Задания на самостоятельное выполнение

1 Задание №0

Разработать программу, обеспечивающую вывод на экран следующих сообщений:

```
Hello, world!
Здравствуй, мир!
```

2 Задание №1

Разработать программу, обеспечивающую перевод температуры по Фаренгейту в температуру по Цельсию.

Формула перевода температуры по Фаренгейту в температуту по Цельсию:

$$C^o = \frac{5}{9} * (F^o - 32) \tag{1}$$

Программа должна обеспечивать:

- выдачу следующего сообщения для ввода значения температуры в градусах Фаренгейта:

  Enter Farenheit:

  (для выдачи сообщений использовать функцию **printf(...)**);

- ввод значения температуры в градусах Фаренгейта в виде вещественного числа (для ввода значения использовать функцию **scanf(...)**);

- выдачу следующего сообщения с вычисленным значением температуры в градусах Цельсия:

  Celsii: X

  где X — вычисленное значение температуры в градусах Цельсия.

Каждое выдаваемое сообщение должно начинаться с новой строки.

3 Задание №2

Используя результаты разработанной в рамках задания 1 программы, реализовать программу, обеспечивающую выдачу таблицы соответствия температур в градусах Цельсия температурам в градусах Фаренгейта в диапазоне значений от $0F^o$ до $150F^o$ с шагом $10F^o$.

Таблица должна состоять из двух столбцов. Каждый столбец должен иметь ширину 12 символов. В первом столбце выводится значение F^o, во втором — соответствующее значение C^o. Столбцы должны быть выровнены по вертикали. Значения выводить с точностью до двух знаков после десятичной точки.

Пример таблицы:

```
 0.00       17.77
10.00       12.22
.............
```

4 Задание №3

Разработать программу, обеспечивающую выдачу таблицы соответствия температур в градусах Цельсия температурам в градусах Фаренгейта в диапазоне значений от $0F^o$ до $150F^o$ с шагом $10F^o$.

Таблица должна состоять из двух столбцов. В первом столбце выводится значение F^o, во втором — соответствующее значение C^o. Столбцы должны быть выровнены по вертикали. Значения выводить с точностью до двух знаков после десятичной точки.

Пример таблицы (знак ␣ означает знак пробела):

```
+-----------+------------+
|␣␣␣␣␣F␣␣␣␣␣|␣␣␣␣␣C␣␣␣␣␣␣|
+-----------+------------+
|␣␣␣␣␣␣␣0.00|␣␣␣␣␣␣␣17.77|
|␣␣␣␣␣␣10.00|␣␣␣␣␣␣␣12.22|
|␣␣␣␣␣␣.....|␣␣␣␣␣␣␣.....|
+-----------+------------+
```

Реализовать три варианта программы:

1. с использованием цикла **while**;
2. с использованием цикла **do-while**;
3. с использованием цикла **for**.

Варианты должны быть реализованы в одном и том же исходном модуле.

Рекомендация: для вывода результата использовать функцию стандартной библиотеки printf().

5 Задание №4

Разработать программу, обеспечивающую получение последовательности $x_1, x_2, \ldots, x_{20}$, образованной по следующему закону:

$$x_1 = 0; x_2 = \frac{5}{8}; x_i = \frac{x_{i-1}}{2} + \frac{3}{4}x_{i-2}, i = 3, 4, \ldots, 20$$

Результат работы программы должен быть представлен в форме таблицы следующего вида (знак ␣ означает знак пробела):

```
x01␣=␣␣␣␣␣0.000
x02␣=␣␣␣␣␣0.625
.................
x20␣=␣.........
```

Столбцы таблицы должны быть выровнены по вертикали. Значения выводить с точностью до трёх знаков после десятичной точки.

Реализовать три варианта программы:

1. с использованием цикла **while**;
2. с использованием цикла **do-while**;
3. с использованием цикла **for**.

Варианты должны быть реализованы в одном и том же исходном модуле.

Рекомендация: для вывода результата использовать функцию стандартной библиотеки printf().

6 Задание №5

Разработать программу, обеспечивающую ввод с экрана натурального значения n, последовательности целых чисел $a_1, a_2, \ldots, a_n$ и, далее, вычисление суммы положительных и количества отрицательных членов последовательности $a_1, a_2, \ldots, a_n$.

Сценарий работы программы должен выглядеть следующим образом (знак ␣ означает знак пробела):

```
SOURCE␣DATA:
Enter␣natural␣value:
n␣=␣...

Enter␣sequence:
a01␣=␣...
...

RESULT:
Sum␣of␣positive␣elements␣␣␣:␣...
Number␣of␣negative␣elements:␣...
```

Реализовать три варианта программы:

1. с использованием цикла **while**;
2. с использованием цикла **do-while**;
3. с использованием цикла **for**.

