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»

Выполнил: ст. группы ПВ-223
Игнатьев Артур Олегович

Проверил:
асс. Солонченко Роман Евгеньевич

Белгород 2023г.

# Лабораторная работа №1

# «Встроенные структуры данных»

**Цель работы:** изучение базовых типов данных языка С как структур данных (СД).

**Содержание отчета:**

- Тема лабораторной работы.
- Цель лабораторной работы.
- Условия задач и их решение.
- Выводы.

**Задание к лабораторной работе :**

Вариант №3

unsigned short

double

{winter, spring, summer, autumn} seeson

1. Для типов данных определить

(a) Абстрактный уровень представления СД

i. Характер организованности и изменчивости

• unsigned short: встроенный, статический, простейший.

• double: встроенный, статический, простейший.

• {winter, spring, summer, autumn} seeson: встроенный, статический, сложный.

ii. Набор допустимых операций

• unsigned short:

Операции сравнения, инкрементирование, декрементирование, сложение, вычитание, умножение, деление, унарные операции, взятие адреса, логические операции, операции присваивания, приведение типов.

• double:

Операции сравнения, инкрементирование, декрементирование, сложение, вычитание, умножение, деление, унарные операции, взятие адреса, логические операции, операции присваивания, приведение типов.

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

Операция доступа и присваивания.

(b) Физический уровень представления СД

i. Схема хранения

• unsigned short: последовательная.

• double: последовательная.

• {winter, spring, summer, autumn} seeson: последовательная.

ii. Объем памяти, занимаемый экземпляром СД

• unsigned short: 2 байта.

• double: 8 байта.

• {winter, spring, summer, autumn} seeson: 4 * 1 = 4 байта.

iii. Формат внутреннего представления СД и способ его

интерпретации

• unsigned short: 16 бит, старший бит не отводиться под знак, все 16 бит отводится под значение.

• double: 64 бита, старший бит отводиться под знак, следующие 11 под величину целой части, а оставшиеся под мантиссу.

• {winter, spring, summer, autumn} seeson: -.

iv. Характеристика допустимых значений

• unsigned short: [0; 65535].

• double: [2.22507E–308; 2.225E+308].

• {winter, spring, summer, autumn} seeson: -.

v. Тип доступа к элементам

• unsigned short: прямой.

• double: прямой.

• {winter, spring, summer, autumn} seeson: прямой, по имени поля.

(c) Логический уровень представления СД

i. Способ описания СД и экземпляра СД на языке программирования

• unsigned short:

```
unsigned short/*имя переменной*/;
```

•double:

```
double /*имя переменной*/;
```

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

```
enum {/* именованная константа */} /* имя переменной */ ;
```

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

```
int main() {

    unsigned short a = 25 ;
    double b = 125.25 ;
    enum {winter, spring, summer, autumn }season;

    return 0;
}
```

3. Преобразовать значения в двоичный код

- a = 25

  $25_{10} = 0000000000011001_2$

- b = 125,25

  $125,25_{10} = 00000000000000000000000000000000000000000000000000011111 01.01_2$

  $125_{10} = 1111101_2$

  $0.25_{10} = 01_2$

  0.25 * 2 = 0.5

  0.5 * 2 = 1

4. Преобразовать двоичный код в значение

- $11001_2 = (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) = 16 + 8 + 0 + 0 + 1 = 25_{10}$
- $1111101.01_2 = (1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) = 64 + 32 + 16 + 8 + 4 + 0 + 1 + 0 + 0.25 = 125.25_{10}$

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

```
// Функция для вывода двоичного представления unsigned char
void printByte(unsigned char a) {
    // Создаем маску, начиная с самого старшего бита
    unsigned char mask = 0x80; // 0x80 представляет старший бит
    unsigned char t = a;

    // Используем цикл для обхода битов переменной
    for (size_t i = 0; i < 8; i++) {
        // Проверяем текущий бит и выводим 0 или 1
        printf("%d", (t & mask) ? 1 : 0);

        t <<= 1;
    }
}

// Функция для вывода двоичного представления переменной произвольного типа
void printVar(void *a, unsigned int size) {
    unsigned char *bytePtr = (unsigned int *) a; // Приводим указатель к типу
"массив байт"

    // Используем цикл для обхода байтов переменной
    for (int i = size - 1; i >= 0; i--)
        printByte(bytePtr[i]);

    bytePtr = NULL;
    printf("\n");
}
```

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

```
int main() {

    unsigned short a = 25;
    double b = 125.25;
    enum {
        winter, spring, summer, autumn
    } season;

    printf("a: ");
    printVar(&a, sizeof(unsigned short));

    printf("b: ");
    printVar(&b, sizeof(double));

    int summers = summer;
    printf("summers: ");
    printVar(&summers, sizeof(int));

    return 0;
}
```

```
C:\Users\NTK\CLionProjects\ASDLab1\cmake-build-debug\ASDLab1.exe
a: 0000000000011001
b: 01000000101111101010000000000000000000000000000000000000000000000
summers: 00000000000000000000000000000010

Process finished with exit code 0
```

