

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-223
Пахомов Владислав Андреевич

Проверили: асс. Солонченко Роман
Евгеньевич

Белгород 2023 г.

Лабораторная работа №1
Встроенные структуры данных (Pascal/C)
Вариант 10

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

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.

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

Задания для С

Тип 1	Тип 2	Тип 3
signed char	float	{red, yellow, green} colors

signed char

1.1. Абстрактный уровень представления СД:

1.1.1. Характер организованности - **простой**

Характер изменчивости - **статический**

1.1.2. Набор допустимых операций - **математические операции, побитовые операции, присваивание, инициализация, логические операции, приведение типа, взятие адреса**

1.2. Физический уровень представления СД:

1.2.1. Схема хранения - **последовательная память.**

1.2.2. Объем памяти, занимаемый экземпляром СД. Размер **signed char** в современном С гарантированно равен **1 байту или 8 битам.**

1.2.3. Формат внутреннего представления СД и способ его интерпретации. **8-битное число.** Старший бит отводится под хранение знака числа, отрицательные числа хранятся в **дополнительном коде**, положительные - **в прямом.**

1.2.4. Характеристика допустимых значений. $E(\text{signed char}) \in [-2^7; 2^7 - 1]$ или $E(\text{signed char}) \in [-128; 127]$.

1.2.5. Тип доступа к элементам - **прямой**.

1.3. Логический уровень представления СД.

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

```
signed char a;  
char a;
```

float

1.1. Абстрактный уровень представления СД:

1.1.1. Характер организованности - **простой**
Характер изменчивости - **статический**

1.1.2. Набор допустимых операций - **математические операции, побитовые операции, присваивание, инициализация, логические операции, приведение типа, взятие адреса**

1.2. Физический уровень представления СД:

1.2.1. Схема хранения - **последовательная память**.

1.2.2. Объем памяти, занимаемый экземпляром СД. Размер `float` может различаться на различных системах. Однако на большинстве ПК размер `float` равен **4 байтам или 32 битам**.

1.2.3. Формат внутреннего представления СД и способ его интерпретации. **32-битное число**. Старший бит `s` отводится под хранение знака числа. Следующие 8 бит `e` содержат порядок числа, последние 23 бита содержат мантиссу числа `m`. Число в памяти можно выразить следующей формулой: $(-1)^s \cdot 1.m \cdot 2^e$

1.2.4. Характеристика допустимых значений.
 $E(\text{float}) \in [1.1754943 \cdot 10^{-38}; 1.1754943 \cdot 10^{38}]$.

1.2.5. Тип доступа к элементам - **прямой**.

1.3. Логический уровень представления СД.

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

```
float a;
```

{red, yellow, green} colors

1.1. Абстрактный уровень представления СД:

1.1.1. Характер организованности - **линейный**
Характер изменчивости - **статический**

1.1.2. Набор допустимых операций - **объявление, получение значения по идентификатору**

1.2. Физический уровень представления СД:

- 1.2.1. Схема хранения - **последовательная память**.
- 1.2.2. Объем памяти, занимаемый экземпляром СД.
enum представляет из себя список констант, каждой из которых присвоено значение типа int. Размер СД будет равен $N \cdot S$, где N - количество элементов, $S = \text{sizeof}(\text{int})$.
- 1.2.3. Формат внутреннего представления СД и способ его интерпретации.
Последовательность N элементов одного типа int.
- 1.2.4. Характеристика допустимых значений.
Максимальная мощность равна $2^{\text{sizeof}(\text{int}) \cdot 8} = 4294967296$.
- 1.2.5. Тип доступа к элементам - **прямой**.
- 1.3. Логический уровень представления СД.
 - 1.3.1. Способ описания СД и экземпляра СД на языке программирования.

```
typedef enum {  
    RED, // 0  
    GREEN, // 1  
    BLUE // 2  
} Color;  
// Можно также вручную задать значения констант.  
typedef enum {  
    RED_C = 45,  
    GREEN_C, // 46  
    BLUE_C = RED_C + 4 //49  
} ColorCustomNumeration;
```

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

signed char

1. -12
2. 55

float

1. 12.5
2. -431.75

{red, yellow, green} colors

1. red
2. green

3. Преобразовать значения в двоичный код.

signed char

1. -12

$$-12_{10} = -00001100_2(\text{прямой код}) = 1'1110011_2(\text{обратный код}) = 1'1110100(\text{дополнительный код})$$

Запись в памяти:

