ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Итоги теста №3

СТРОКИ И ФУНКЦИИ

Подготовила Остроухова С.А.

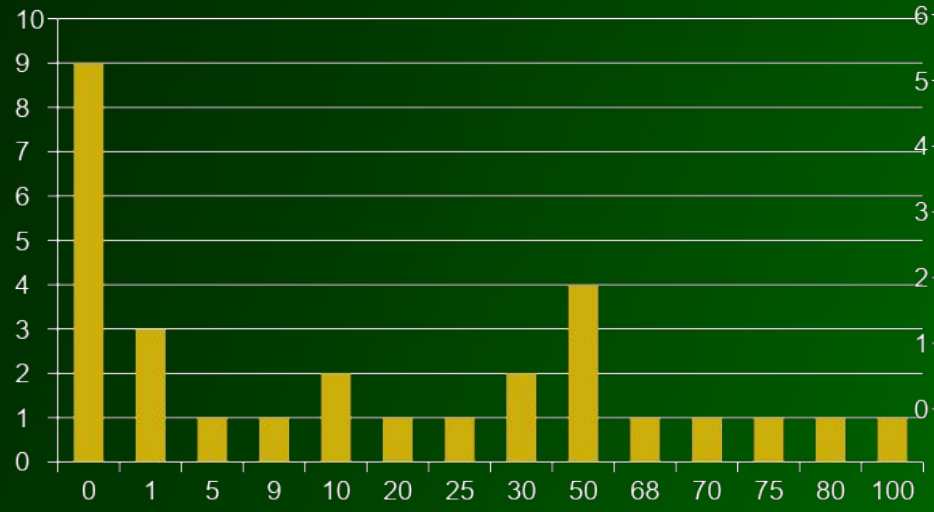
Итоги

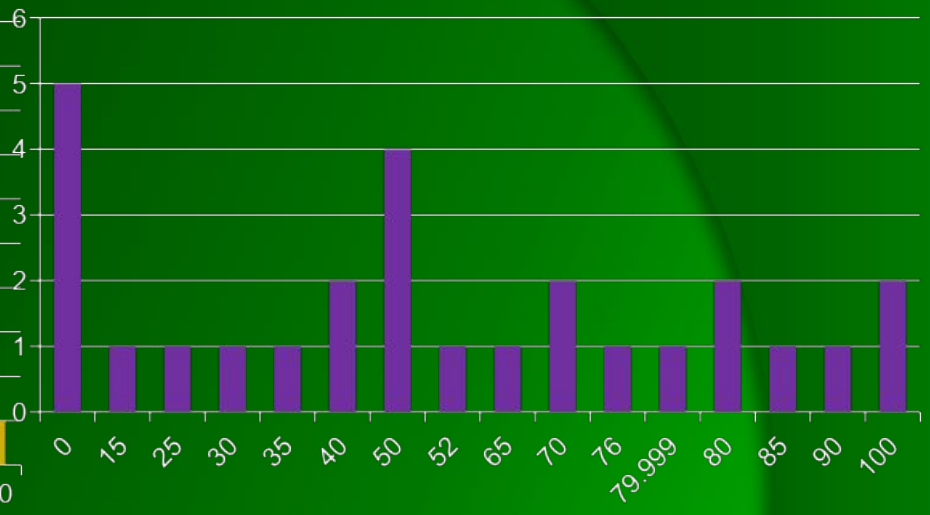
	650502	650503
Написано работ	29/30	29/30
Минимальная оценка	0	0
Максимальная оценка (11)	10,5	8,5
Средний балл	4,05	3,34