7. Разработать и отладить программу, определяющую значение переменной по ее двоичному представлению

```
#define BIT 8

// Функция transformToByte конвертирует строку битов (представленных как символы '0' и '1')
// в беззнаковое целое число типа unsigned char. Возвращает результат.
unsigned char transformToByte(const char *str) {
    unsigned char result = 0;
    int currentBit = 0;

    // Проходим по строке до тех пор, пока не пройдем все биты или не достигнем конца строки.
    while (currentBit < BIT && *str != '\0') {
        // Сдвигаем текущий результат влево на 1 бит и добавляем новый бит из строки.
        result = (result << 1) | (*str - '0');
        currentBit++, str++;
    }
    return result;
}

// Функция extractionBytes извлекает последовательности бит из строки и конвертирует их в
// беззнаковые байты, сохраняя результат в выходном массиве 'out'. 'size' определяет
// количество байт, которые нужно прочитать из строки.
void extractionBytes(const char *str, const unsigned size, char *out) {
    char buffer[BIT + 1];
    buffer[BIT] = '\0';

    const char *end = str + strlen(str);
    const char *begin = end - BIT;

    int currentByteIndex = 0;

    // Пока не закончилась строка и не достигнуто требуемое количество байт.
    while (end > str && currentByteIndex < size) {
        // Если начало окна выходит за границы строки, то устанавливаем его в начало строки.
        if (begin < str) {
            begin = str;
        }

        // Копируем последовательность бит из строки в буфер.
        memcpy(buffer, begin, end - begin);
```

```
        buffer[end - begin] = '\0';

        // Конвертируем буфер в беззнаковый байт и сохраняем его в выходном
массиве.
        out[currentByteIndex++] = transformToByte(buffer);

        // Сдвигаем начало и конец окна на размер одного байта (BIT битов).
        begin -= BIT;
        end -= BIT;
    }
}


// Функция transformToInt конвертирует строку байтов в целое число типа int.
// Она извлекает байты из строки и объединяет их в int, затем возвращает ре-
зультат.
int transformToInt(const char *str) {
    char bytes[sizeof(int)];

    // Извлекаем байты из строки и сохраняем их в массив bytes.
    extractionBytes(str, sizeof(int), bytes);

    // Преобразуем массив байтов в int и возвращаем результат.
    return *((int *) bytes);
}

// Функция transformToUnsignedShort конвертирует строку байтов в беззнаковое
короткое
// целое число типа unsigned short. Она извлекает байты из строки и объеди-
няет их в unsigned short,
// затем возвращает результат.
unsigned short transformToUnsignedShort(const char *str) {
    char bytes[sizeof(unsigned short)];

    // Извлекаем байты из строки и сохраняем их в массив bytes.
    extractionBytes(str, sizeof(unsigned short), bytes);

    // Преобразуем массив байтов в unsigned short и возвращаем результат.
    return *((unsigned short *) bytes);
}

// Функция transformToDouble конвертирует строку байтов в число с плавающей
запятой
// двойной точности типа double. Она извлекает байты из строки и объединяет
их в double,
// затем возвращает результат.
double transformToDouble(const char *str) {
    char bytes[sizeof(double)];

    // Извлекаем байты из строки и сохраняем их в массив bytes.
    extractionBytes(str, sizeof(double), bytes);

    // Преобразуем массив байтов в double и возвращаем результат.
    return *((double *) bytes);
}
```

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

```
int main() {

    char a2[] = "0000000000011001";
    char b2[] =
"01000000101111101010000000000000000000000000000000000000000000000";
    char summers2[] = "00000000000000000000000000000010";

    unsigned short transformed_a2 = transformToUnsignedShort(a2);
    double transformed_b2 = transformToDouble(b2);
    int transformed_summers2 = transformToInt(summers2);

    printf("a2: %s: %ld\n", a2, transformed_a2);
    printf("b2: %s: %f\n", b2, transformed_b2);
    printf("summers2:  %s: %d\n", summers2, transformed_summers2);

    return 0;
}
```

```
C:\Users\NTK\CLionProjects\ASDLab1\cmake-build-debug\ASDLab1.exe
a2: 0000000000011001: 25
b2: 01000000101111101010000000000000000000000000000000000000000000000: 125.250000
summers2:  00000000000000000000000000000010: 2

Process finished with exit code 0
```

Вывод: В ходе выполнения лабораторной работы были изучены базовые типы данных как структур данных, были переведены значения чисел из десятичной системы счисления в двоичную систему счисления и наоборот. Значения совпадают с теми, которые были получены в результатах работы программы.