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 ПВ-211

Чувилко Илья Романович

Проверил:

Синюк Василий Григорьевич

Белгород 2022 г.

Встроенные структуры данных (Pascal/C)
Вариант 24

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

Задание

1. Для типов данных 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 `S` в 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;

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

Тип3: int массив [6]

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: 24 байта;

1.2.3. Формат внутреннего представления СД и способ его интерпретации: принимает только те значения, которые были заданы изначально. Максимальная мощность перечисляемого типа составляет $6*2^{32}$ значения.

1.2.4. Характеристику допустимых значений: (-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 a[6] = {0, 1, 2, 3, 4, 5}
```

Задание 2:

int:

1. 178
2. -7895

long double:

1. 56.48
2. -126.234

int массив [6]:

1. {0, 1, 2, 3, 4, 5}
2. {253, 148, 1111, 777, 321, 999999}

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;
long double:
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} = $1\ 1000\ 0101\ 111\ 1100\ 0111\ 0111\ 1100\ 0000_2$;

int массив [6]:

1. Представление массива {0, 1, 2, 3, 4, 5} в памяти:

0_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2$

1_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2$

2_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2$

3_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0011_2$

4_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0100_2$

5_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0101_2$

Адрес в памяти	Запись в двоичном коде
ptr + 0 * sizeof(int)	0000 0000 0000 0000 0000 0000 0000 0000
ptr + 1 * sizeof(int)	0000 0000 0000 0000 0000 0000 0000 0001
ptr + 2 * sizeof(int)	0000 0000 0000 0000 0000 0000 0000 0010
ptr + 3 * sizeof(int)	0000 0000 0000 0000 0000 0000 0000 0011
ptr + 4 * sizeof(int)	0000 0000 0000 0000 0000 0000 0000 0100
ptr + 5 * sizeof(int)	0000 0000 0000 0000 0000 0000 0000 0101

2. Представление массива {253, 148, 1111, 777, 321, 999999} в памяти:

253_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111\ 1101_2$

148_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1001\ 0100_2$

$1\ 111_{10}$ = $0000\ 0000\ 0000\ 0000\ 0000\ 0100\ 0101\ 0111_2$

777_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0011\ 0000\ 1001_2$

321_{10} = $0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 0100\ 0001_2$

$999\ 999_{10}$ = $0000\ 0000\ 0000\ 1111\ 0100\ 0010\ 0011\ 1111_2$

Адрес в памяти	Запись в двоичном коде
ptr + 0 * sizeof(int)	0000 0000 0000 0000 0000 0000 1111 1101
ptr + 1 * sizeof(int)	0000 0000 0000 0000 0000 0000 1001 0100
ptr + 2 * sizeof(int)	0000 0000 0000 0000 0000 0100 0101 0111
ptr + 3 * sizeof(int)	0000 0000 0000 0000 0000 0011 0000 1001
ptr + 4 * sizeof(int)	0000 0000 0000 0000 0000 0001 0100 0001
ptr + 5 * sizeof(int)	0000 0000 0000 1111 0100 0010 0011 1111

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.

$10110010_2 = 2^1*1 + 2^4*1 + 2^5*1 + 2^7*1 = 2 + 16 + 32 + 128 = \mathbf{178_{10}}$;

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.

1$110\ 0001\ 0010\ 1001_2$(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$_{10}$**

long double:

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$_{10}$;

int массив [6]:

1. {0, 1, 2, 3, 4, 5}:

0000 0000 0000 0000 0000 0000 0000 0000$_2$ = **0$_{10}$**

0000 0000 0000 0000 0000 0000 0000 0001$_2$ = 2^0 = **1$_{10}$**

0000 0000 0000 0000 0000 0000 0000 0010$_2$ = 2^1 = **2$_{10}$**

0000 0000 0000 0000 0000 0000 0000 0011$_2$ = 2^0+2^1 = **3$_{10}$**

0000 0000 0000 0000 0000 0000 0000 0100$_2$ = 2^2 = **4$_{10}$**

0000 0000 0000 0000 0000 0000 0000 0101$_2$ = 2^2+2^0 = **5$_{10}$**

2. {253, 148, 1111, 777, 321, 999999}:

0000 0000 0000 0000 0000 0000 1111 1101$_2$ = 2^0+2^2+2^3+2^4+2^5+2^6+2^7 = **253$_{10}$**

0000 0000 0000 0000 0000 0000 1001 0100$_2$ = 2^2+2^4+2^7 = **148$_{10}$**

0000 0000 0000 0000 0000 0100 0101 0111$_2$ = 2^0+2^1+2^2+2^4+2^6+2^{10} = **1 111$_{10}$**

0000 0000 0000 0000 0000 0011 0000 1001$_2$ = 2^0+2^3+2^8+2^9 = **777$_{10}$**

0000 0000 0000 0000 0000 0001 0100 0001$_2$ = 2^0+2^6+2^8 = **321$_{10}$**

0000 0000 0000 1111 0100 0010 0011 1111$_2$ = 2^0+2^1+2^2+2^3+2^4+2^5+2^9+2^{14}+2^{16}+2^{17}+2^{18}+2^{19} = **999 999$_{10}$**

Задание 5:

```c
#include <stdio.h>

#define ARRAY_SIZE 6

// Выводит представление в память одного байта по адресу a
void PrintByte(const unsigned char *a) {
  for (int i = sizeof(char) * 8 - 1; i >= 0; i--)
    printf("%d", ((1 << i) & *a) != 0 ? 1 : 0);
  printf(" ");
}

// Выводит представление в памяти переменной по адресу a, размера size
void PrintVar(void *a, int size) {
  for (int i = size - 1; i >= 0; i--)
    PrintByte((unsigned char *) a + i);
  printf("\n");
}

// Выводит на экран представление структуру по адресу a размером size.
// s - именование переменной для вывода строки
void PrintTest(void *a, int size, char *s) {
  printf("%s", s);
  PrintVar(a, size);
}

// Для целочисленных типа int
void Test_Integers_DTB() {
  printf("INTEGERS\n");
  int a = 178;
  PrintTest(&a, sizeof(int), "int a = ");
  int b = -7895;
  PrintTest(&b, sizeof(int), "int b = ");
  printf("\n");
}

// Для вещественных типа long double
void Test_LongDoubles_DTB() {
  printf("LONG DOUBLE\n");
  long double x = 56.48;
  PrintTest(&x, sizeof(long double), "long double x = ");
  long double y = -126.234;
  PrintTest(&y, sizeof(long double), "long double y = ");
  printf("\n");
}

// Массив целых чисел
void Test_IntArray_DTB() {
  printf("INT ARRAY\n");
  int n[ARRAY_SIZE] = {0, 1, 2, 3, 4, 5};
  PrintTest(n, sizeof(int) * ARRAY_SIZE, "int n[6] = ");
  int m[ARRAY_SIZE] = {253, 148, 1111, 777, 321, 999999};
  PrintTest(m, sizeof(int) * ARRAY_SIZE, "int m[6] = ");
  printf("\n");
}

void TestsDecToBin() {
  Test_Integers_DTB();
  Test_LongDoubles_DTB();
  Test_IntArray_DTB();
}
```

Задание 6:

```
       Task 5
INTEGERS
int a = 00000000 00000000 00000000 10110010
int b = 11111111 11111111 11100001 00101001

LONG DOUBLE
long double x = 00000000 00000000 00000000 10110010 11111111 11111111 01000000 00000100 11100001 11101011 10000101 00011
110 10111000 01010001 11101000 00000000
long double y = 00000000 00000000 00000000 00000000 00000000 00000000 11000000 00000101 11111100 01110111 11001110 11011
001 00010110 10000111 00101000 00000000