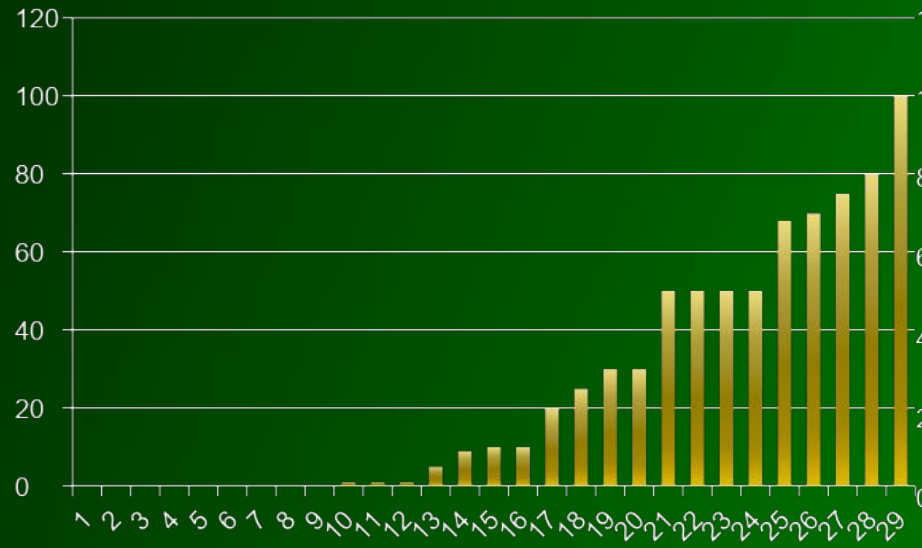
Итоги

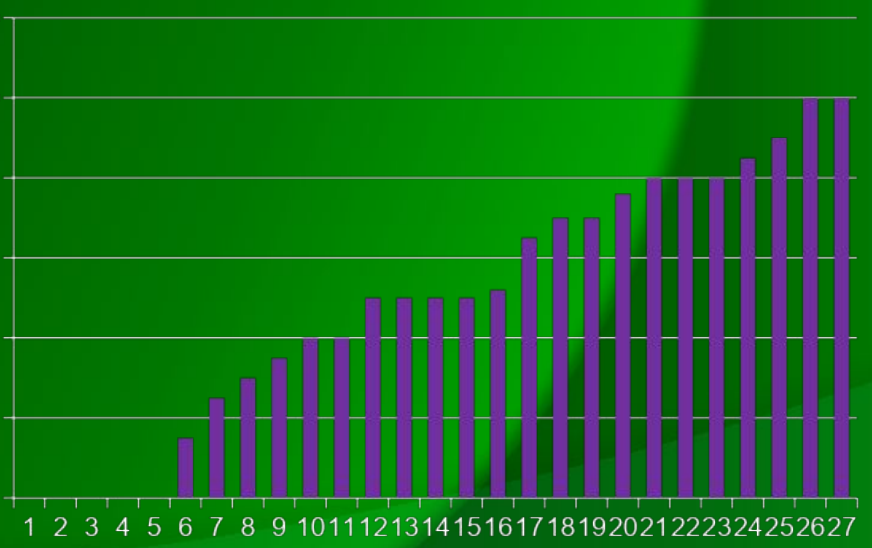
	650502	650503
Написано работ	29 / 30	29/30
За индивидуальную работу	21 (72%)	22 (79%)
За командную работу	8 (28%)	6 (21%)
Средняя польза от блок-схемы соседа	25,34%	49,37%

