|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет экономики, менеджмента и бизнес-информатики* |
|  |
| Зайцев Константин Николаевич  **САМОСТОЯТЕЛЬНАЯ РАБОТА №1**  по направлению подготовки *09.03.04 Программная инженерия*  образовательная программа «Программная инженерия»  *Архитектура вычислительных систем*   |  |  | | --- | --- | |  | Руководитель  Доцент кафедры информационных технологий в бизнесе  Л.Н. Лядова | |

Пермь, 2019 год

Оглавление

[Задача №1 5](#_Toc8125258)

[Постановка задачи 5](#_Toc8125259)

[Алгоритм, используемый тип данных 5](#_Toc8125260)

[Листинг программы 5](#_Toc8125261)

[Задача №2 6](#_Toc8125262)

[Постановка задачи 6](#_Toc8125263)

[Алгоритм, используемый тип данных 6](#_Toc8125264)

[Листинг программы 6](#_Toc8125265)

[Задача №3 7](#_Toc8125266)

[Постановка задачи 7](#_Toc8125267)

[Алгоритм, используемый тип данных 7](#_Toc8125268)

[Листинг программы 7](#_Toc8125269)

[Задача №4 8](#_Toc8125270)

[Постановка задачи 8](#_Toc8125271)

[Алгоритм, используемый тип данных 8](#_Toc8125272)

[Листинг программы 8](#_Toc8125273)

[Задача №5 9](#_Toc8125274)

[Постановка задачи 9](#_Toc8125275)

[Алгоритм, используемый тип данных 9](#_Toc8125276)

[Листинг программы 9](#_Toc8125277)

[Задача №6 10](#_Toc8125278)

[Постановка задачи 10](#_Toc8125279)

[Алгоритм, используемый тип данных 10](#_Toc8125280)

[Листинг программы 10](#_Toc8125281)

[Задача №7 11](#_Toc8125282)

[Постановка задачи 11](#_Toc8125283)

[Алгоритмы, используемый тип данных 11](#_Toc8125284)

[Листинг программы 12](#_Toc8125285)

[Задача №8 13](#_Toc8125286)

[Постановка задачи 13](#_Toc8125287)

[Алгоритмы, используемый тип данных 13](#_Toc8125288)

[Листинг программы 13](#_Toc8125289)

[Задача №9 14](#_Toc8125290)

[Постановка задачи 14](#_Toc8125291)

[Алгоритмы, используемый тип данных 14](#_Toc8125292)

[Листинг программы 15](#_Toc8125293)

[Задача №10 16](#_Toc8125294)

[Постановка задачи 16](#_Toc8125295)

[Алгоритмы, используемый тип данных 16](#_Toc8125296)

[Листинг программы 17](#_Toc8125297)

[Приложение А. Листинг задачи №1 (Основная) 18](#_Toc8125298)

[Приложение B. Листинг задачи №1 (с дополнительными заданиями) 20](#_Toc8125299)

[Приложение С. Листинг задачи №2 22](#_Toc8125300)

[Приложение D. Листинг задачи №3 23](#_Toc8125301)

[Приложение Е. Листинг задачи №4 24](#_Toc8125302)

[Приложение F. Листинг задачи №5 25](#_Toc8125303)

[Приложение G. Листинг задачи №6 27](#_Toc8125304)

[Приложение H. Листинг программ задачи №7 29](#_Toc8125305)

[Приложение H.1. Задача №7.1 29](#_Toc8125306)

[Приложение H.2. Задача №7.2 30](#_Toc8125307)

[Приложение H.3. Задача №7.3 31](#_Toc8125308)

[Приложение H.4. Задача №7.4 32](#_Toc8125309)

[Приложение H.5. Задача №7.5 33](#_Toc8125310)

[Приложение H.6. Задача №7.6 34](#_Toc8125311)

[Приложение H.7. Задача №7.7 35](#_Toc8125312)

[Приложение H.8. Задача №7.8 36](#_Toc8125313)

[Приложение I. Листинг к задаче № 8 37](#_Toc8125314)

[Приложение J. Листинги к задаче № 9 38](#_Toc8125315)

[Приложение J.1. Задача №9.1-3 38](#_Toc8125316)

[Приложение J.2. Задача №9.4 40](#_Toc8125317)

[Приложение J.3. Задача №9. Альтернативное решение 41](#_Toc8125318)

[Приложение K. Листинг программ задач №10 43](#_Toc8125319)

[Приложение K.1. Задача №10.1. m и k входные 43](#_Toc8125320)

[Приложение K.2. Задача №10.1. m и n входные 44](#_Toc8125321)

[Приложение K.3. Задача №10.2. m и k входные 45](#_Toc8125322)

[Приложение K.4. Задача №10.2. m и n входные 46](#_Toc8125323)

# Задача №1

## Постановка задачи

Разработайте процедуру поиска «седлового» элемента матрицы: элемента, являющегося минимумом в строке и максимумом в столбце. Результат – индексы (номер строки и номер столбца) найденного элемента. Если элемент не найден, то в качестве результата – нули. Уточнение: если таких элементов несколько, то выводится только первый.

