементам: прямой;

1.3. Логический уровень представления СД.
Способ описания СД и экземпляра СД на языке программирования: int i;

Тип2: long double

1.1. Абстрактный уровень представления СД:
- 1.1.1. Характер организованности и изменчивости: простейший, статический;
- 1.1.2. Набор допустимых операций: инициализация, доступ, присваивание, сравнения, побитовые, арифметические;

1.2. Физический уровень представления СД:
- 1.2.1. Схему хранения: последовательная;
- 1.2.2.Объем памяти, занимаемый экземпляром СД: 10 байт;
- 1.2.3. Формат внутреннего представления СД и способ его интерпретации: хранится как 10 байтовое целое, первый бит отвечает за знак числа, следующие 11 бит отвечают порядок, следующие 52 бита отвечают за мантису.
- 1.2.4. Характеристику допустимых значений: 3.4E–4932…3.4E+4932;
- 1.2.5. Тип доступа к элементам: прямой;

1.3. Логический уровень представления СД.
Способ описания СД и экземпляра СД на языке программирования: long double f;

Тип3: {winter, spring, summer,autumn}seeson

1.1. Абстрактный уровень представления СД:
- 1.1.1. Характер организованности и изменчивости: простейший, статический;
- 1.1.2. Набор допустимых операций: инициализация, доступ, присваивание, сравнения, побитовые, арифметические;

1.2. Физический уровень представления СД:

1.2.1. Схему хранения: последовательная;

1.2.2. Объем памяти, занимаемый экземпляром СД: 8 байт;

1.2.3. Формат внутреннего представления СД и способ его интерпретации: принимает только те значения, которые были заданы изначально. Максимальная мощность перечисляемого типа составляет 4 значения.

1.2.4. Характеристику допустимых значений: 0...3;

1.2.5. Тип доступа к элементам: прямой;

1.3. 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. $178/2=89(0)$;
   $89/2=44(1)$;
   $44/2=22(0)$;
   $22/2=11(0)$;
   $11/2=5(1)$;
   $5/2=2(1)$;
   $2/2=1(0)$;

178_{10} = 0000 0000 0000 0000 0000 0000 1011 0010_2;

2. $7895/2=3947(1)$;
   $3947/2=1973(1)$;
   $1973/2=986(1)$;
   $986/2=493(0)$;
   $493/2=246(1)$;
   $246/2=123(0)$;
   $123/2=61(1)$;
   $61/2=30(1)$;
   $30/2=15(0)$;
   $15/2=7(1)$;
   $7/2=3(1)$;
   $3/2=1(1)$;

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;
-7895_{10} = 1111 1111 1111 1111 1110 0001 0010 1001_2;

float:

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

$56/2=28(0)$; $0 \vee 48$;
$28/2=14(0)$; $0 \vee 96$;
$14/2=7(0)$; $1 \vee 92$;
$7/2=3(1)$; $1 \vee 84$;
$3/2=1(1)$; $1 \vee 68$;
$1 \vee 36$;
$0 \vee 72$;

Полученное двоично число приведём к виду $M * 2^p$, где целая часть равна 1_2:

111000.01111 = $1.1100001111*2^5$;

Получаем смещённый порядок:

$5+127 = 132_{10}$ = 1000 0100;

Запишем полученное число в виде: $1_{\text{й}}$ бит – знаковый, 8 бит – порядок, остальные биты – число, учитываем, что целая единица не записывается.

56.48_{10} = 0 1000 0100 110 0001 1110 0000 0000 0000_2;

2. $126/2=63(0)$; $0 \vee 234$;

$63/2=31(1)$; $0 \vee 468$;

$31/2=15(1)$; $0|936$;

$15/2=7(1)$; $1|872$;

$7/2=3(1)$; $1|744$;

$3/2=1(1)$; $1|488$;

$1111110.00111 = 1.11111000111*2^6$;

$6+127 = 133_{10} = 1000\ 0101_2$;

-126.234_{10} = 1 1000 0101 111 1100 0111 0111 1100 0000_2;

{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.

$$10110010_2=2^0*0+2^1*1+2^2*0+2^3*0+2^4*1+2^5*1+2^6*0+2^7*1=¿2+16+32+128=178_{10};$$

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.

1110 0001 0010 1001_2(ДК) = **1**110 0001 0010 1000_2(ОК) = **1**001 1110 1101 0111_2(ПК)

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

$$1\,11111011010111_2=1+2+4+16+64+128+512+1024+2048+4096=7895_{10};$$

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.

$1000\ 0100_2 = 132 = 127+5$; k=5;

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

$1.1100001111*2^5=111000.01111$

Знаем, как перевести целую часть:

$111000_2 = 8+16+32 = 56_{10}$;

Дробная часть переводится путём сложения числа 0.5 (т.е. $\frac{1}{2}$) в степени позиции значащего бита слева направо, начиная с первой.

$0.0111_2 = \frac{1}{2^1}*0+\frac{1}{2^2}*1+\frac{1}{2^3}*1+\frac{1}{2^4}*1=0.25+0.125+0.0625++0.03125=¿0.46875;$

В итоге получим

$0\ 1000\ 0100\ 110\ 0001\ 1110\ 0000\ 0000\ 0000_2 = 56.46875$

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

В нашем случае:

Порядок: $1000\ 0101_2 = 133 = 127+6;$

$1.11111000111011111*2^6 = 1111110.00111011111;$

Снизим точность до 6 знаков после запятой.

Как нетрудно было догадаться $\frac{1}{2}=2^{-1}$, поэтому получим:

$1111110.00111_2 = 2^6*1 + 2^5*1 + 2^4*1 + 2^3*1 + 2^2*1 + 2^1*1 + 2^0*0 + 2^{-1}*0 + 2^{-2}*0 + 2^{-3}*1 +$
$+ 2^{-4}*1 + 2^{-5}*1 = 64+32+16+8+4+2+0.125+0.0625+0.03125+0.015625=126.234375_{10};$

Не забываем про знаковый бит. Получим:

$1\ 1000\ 0101\ 111\ 1100\ 0111\ 0111\ 1100\ 0000_2 = -126.234_{10};$

{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:

```
00000000 00000000 00000000 10110010
11111111 11111111 11100001 00101001
01000010 01100001 11101011 10000101
11000010 11111100 01110111 11001111
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000010
```

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));
}
```

```
178
-7895
56.4800
-126.2339
```

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.