Итоги


10
9
8
7
6
5
4
3
2
1
0
0 1 5 9 10 20 25 30 50 68 70 75 80 100


6
5
4
3
2
1
0
0 15 25 30 35 40 50 52 65 70 76 79,999 80 85 90 100


120
100
80
60
40
20
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29


120
100
80
60
40
20
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

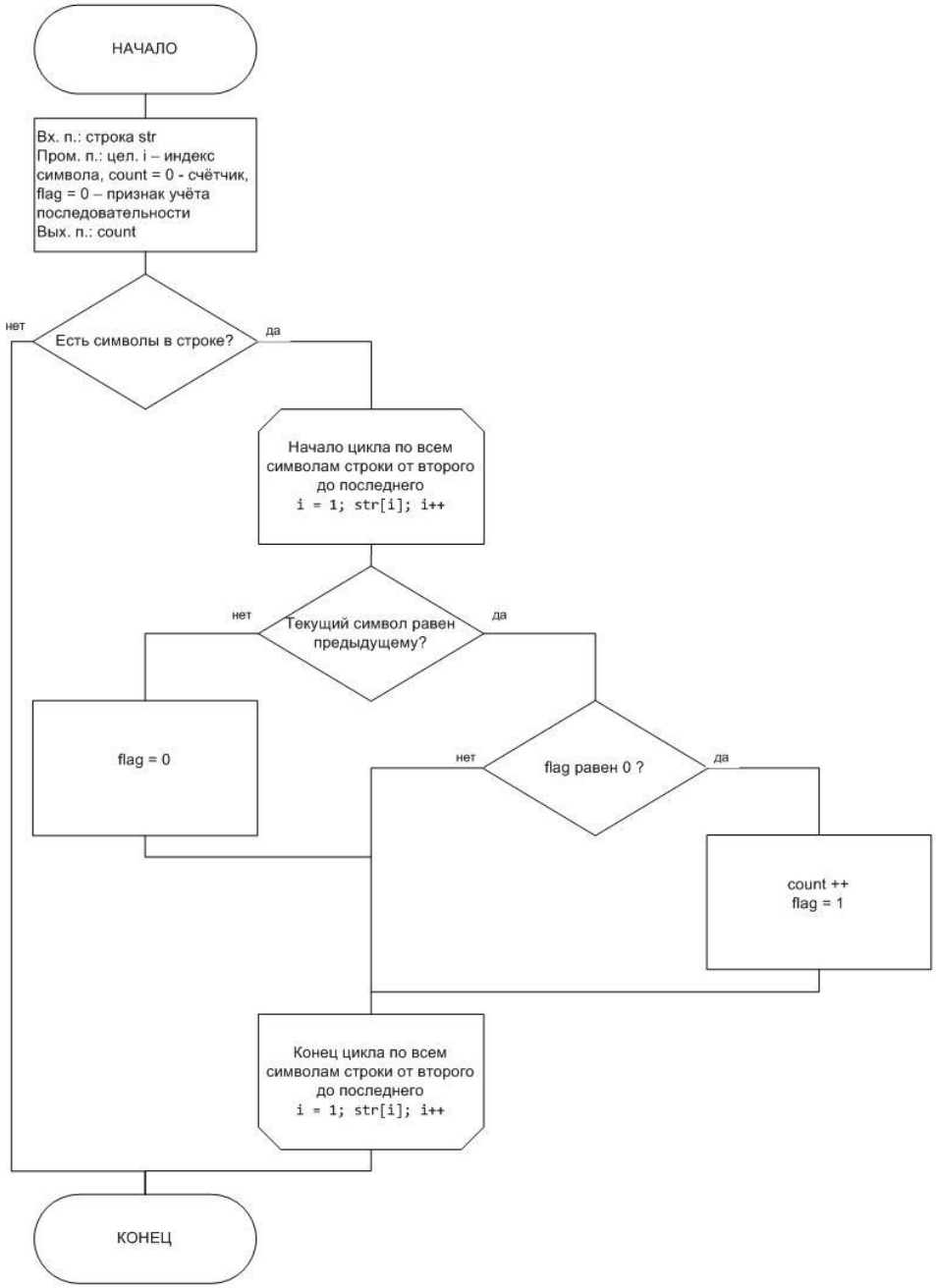
Типичные ошибки

- Определение длины строки.
- Цикл перебора строки посимвольно.
- Является ли символ цифрой.
- Зануление символов строки.

Вариант 1. Функция, которая…

- принимает указатель на строку
- возвращает количество последовательностей одинаковых символов, идущих подряд (целое число)
- считает количество последовательностей одинаковых символов, идущих подряд.

```
Счёт количества последовательностей одинаковых символов, идущих подряд (два или более)

int nSequences (char *str)
```



```c
//Счёт количества последовательностей одинаковых символов, идущих подряд (два или более)
int nSequences (char *str)
{
    int i, count = 0, flag = 0;

    if(str && *str)                     // строка не пуста и есть символы, кроме конца строки
    {
        for(i = 1; *(str+i); i++)       // i=1, так как сравниваем с предыдущим символом
            if(*(str+i) == *(str+i-1))  // одинаковые символы рядом
            {
                if(!flag)               // ещё не считали эту последовательность
                {
                    count++;            // посчитать
                    flag = 1;           // отметить, что посчитали
                }
            }                           // Скобка, чтобы показать, что ELSE относится к первому IF
            else
                flag = 0;               // сбросить флаг для новой последовательности
    }

    return count;
}
```