11110100

2. 55

$$55_{10} = 00110111_2(\text{прямой код})$$

Запись в памяти:

00110111

float

1. 12.5

Переведём 12.5 в двоичную систему счисления

$$12.5_{10} = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^{-1} = 1100.1_2$$

Полученное число приведём к форме $(-1)^s \cdot 1.m \cdot 2^e$

$$1100.1_2 = (-1)^0 \cdot 1.1001 \cdot 2^{11}$$

$$s = 0; m = 1001; e = 3$$

$$\text{Смещённый порядок } e: 3_{10} + 127_{10} = 130_{10} = 10000010$$

Запись в памяти:

01000001 01001000 00000000 00000000

2. -431.75

Переведём -431.75 в двоичную систему счисления

$$-431.75_{10} = -(1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^5 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}) = -110101111.11_2$$

Полученное число приведём к форме $(-1)^s \cdot 1.m \cdot 2^e$

$$-110101111.11_2 = (-1)^1 \cdot 1.101011111 \cdot 2^{1000}$$

$$s = 1; m = 101011111; e = 8$$

$$\text{Смещённый порядок } e: 8_{10} + 127_{10} = 135_{10} = 10000111$$

Запись в памяти:

11000011 11010111 11100000 00000000

{red, yellow, green} colors

1. red

red - первый элемент enum и ему не присвоено значение, а значит его значение будет равно значению по умолчанию - 0. Тип значения - int, значит под хранение элемента на большинстве современных ПК будет выделено 32 бит.

Запись в памяти:

00000000 00000000 00000000 00000000

2. green

green - третий элемент enum, ему не присвоено значение и элементам до него не было присвоено значений, в таком случае номера элементов присваиваются по порядку. green будет равен 2. Тип значения - int, значит под хранение элемента на большинстве современных ПК будет выделено 32 бит. Запись в памяти:

00000000 00000000 00000000 00000010

4. Преобразовать двоичный код в значение.

signed char

1. 11110100

Для перевода из дополнительного кода число необходимо инвертировать биты числа и прибавить 1 (алгоритм перевода числа из прямого кода в дополнительный аналогичен).

$$11110100 \rightarrow 00001011 \rightarrow 00001100 (\text{прямой код})$$

$$00001100_2 = -(1 \cdot 2^3 + 1 \cdot 2^2) = -12_{10}.$$

Значения до и после перевода совпали.

2. 00110111

Число уже находится в прямом коде, дополнительные действия выполнять не требуется.

$$00110111_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 55_{10}.$$

float

1. 01000001 01001000 00000000 00000000

$$s: 01000001010010000000000000000000 = 0$$

$$\text{Смещённый порядок: } 01000001 \ 010010000000000000000000 = 10000010_2 = 130_{10}$$

$$\text{Порядок } e: 130_{10} - 127 = 3$$

$$m: 0100000101001000 \ 00000000 \ 00000000 = 1001_2$$

$$(-1)^s \cdot 1.m \cdot 2^e = (-1)^0 \cdot 1.1001 \cdot 2^3 = 1 \cdot 1.1001 \cdot 2^3 = 1100.1_2$$

$$1100.1_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^{-1} = 12.5$$

Значения до и после перевода совпали.

2. 11000011 11010111 11100000 00000000

$$s: 11000011110101111110000000000000 = 1$$

$$\text{Смещённый порядок: } 11000011 \ 110101111110000000000000 = 10000111_2 = 135_{10}$$

$$\text{Порядок } e: 135_{10} - 127 = 8$$

$$m: 1100001111010111 \ 11100000 \ 00000000 = 101011111_2$$

$$(-1)^s \cdot 1.m \cdot 2^e = (-1)^1 \cdot 1.101011111 \cdot 2^8 = -1 \cdot 1.101011111 \cdot 2^8 = 110101111.11_2$$

$$-110101111.11_2 = -(1 \cdot 2^8 + 1 \cdot 2^7 + 1 \cdot 2^5 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}) = -431.75$$

Значения до и после перевода совпали.

{red, yellow, green} colors

Константам из colors будут соответствовать следующие значения:

red: 0

yellow: 1

green: 2

1. 00000000 00000000 00000000 00000000

00000000000000000000000000000000₂ = 0₁₀. Значению 0 соответствует константа red.

Значения до и после перевода совпали.

2. 00000000 00000000 00000000 00000010

00000000000000000000000000000010₂ = 2₁₀. Значению 2 соответствует константа green.