Варианты должны быть реализованы в одном и том же исходном модуле.
Рекомендации:

- для ввода данных использовать функцию стандартной библиотеки scanf();
- для вывода данных использовать функцию стандартной библиотеки printf().

7 Задание №6

Разработать программу, обеспечивающую побайтовую печать значения целой переменной.

Дано:

```
int a = 0x01020304;
```

Получить:

- целое значение первого байта памяти, занимаемой переменной a;
- целое значение второго байта памяти, занимаемой переменной a;
- целое значение третьего байта памяти, занимаемой переменной a;
- целое значение четвёртого байта памяти, занимаемой переменной a.

Формат вывода:

```
Byte␣1:␣...
Byte␣2:␣...
Byte␣3:␣...
Byte␣4:␣...
```

Значения вывести в шестнадцатиричном формате в виде NN, где N — шестнадцатиричная цифра. То есть выводимый результат должен занимать ровно две позиции. Если значение результата представляется одной позицией, то должен быть напечатан ведущий 0. Например, значение `9` должно быть напечатано в виде `09`.

Для вывода значений в шестнадцатеричном виде используется спецификация формата `%x`.

Получить десятичное значение шестнадцатиричного числа 0x01020304.

Дополнить программу следующими действиями::

1. `int a = ...;`
   где вместо многоточия должно быть подставлено полученное десятичное число.

2. Получить:

   - целое значение первого байта памяти, занимаемой переменной a;
   - целое значение второго байта памяти, занимаемой переменной a;
   - целое значение третьего байта памяти, занимаемой переменной a;
   - целое значение четвёртого байта памяти, занимаемой переменной a.

   Выполнить программу для данного случая.
   Ответить на вопрос: изменился ли результат?

8 Задание №7

Разработать и реализовать функцию, обеспечивающую сортировку элементов целочисленного массива. Спецификация функции:

```
int mysort(int array[], int n);
```

Здесь:

- `array` — имя массива для сортировки;
- `n` — количество элементов массива.

В случае успешного выполнения функция `mysort()` должна возвращать значение `0`, иначе — значение `1`. Например, если значение параметра `n` меньше или равно `0`, то функция должна вернуть в качестве результата значение `1`.

Разработать и реализовать функцию `main`, обеспечивающую объявление и заполнение целочисленного массива srcarray значениями, а также обеспечивающую вывод отсортированного массива. Тестирование проводить на массиве с количеством элементов не меньше 10 и не больше 99.

Вывод результата должен быть представлен в следующем виде (пример для массива из 12 элементов):

```
SORTED␣ARRAY:
srcarray[01]␣=␣...
srcarray[02]␣=␣...
...
srcarray[12]␣=␣...
```

Выбрать метод сортировки по собственному усмотрению.
Выполнить следующию требования:

- функция `main` должна быть реализована в файле с именем `test07.c`;
- функция `mysort` должна быть реализована в файле с именем `mysort.c`;
- описание функции `mysort` должно быть реализовано в файле с именем `mysort.h`;
- в файле `test07.c` в соответствующем месте должна быть строка `#include "mysort.h"`.

Рекомендуемая литература для выбора алгоритма сортировки:

- Никлас Вирт "Алгоритмы и структуры данных";
- Дональд Кнут "Искусство программирования на ЭВМ", том 3 "Сортировка и поиск".

9 Задание №8

1. Разработать и реализовать функции, обеспечивающие:
   - определение длины строки символов;
   - лексикографическое сравнение двух строк символов;
   - обмен значениями двух строк символов.
2. Разработать и реализовать функцию `main()`, обеспечивающую тестирование работы функций обработки строк символов, указанных выше.

9.1 Определение длины строки символов

Спецификация функции:

```
int stringlength(char *string);
```

Функция возвращает в качестве результата длину в байтах строки `string`.

9.2 Лексикографическое сравнение строк символов

Спецификация функции:

```
int stringcompare(char *string01, char *string02);
```

Функция возвращает в качестве результата одно из следующих значений:

- 0 — если стоки `string01` и `string02` лексикографически равны;
- -1 — если строка `string01` лексикографически меньше строки `string02`;
- 1 — если строка `string01` лексикографически больше строки `string02`.

Понятие лексикографического порядка означает, что строка `string01` меньше строки `string02` , если первые m символов строк совпадают, а символ $m+1$ строки `string01` меньше символа $m+1$ строки `string02`.