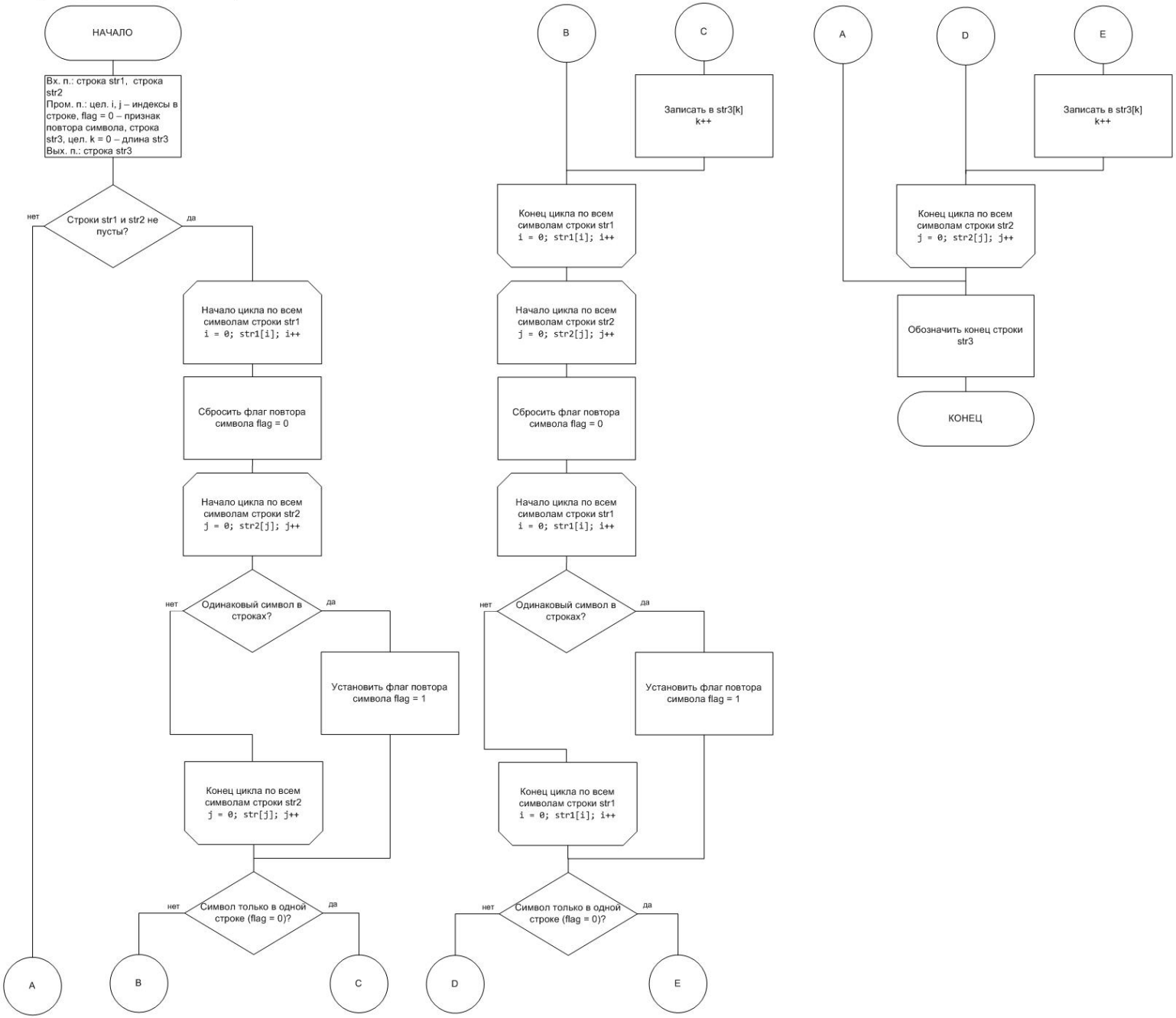
```c
#include<stdio.h>

int nSequences (char *str);

int main(int argc, char **argv)
{
    int i;

    for(i = 1; i < argc; i++)
    {
        puts(*(argv+i));
        printf("Sequences: %d\n\n", nSequences(*(argv+i)));
    }

    return 1;
}
```

```
aaaaa cdae aa bbb
Sequences: 3

test
Sequences: 0

Для продолжения нажмите любую клавишу
```