Значения до и после перевода совпали.

5. Разработать и отладить программу, выдающую двоичное представление значений, заданных СД.

alg.h

```
void PrintByte(unsigned char *a);

void PrintVar(void *a, unsigned int size);
```

task5.c

```
#include <stdio.h>

#include "../alg.h"

void PrintByte(unsigned char *a) {
    for (int bitIndex = 7; bitIndex >= 0; bitIndex--) {
        int bit = ((*a) >> bitIndex) & 1;
        printf("%d", bit);
    }
}

void PrintVar(void *a, unsigned int size) {
    for (int i = size - 1; i >= 0; i--) {
        PrintByte(a + i);
        printf(" ");
    }
}
```

Порядок хранимых данных в памяти ПК автора отчёта - обратный, поэтому для удобства функция PrintVar выводит байты в обратном порядке.

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

main.c

```

#include <stdio.h>

#include "../libs/alg/algc.h"

typedef enum {red, yellow, green} colors;

int main() {
    char charVarA = -12;
    PrintVar(&charVarA, sizeof(charVarA));
    printf("\n");
    char charVarB = 55;
    PrintVar(&charVarB, sizeof(charVarB));
    printf("\n");
    printf("\n");

    float floatVarA = 12.5;
    PrintVar(&floatVarA, sizeof(floatVarA));
    printf("\n");
    float floatVarB = -431.75;
    PrintVar(&floatVarB, sizeof(floatVarB));
    printf("\n");
    printf("\n");

    colors colorsVarA = red;
    PrintVar(&colorsVarA, sizeof(colorsVarA));
    printf("\n");
    colors colorsVarB = green;
    PrintVar(&colorsVarB, sizeof(colorsVarB));
    printf("\n");

    return 0;
}

```

Результат выполнения программы:

```

11110100
00110111

01000001 01001000 00000000 00000000
11000011 11010111 11100000 00000000

00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000010

```

Результаты программы совпали с ручными вычислениями.

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 В типа «массив байт».
3. Привести В к заданному типу. Вывести значение.
4. Конец.

alg.h

```
typedef enum
{
    red,
    yellow,
    green
} colors;

char strByteToSignedChar(char *in);

float strByteToFloat(char *in);

colors strByteToEnumColors(char *in);

char *strByteToByteArray(char *in);
```

task7.c

```
#include "../alg.h"

typedef enum
{
    red,
    yellow,
    green
} colors;

char strByteToSignedChar(char *in) {
    char *dat = strByteToByteArray(in);
    char val = *((char*)dat);

    free(dat);

    return val;
```

```

}

float strByteToFloat(char *in) {
    char *dat = strByteToByteArray(in);
    float val = *((float*)dat);

    free(dat);

    return val;
}

colors strByteToEnumColors(char *in) {
    char *dat = strByteToByteArray(in);
    colors val = *((colors*)dat);

    free(dat);

    return val;
}

char *strByteToByteArray(char *S) {
    int len = 0;
    int input;
    while (S[len] != '\0') {
        len++;
    }

    int bitSize = len;
    int byteSize = (bitSize) / 8 + ((bitSize) % 8 != 0);
    char *B = malloc(byteSize);
    for (int i = 0; i < bitSize; i++) {
        B[byteSize - 1 - (i / 8)] |= ((S[i] == '1') << (8 - 1 - (i % 8)) );
    }

    return B;
}

```

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

```

#include <stdio.h>
#include <stdlib.h>

```

```

#include <assert.h>


#include "../libs/alg/alg.h"

int main() {
    assert(strByteToSignedChar("11110100") == -12);
    assert(strByteToSignedChar("00110111") == 55);
    assert(strByteToFloat("010000010100100000000000000000") == 12.5);
    assert(strByteToFloat("110000111101011111000000000000") == -431.75);
    assert(strByteToEnumColors("0000000000000000000000000000") == red);
    assert(strByteToEnumColors("00000000000000000000000000010") == green);
    printf("Tests done!");

    return 0;
}

```

Результаты выполнения программы:



```

vlad@Mac-Pro-Vladislav build % /Users/vlad/Desktop/C/algorithms_and_data_structures/build/lab1_task8
Tests done!

```

Программа выполнялась успешно, ассерты не упали, следовательно перевод из двоичной системы был выполнен корректно.

Вывод: в ходе лабораторной работы изучили способы задания отношений, операции над отношениями и свойства отношений, научились программно реализовывать операции и определять свойства отношений.