**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ 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 м СД

Тема: «Встроенные структуры данных(Pascal/C)»

Выполнил: ст.группы

Проверил: Синюк В.Г.

Белгород

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

З а д а н и е

1. Для типов данных (см. Варианты заданий в таблицах 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 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.

**Вариант 4**

**Тип 1: unsigned char**

**Тип 2: float**

**Тип 3: float массив[10][10]**

**Задание 1:**

**Тип: unsigned char**

1. Абстрактный уровень представления СД:
   1. Характер организованности и изменчивости: **упорядоченный, статический.**
   2. Набор допустимых операций: **доступ, инициализация, сравнение, присваивание, арифметические операции.**
2. Физический уровень представления СД:
   1. Схема хранения: **последовательная**
   2. Объём памяти, занимаемой экземпляром СД: **1 байт.**
   3. Формат внутреннего представления СД и способ его интерпретации: **Значение типа сhar хранится в памяти как 1-ух байтовое число.**
   4. Характеристика допустимых значений: **[0..255]**
   5. Тип доступа к элементам: **прямой**
3. Логический уровень представления СД:

3.1 Способ описания СД и экземпляра СД на языке программирования: **char n1;**

**Тип: float**

1. Абстрактный уровень представления СД:
   1. Характер организованности и изменчивости. **неупорядоченный, статический.**
   2. Набор допустимых операций: **доступ, инициализация, сравнение, присваивание, арифметические операции.**
2. Физический уровень представления СД:
   1. Схема хранения: **последовательная.**
   2. Объём памяти, занимаемой экземпляром СД: **4 байт.**
   3. Формат внутреннего представления СД и способ его интерпретации: **значение типа flaot хранится в памяти как 4 байтное число со знаком. Условно можно разбить на поля:**

**Поле знака – 1 бит, поле мантиссы – 8 бит, поле порядка – 23 бит.**

s=1бит

m=23 бит

e=8 бит

* 1. Характеристика допустимых значений: **[1.1754943E–38...1.175494E+38]**
  2. Тип доступа к элементам: **прямой.**

1. Логический уровень представления СД:
   1. Способ описания СД и экземпляра СД на языке программирования: **float n2;**

**Тип: float массив[10][10]**

1. Абстрактный уровень представления СД:
   1. Характер организованности и изменчивости. **упорядоченный, статический.**
   2. Набор допустимых операций: **доступ, инициализация, сравнение, присваивание, арифметические операции .**
2. Физический уровень представления СД:
   1. Схема хранения: **последовательная.**
   2. Объём памяти, занимаемой экземпляром СД: **4 байта.**
   3. Формат внутреннего представления СД и способ его интерпретации: **значение типа float массив [10][10] хранится в памяти как 4-ёх байтовое число. Каждое число получает порядковый номер строки и номер столбца начиная с 0 .**
   4. Характеристика допустимых значений: **[0..10][0..10].**
   5. Тип доступа к элементам: **прямой.**
3. Логический уровень представления СД:
   1. Способ описания СД и экземпляра СД на языке программирования:**float a[10][10];**

**Задание 2:**

Тип 1. 120 (x)

Тип 2. 16,1234

Тип 3. 21,45 (a[0][1])

**Задание 3:**

Тип 1. 00000000 01111000

Тип 2. 01000001 10000000 11111100 10111001

Тип 3. 01000001 10101011 10011001 10011010

**Задание 4:**

Тип 1. 120 (x)

Тип 2. 16,1234

Тип 3. 21,45 (a[0][1])

**Задание 5:**

Спецификация процедуры PrintByte:

1. Заголовок: procedure 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.

Спецификация процедуры PrintVar:

1. Заголовок: procedure 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.

#include <stdio.h>

void PrintByte (unsigned char a)

{

int i;

for (i = 7; i >= 0; i--)

{

printf ("%d", (a & 128) == 128);

a = a << 1;

}

printf(" ");

}

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

{

char \*s=(char \*)a;

int i;

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

{

PrintByte(s[i]);

printf(" ");

}

}

int main ()

{

unsigned short n1 = 120;

float n2 = 16.1234;

float a[10][10];

a[0][1]=21.45;

printf ("\nЗначение числа %hu типа unsigned short: \n", n1);

PrintVar(&n1, sizeof(unsigned short));

printf ("\nЗначение числа %.4f типа float: \n", n2);

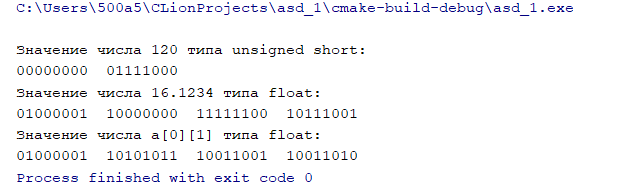
PrintVar(&n2, sizeof(float));

printf("\nЗначение числа a[0][1] типа float: \n");

PrintVar(&a[0][1], sizeof(float));

}

**Задание 6:**



Вывод: Мы получили такие же результаты, что и при переводе в ручную.

**Задание 7:**

Спецификация процедуры m\_byte:

1. Заголовок: char \* m\_byte (char \*s)

2. Назначение: возвращает массив байтов соответствующий массиву битов в строке s

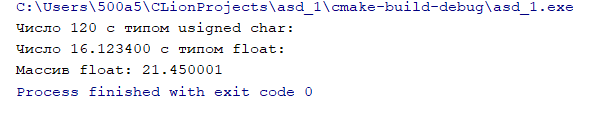
3. Входные параметры: s

4. Выходные параметры: нет.

#include **<stdio.h>**  
**char** \*m\_byte(**char** \*s)  
{  
 size\_t len = strlen(s);  
  
 **unsigned char** \*a = calloc(len / 8, 1);  
 **unsigned char** t;  
 **int** i\_byte = 0, i\_bite = 0;  
 **for** (**int** i = len - 1; i > -1 ; --i){  
  
 t = s[i] - **'0'**;  
  
 a[i\_byte] |= t << i\_bite;  
  
 i\_bite++;  
  
 **if**(i\_bite == 8){  
  
 i\_bite = 0;  
  
 i\_byte++;  
  
 }  
  
  
 }  
  
 **return** a;  
}  
**int** main ()  
{  
 **char** \*s=**"01000001100000001111110010111001"**;  
 **char** \*m;  
 **char** \*s1=**"0000000001111000"**;  
 **char** \*s2=**"01000001100000001111110010111001"**;  
 **char** \*s3=**"01000001101010111001100110011010"**;  
  
 **unsigned short** ,b;  
 **float** *n2* ;  
 **float** a[10][10];  
  
  
  
 m = m\_byte(s1);  
b = \*((**unsigned char**\*)m);  
printf(**"Число %u с типом usigned char: \n"**,b);  
free(m);  
m = m\_byte(s2);  
c = \*((**float** \*)m);  
printf(**"Число %lf с типом float:\n"**, c);  
free(m);  
m = m\_byte(s3);  
a[0][1] = \*((**float** \*)m);  
printf(**"Массив float: %f"**,a[0][1]);  
free(a);

}

**Задание 8:**



Вывод: Мы получили практически те же результаты, что и при переводе в ручную. Погрешность минимальная.