Вариант 2. Функция, которая…

- принимает указатели на две строки
- возвращает указатель на строку-результат
- формирует третью строку из символов, которые встречаются только в одной из строк (в том числе повторяющиеся). Например, если заданные строки *процессор* и *информация*, то ответом должно быть: *пессифмаия*.

```
Объединение в третью строку символов, которые встречаются только в одной из строк (в том
числе повторяющиеся)

char* joinUniq  (char *str1, char *str2)
```



еть полностью

```c
/*Формирует третью строку из символов, которые встречаются только в одной из строк (в том числе повторяющиеся).
Например, если для строк "процессор" и "информация" ответом должно быть "пессифмаия".*/
char* joinUniq (char *str1, char *str2)
{
    int i, j, k = 0, flag;
    char *str3 = (char *)malloc(STR_LEN * 2 * sizeof(char)); // Двойная длина
    if(!str3) return NULL;

    if(str1 && str2)                                // Указатели не в NULL
    {
        for(i = 0; *(str1+i); i++)                  // Проход по первой строке
        {
            for(flag = 0, j = 0; *(str2+j); j++)    // Проход по второй строке
            {
                if(*(str1+i) == *(str2+j))          // Символ есть в обеих строках
                {
                    flag = 1;                       // Символ найден
                    break;                          // Больше не искать его
                }
            }
            if(!flag) *(str3+k++) = *(str1+i);      // Символ только в первой строке, записать в результат
        }

        for(j = 0; *(str2+j); j++)                  // Проход по второй строке
        {
            for(flag = 0, i = 0; *(str1+i); i++)    // Проход по первой строке
            {
                if(*(str1+i) == *(str2+j))          // Символ есть в обеих строках
                {
                    flag = 1;                       // Символ найден
                    break;                          // Больше не искать его
                }
            }
            if(!flag) *(str3+k++) = *(str2+j);      // Символ только во второй строке, записать в результат
        }
    }
    *(str3+k) = 0;                                  // Обозначить конец строки

    return str3;
}
```

38 строк

```c
int joinUniqHelper (char *str1, char *str2, char *str3)
{
    int len3 = 0, i, j, flag;

    for(i = 0; *(str1+i); i++)                      // Проход по первой строке
        {
            for(flag = 0, j = 0; *(str2+j); j++)    // Проход по второй строке
            {
                if(*(str1+i) == *(str2+j))          // Символ есть в обеих строках
                {
                    flag = 1;                       // Символ найден
                    break;                          // Больше не искать его
                }
            }
            if(!flag) *(str3+len3++) = *(str1+i);   // Символ только в первой строке, записать в результат
        }

    return len3;
}


char* joinUniq2 (char *str1, char *str2)
{
    int k = 0;
    char *str3 = (char *)malloc(STR_LEN * 2 * sizeof(char)); // Двойная длина
    if(!str3) return NULL;

    if(str1 && str2)                                // Указатели не в NULL
    {
        k = joinUniqHelper(str1, str2, str3+k);
        k += joinUniqHelper(str2, str1, str3+k);
    }
    *(str3+k) = 0;                                  // Обозначить конец строки

    return str3;
}
```

36 строк

```c
#define STR_LEN 100

char* joinUniq (char *str1, char *str2);
char* joinUniq2 (char *str1, char *str2);

int main(int argc, char **argv)
{
    int i;
    char *result; //Память выделится в функции

    // i < argc-1, потому что по 2 строки
    // если последний набор не полный, не обрабатывать
    for(i = 1; i < argc-1; i+=2)
    {
        puts(*(argv+i));
        puts(*(argv+i+1));
        result = joinUniq(*(argv+i), *(argv+i+1)); //joinUniq2(*(argv+i), *(argv+i+1));
        printf("Result: %s\n\n", result);

        free(result); //Освободить память
    }

    return 1;
}
```

```
processor
info
Result: prcessrinf

test test
t    loop
Result: esesloop

abc
def
Result: abcdef

Для продолжения нажмите любую клавишу
```