INT ARRAY
int n[6] = 00000000 00000000 00000000 00000101 00000000 00000000 00000000 00000100 00000000 00000000 00000000 00000011 0
0000000 00000000 00000000 00000010 00000000 00000000 00000000 00000001 00000000 00000000 00000000 00000000
int m[6] = 00000000 00001111 01000010 00111111 00000000 00000000 00000001 01000001 00000000 00000000 00000011 00001001 0
0000000 00000000 00000100 01010111 00000000 00000000 00000000 10010100 00000000 00000000 00000000 11111101
```

Задача 7

https://ejudge.179.ru/tasks/cpp/total/061.html

```c
#include  <stdio.h>

#define BIT_IN_BYTE 8
#define ARRAY_SIZE 6

void GetByteFromString(const char *s, char *res) {
  for (int i = BIT_IN_BYTE; i > 0; i--)
    if (s[i - 1] == '1')
      *res |= 1 << (BIT_IN_BYTE - i);
}

void GetVarFromString(char *s, void *res, int size) {
  for (int i = 0; i < size; i++)
    GetByteFromString(s + i * BIT_IN_BYTE, res + size - i - 1);
}

void Test_Integers_BTD() {
  printf("INTEGERS\n");
  int a = 0;
  char s1[] = "00000000000000000000000010110010";
  GetVarFromString(s1, &a, sizeof(int));
  printf("int a = %d\n", a);

  int b = 0;
  char s2[] = "11111111111111111110000100101001";
  GetVarFromString(s2, &b, sizeof(int));
  printf("int b = %d\n\n", b);
}

void Test_LongDouble_BTD() {
  printf("LONG DOUBLE\n");
  long double x = 0;
  char s1[] = "00000000000000000000000010110010"
              "11111111111111110100000000000100"
              "11100001110101110000101000111110"
              "10111000010100011110100000000000";
  GetVarFromString(s1, &x, sizeof(long double));
  printf("long double x = %Lf\n", x);
```

```
  long double y = 0;
  char s2[] = "00000000000000000000000000000000"
              "00010000000000011100000000000101"
              "11111100011101111100111011011001"
              "00010110100001110010100000000000";
  GetVarFromString(s2, &y, sizeof(long double));
  printf("long double y = %Lf\n\n", y);
}

// Массив целых чисел
void Test_IntArray_BTD() {
  printf("INT ARRAY\n");
  int n[ARRAY_SIZE] = {0,0,0,0,0,0};
  char s1[] = "00000000000000000000000000000101"
              "00000000000000000000000000000100"
              "00000000000000000000000000000011"
              "00000000000000000000000000000010"
              "00000000000000000000000000000001"
              "00000000000000000000000000000000";
  GetVarFromString(s1, n, sizeof(int) * ARRAY_SIZE);
  printf("int n[6] = {");
  for (int i = 0; i < ARRAY_SIZE; i++)
    printf("%d, ", n[i]);
  printf("\b\b}\n");

  int m[ARRAY_SIZE] = {0,0,0,0,0,0};
  char s2[] = "00000000000011110100001000111111"
              "00000000000000000000000101000001"
              "00000000000000000000001100001001"
              "00000000000000000000010001010111"
              "00000000000000000000000010010100"
              "00000000000000000000000011111101";
  GetVarFromString(s2, m, sizeof(int) * ARRAY_SIZE);
  printf("int m[6] = {");
  for (int i = 0; i < ARRAY_SIZE; i++)
    printf("%d, ", m[i]);
  printf("\b\b}");
}

void TestsBinToDec() {
  Test_Integers_BTD();
  Test_LongDouble_BTD();
  Test_IntArray_BTD();
}
```

Задание 8

```
        Task 7
INTEGERS
int a = 178
int b = -7895

LONG DOUBLE
long double x = 0.000000
long double y = 0.000000

INT ARRAY
int n[6] = {0, 1, 2, 3, 4, 5}
int m[6] = {253, 148, 1111, 777, 321, 999999}
Process finished with exit code 0
```

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.