9.3 Обмен значениями двух строк символов

Спецификация функции:

```
int stringswap(char *string01, char *string02);
```

Функция обеспечивает обмен содержимого строк `string01` и `string02`. Функция возвращает в качестве результата значение 0.

Требования к реализации функции main()

Функция `main()` должна обеспечить возможность ввода исходных данных и выдачу резуальтатов в соответствии со следующим сценарием:

```
Enter␣string␣for␣stringlength:

stringlength␣result:

***

Enter␣string01␣for␣stringcompare:

Enter␣string02␣for␣stringcompare:

stringcompare␣result:

***

Enter␣string01␣for␣stringswap:

Enter␣string02␣for␣stringswap:

string01␣after␣stringswap:
string02␣after␣stringswap:
```

9.4 Требования к выполнению задания

1. Вводимое количество символов для отдельной строки не будет превышать значения 80 (без учёта завершающего нуль–символа).
2. Функции обработки строк символов должны быть реализованы в файле с именем `mystring.c`.
3. Описания реализованных функций обработки строк символов должны быть размещены в файле `mystring.h`.
4. Реализация функции `main()` должна быть размещена в файле `test08.c`.

5. При реализации функций обработки строк символов (`stringlength()`, `sringcompare()`, `stringswap()`) **ЗАПРЕЩАЕТСЯ ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ СТАНДАРТНОЙ БИБЛИОТЕКИ ФУНКЦИЙ СИСТЕМЫ ПРОГРАММИРОВАНИЯ C**.

9.5 Используемые функции ввода/вывода

Для ввода строк символов с экрана можно использовать одну из следующих двух функций стандартной библиотеки системы программирования C:

- gets() — считывание строки символов с экрана до перехода на новую строку;
- scanf() — форматный ввод; для ввода строки символов использовать спецификацию формата `%s`.

Для вывода строк символов на экран можно использовать одну из следующих двух функций стандартной библиотеки системы программирования C:

- puts() — вывод строки символов на экран;
- printf() — форматный вывод; для вывода строки символов использовать спецификацию формата `%s`.

10 Задание №9

Разработать и реализовать функцию, обеспечивающую сортировку в лексикографическом порядке строк символов.

Спецификация функции:

```
int mysortstr01(char array[][81], int n);
```

Здесь:

- `array` — имя массива строк символов для сортировки;
- `n` — количество элементов массива.

Разработать и реализовать функцию `main`, обеспечивающую объявление и заполнение массива строк символов значениями, а также обеспечивающую вывод отсортированного массива.

Тестирование проводить на массиве с количеством строк не меньше 10.

Вывод результата должен быть представлен в следующем виде (пример):

```
SORTED␣STRINGS:
string[01]␣=␣...
string[02]␣=␣...
...
```

Здесь:

- `string[01]` — первый элемент массива (элемент с индексом `0`);
- `string[02]` — второй элемент массива (элемент с индексом `1`);
- и т.д.

Выбрать метод сортировки по собственному усмотрению. Допускается использовать метод сортировки, реализованный в рамках задания №7.

Выполнить следующие требования:

- длина отдельной строки символов не должна превышать значения 80 (без учёта завершающего нуль–символа);
- при реализации функций **ЗАПРЕЩАЕТСЯ ИСПОЛЬЗОВАНИЕ ФУНКЦИЙ СТАНДАРТНОЙ БИБЛИОТЕКИ СИСТЕМЫ ПРОГРАММИРОВАНИЯ С**. Использование функций стандартной библиотеки разрешается только при реализации функции `main()`;
- при реализации функции `mysortstr01` использовать функции работы со строками символов, реализованные в рамках задания №8 и, соответственно, при сборке программы тестирования, необходимо использовать уже существующий файл `mystring.c` .
- функция `main` должна быть реализована в файле с именем `test09.c`;
- функция `mysortstr01` должна быть реализована в файле с именем `mysortstr.c`;
- описание функции `mysortstr01` должно быть реализовано в файле с именем `mysortstr.h`;
- в качестве результата выполнения работы должны быть предоставлены следующие файлы:
  - `mysortstr.c` ;
  - `mysortstr.h` ;
  - `mystring.c` ;
  - `mystring.h` ;
  - `test09.c` .

Рекомендуемая литература для выбора алгоритма сортировки:

- Никлас Вирт “Алгоритмы и структуры данных”;
- Дональд Кнут “Искусство программирования на ЭВМ”, том 3 “Сортировка и поиск”.