Вариант 3. Функция, которая…

- принимает указатель на строку, целое число
- возвращает указатель на строку-результат
- формирует третью строку из символов, которые встречаются заданное количество раз.

```c
//Формирует новую строку из символов, которые встречаются заданное количество раз
char* findSymbols (char *str, int n)
{
    int i, j, wasFound, times, k = 0;
    char *str2 = (char *)malloc(STR_LEN * sizeof(char));
    if(!str2) return NULL;                                  // проверка выделения памяти

    for(i = 0; *(str+i); i++)                               // по всей строке
    {
        for(times = 1, wasFound = 0, j=0; j < i; j++)       // просмотр до текущей позиции
            if(*(str+i) == *(str+j))                        // если такой символ был раньше, ...
            {
                wasFound = 1;                               // ... то его уже обработали
                break;
            }

        if(!wasFound)                                       // символ найден впервые
            for(j=i+1; *(str+j); j++)                       // пройти до конца строки
                if(*(str+i) == *(str+j))
                    times++;                                // посчитать количество

        if(times == n)                                      // количество искомое
            *(str2 + k++) = *(str+i);
    }

    *(str2+k) = 0;
    str2 = (char*)realloc (str2, (k+1)*sizeof(char));

    return str2;
}
```

```c
//Для длинной строки, экономия памяти
char* findSymbols2 (char *str, int n)
{
    int i, j, times, k = 0;
    char *str2 = (char *)calloc(STR_LEN, sizeof(char));
    if(!str2) return NULL;                          // проверка выделения памяти

    for(i = 0; i < 256; i++)                        // проверить все символы
    {
        for(times = 0, j = 0; *(str+j); j++)                    // по всей строке
            if(*(str+j) == i)                       // нужный символ
                times++;                            // посчитать

        if(times == n)                              // количество искомое
            *(str2 + k++) = i;
    }

    *(str2+k) = 0;
    str2 = (char*)realloc (str2, (k+1)*sizeof(char));

    return str2;
}
```

```c
//Для длинной строки, производительность
char* findSymbols3 (char *str, int n)
{
    int i, j, times, k = 0;
    int symbolsCounter[256] = {0};

    char *str2 = (char *)calloc(STR_LEN, sizeof(char));
    if(!str2) return NULL;                                  // проверка выделения памяти

    for(i = 0; *(str+i); i++)                               // по всей строке
        symbolsCounter[ *(str+i) ]++;                               // посчитать

    for(i = 0; i < 256; i++)                                // проверить все символы
        if(symbolsCounter[i] == n)                          // количество искомое
            *(str2 + k++) = i;

    *(str2+k) = 0;
    str2 = (char*)realloc (str2, (k+1)*sizeof(char));

    return str2;
}
```

```c
#define STR_LEN 100

char* findSymbols (char *str, int n);
char* findSymbols2 (char *str, int n);
char* findSymbols3 (char *str, int n);

int my_atoi(char *str);

int main(int argc, char **argv)
{
    int i, n;
    char* result;

    for(i = 1; i < argc-1; i+=2)
    {
        puts(*(argv+i));
        n = my_atoi(*(argv+i+1));
        printf("%d\n", n);
        result = findSymbols(*(argv+i), n);
        printf("%s.\n\n", result);

        free(result);
    }

    return 1;
}
```

```
a loop abc ao
3
a o.

1234561
1
23456.
```

```
a loop abc ao
3
 ao.

1234561
1
23456.
```

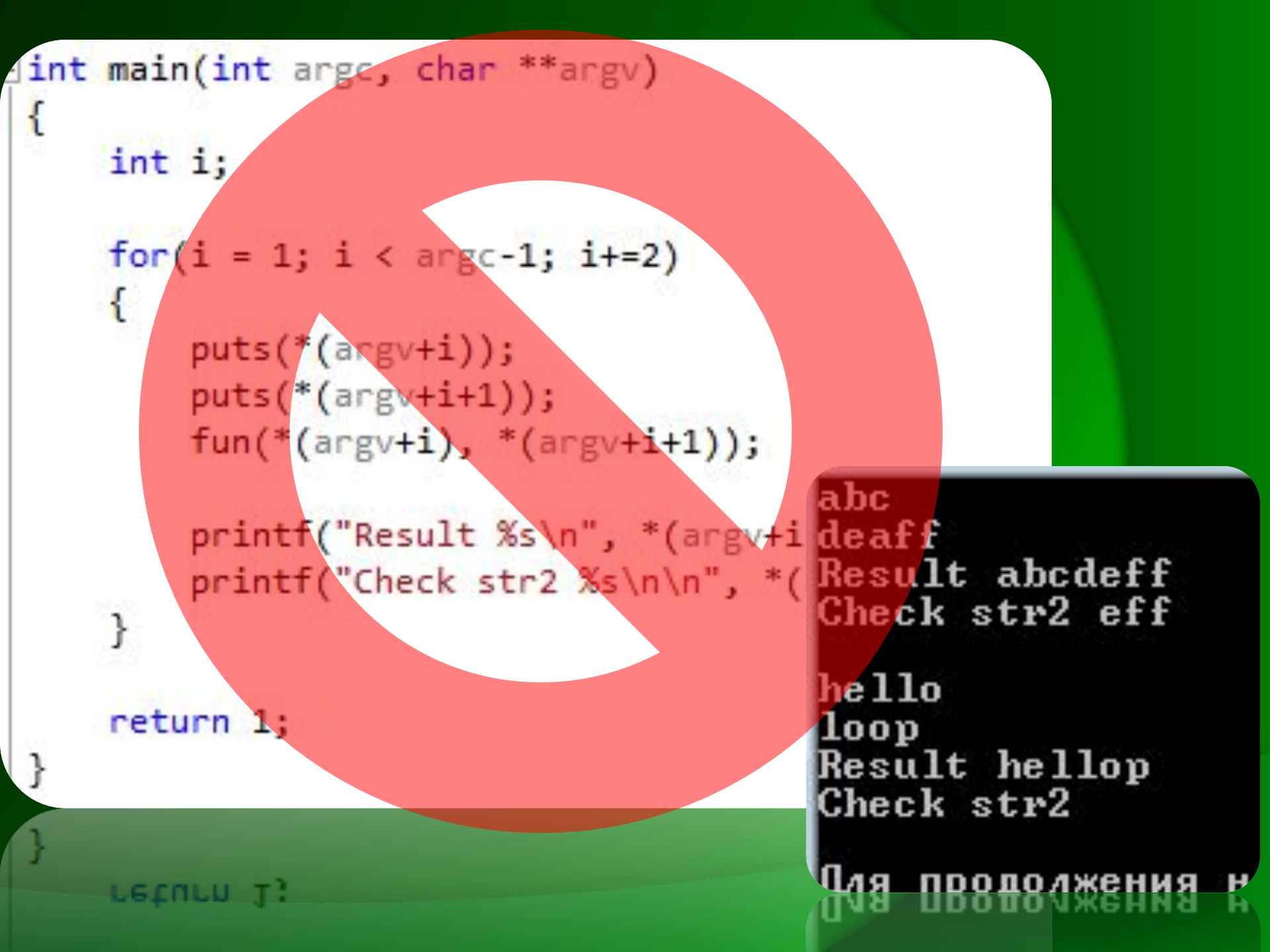
```
a loop abc ao
3
 ao.

1234561
1
23456.
```

	Без доп. массивов	С циклом по ASCII-кодам	С доп. массивом
Память	**5*int = 20 Б**	**4*int = 16 Б**	**258*int > 1КБ**
	STR_LEN 1 .. 100, n 1..10		
Операции if	1 551	12 969	256
Операции for	1 513	12 713	305
	STR_LEN 1000 .. 10000, n 1..10		
Операции if	252 620	1 333 017	256
Операции for	252 570	1 332 761	5 462
	STR_LEN 1 .. 10, n 1..10		
Операции if	24	1 323	256
Операции for	16	1 067	260
Особенность	Вывод в порядке следования	Вывод по возрастанию кодов	Вывод по возрастанию кодов

Вариант 4. Функция, которая…

- принимает указатели на две строки
- возвращает ничего
- добавляет в первую строку из второй все буквы, которых нет в первой строке (включая повторяющиеся).

```c
//добавляет в первую строку из второй все буквы, которых нет в первой строке (включая повторяющиеся)
void fun (char *str1, char *str2)
{
    int i, j, k, isAbsent, len1 = 0;

    while(*(str1+len1)) len1++;
    k = len1;

    for(j = 0; *(str2+j); j++)
    {
        for(isAbsent = 1, i = 0; i < len1; i++) //len1 -> *(str1+i), чтобы не повторялись символы из второй строки
            if(*(str1+i) == *(str2+j))
            {
                isAbsent = 0;
                break;
            }
        if(isAbsent) *(str1+k++) = *(str2+j);
    }
    *(str1+k) = 0;                              //новый конец строки
}
```


int main(int argc, char **argv)
{
int i;
for(i = 1; i < argc-1; i+=2)
{
puts(*(argv+i));
puts(*(argv+i+1));
fun(*(argv+i), *(argv+i+1));
printf("Result %s\n", *(argv+i
printf("Check str2 %s\n\n", *(
}
return 1;
}
abc
deaff
Result abcdeff
Check str2 eff
hello
loop
Result hellop
Check str2

```c
int main(int argc, char
{
    int i;
    char* str1 = (char*)
    if(!str1) {
        puts("Memory all
        return 0;
    }

    for(i = 1; i < argc-1
    {
        puts(*(argv+i));
        puts(*(argv+i+1));
        my_strcpy(str1, *(argv+i));
        fun(str1, *(argv+i+1));

        printf("Result %s\n", str1);
        printf("Check str1 - %s, str2 - %s\n\n", *(argv+i), *(argv+i+1));
    }

    free(str1);

    return 1;
}
```

```
abc
deaff
Result abcdeff
Check str1 - abc, str2 - deaff

hello
loop
Result hellop
Check str1 - hello, str2 - loop

Для продолжения нажмите любую клавишу
```

Вариант 5. Функция, которая…

- принимает указатель на строку, которая не содержит знаков препинания

- возвращает целое число (сумму чисел в строке)

- считает сумму чисел, встречающихся в строке.

```c
//Считает сумму чисел, встречающихся в строке
int sum (char *str)
{
    int i, number = 0, sum = 0;

    for(i = 0; *(str+i); i++)
        if(*(str+i) >= '0' && *(str+i) <= '9')  // цифра
        {
            number *= 10;                       // сдвинуть число
            number += *(str+i) - '0';           // добавить цифру
        }
        else
        {
            sum += number;                      // увеличить сумму
            number = 0;                         // обнулить число
        }

    return sum + number;// Добавить последнее число
}
```

```c
int main(int argc, char **argv)
{
    int i;
    for(i = 1; i < argc; i++)
    {
        puts(*(argv+i));
        printf("Sum = %d\n\n", sum(*(argv+i)));
    }

    return 1;
}
```

```
a1b2c3 123
Sum = 129

01 abs 10
Sum = 11

noNumbers
Sum = 0

-ab01 -100
Sum = 101
```

```
#define DIGIT *(str+i) - '0'
#define MINUS(str, index) *(str+index) == '-'

//Считает сумму чисел (со знаком), встречающихся в строке
int sumSign (char *str)
{
    int i, number = 0, sum = 0, sign = 1;

    for(i = 0; *(str+i); i++)
        if(*(str+i) >= '0' && *(str+i) <= '9')  // ц
        {
            number *= 10;                       // с
            number += DIGIT;                    // д
        }
        else
        {
            sum += sign*number;                 // д
            number = 0;                         // с

            if(MINUS(str, i)) sign = -1;
            else              sign = 1;
        }

    return sum + sign*number;// Добавить последнее ч
}
```

```
a1b2c3 123
Sum = 129

01 abs 10
Sum = 11

noNumbers
Sum = 0

-ab01 -100
Sum = -99
```

Вариант 6. Функция, которая…

- принимает указатель на строку
- возвращает целое число (длину искомой последовательности)
- находит наиболее длинную последовательность повторяющихся букв в строке.

```c
// Находит наиболее длинную последовательность повторяющихся букв в строке
int maxSequence (char *str)
{
    int i, cur = 1, max = 1;

    if(str && *str)                         // строка не пуста
    {
        for(i = 1; *(str+i); i++)           // со второго символа
            if(*(str+i) == *(str+i-1))      // одинаковый с предыдущим
            {
                cur++;                      // длина последовательности
                if(cur > max)               // новый максимум
                    max = cur;
            }
            else
                cur = 1;                    // сбросить текущий счётчик
    }
    else max = 0;

    return max;
}
```

```c
#include<stdio.h>

int maxSequence (char *str);

int main(int argc, char **argv)
{
    int i;
    for(i = 1; i < argc; i++)
    {
        puts(*(argv+i));
        printf("The longest sequence %d\n\n", maxSequence(*(argv+i)));
    }

    return 1;
}
```

```
The longest sequence 0

aaaa bb aaa
The longest sequence 4

ooo
The longest sequence 3

o pp
The longest sequence 2

w
The longest sequence 1
```