## Алгоритм, используемый тип данных

Алгоритм заключается в том, чтобы итерационно проходить по каждой строке. На каждой итерации производить проверку на минимальное число в строке и если число - минимум, то запоминать столбец, в котором оно находится. После нахождения минимума производим проверку по столбцу минимума. Если же минимум является максимумом, то останавливаем программу и выдаем результат. В противном случае переходим на следующую строку. И так до тех пор, пока не обработаем все строки.

Для решения поставленной задачи использовался массив типа беззнакового двойного слова. Индексы – байты без знака.

## Листинг программы

Из-за большого объёма программа вынесена в прил. А.

Также была реализована программа для работы с отрицательными числами, которая использует вместо классических циклов типа «loop» условия, и был изменен способ адресации к элементам массива. Программа с выполнением бонусных заданий вынесена в прил. B.

# Задача №2

## Постановка задачи

Разработайте процедуру перевода числа со знаком, записанного в символьной строке, во внутреннее представление в памяти компьютера в формате байта/слова/двойного слова.

## Алгоритм, используемый тип данных

Для перевода числа каждая цифра переводится во внутреннее представление компьютера: берётся символ из исходной строки, уменьшается его код на 48, чтобы получилась цифра. Затем полученное число умножается на 10 и осуществляется переход на новую итерацию. Так делается до тех пор, пока не достигается конец строки. После осуществляется проверка на то, является ли число положительным или отрицательным. Если второй случай, то число в регистре становится противоположным. Затем в переменные записываются результаты.

Исходная строка является массивом байт. Выходные данные – это три переменные: первая – тип байт, вторая – слово, третья – двойное слово. При этом переполнения не учитываются, потому, если входное число является большим для какого-то типа данных, то может выдаться некорректный ответ. Во входной строке перед цифрами может либо “+”, либо “-“, либо число может быть записано без всяких знаков.

## Листинг программы

Листинг представлен в прил. C.

# Задача №3

## Постановка задачи

Разработайте процедуру перевода двоичного числа со знаком из внутреннего представления в памяти компьютера в заданном формате (байта/слова/двойного слова) в символьную запись (в строку символов).

## Алгоритм, используемый тип данных

На вход подаётся число типа двойного слова. Оно копируется в регистр, а там уже автоматически переводится во внутреннее представление. После в цикле производится перевод числа в символьную запись. Число делится на 10, остаток уходит в ячейку массива символов, а с целым производится операция деления на следующей итерации. И так до тех пор, пока целое не будет равно 0. После цикла проверяется знак числа. Если число отрицательное, то к итоговой строке добавляется минус, в противном случае – плюс.

Вход – двойное слово со знаком. Выход – массив символов.

## Листинг программы

Листинг представлен в прил. D.

# Задача №4

## Постановка задачи

Разработайте процедуру, реализующую операцию сложения для целых чисел, представленных в двоично-десятичном формате. Уточнение: оба числа являются положительными, в противном случае это была бы разность. Также оба числа имеют одинаковое количество цифр.

## Алгоритм, используемый тип данных

На вход подаётся символьная строка с цифрами. Затем в цикле производится сложение цифр, при этом используется ещё и ASCII-коррекция, где запоминается единица, если число получилось больше 9. Затем на следующей итерации эта единица прибавляется к цифре. Цикл идёт до тех пор, пока не будут обработаны все цифры символьной строки. После цикла происходит проверка на перенос единицы. Если она была перенесена, то к общему результату добавляется единица.

Входные данные – символьная байтовая беззнаковая строка с цифрами, выходные – байтовая символьная строка без знака.

## Листинг программы

Листинг программы представлен в прил. Е.

# Задача №5

## Постановка задачи

Разработайте процедуру, реализующую операцию вычитания для целых чисел, представленных в двоично-десятичном формате. Уточнение: числа имеют одинаковое количество цифр.

## Алгоритм, используемый тип данных

На вход подаются символьные строки, состоящие из цифр. В цикле производится вычитание чисел, после него – ASCII-коррекция. Если в регистре ah отрицательное число, то на следующей итерации из цифры вычитается единица. Цикл до тех пор, пока не будут пройдены все цифры. После производится проверка на то, получилось ли число отрицательным. Если да, то производится преобразование в отрицательное число.

Входные данные – массивы байтов без знака. Результат – массив байтов без знака.

## Листинг программы

Листинг представлен в прил. F.

# Задача №6

## Постановка задачи

Разработайте процедуру, реализующую операцию сравнения целых чисел, записанных в двоично-десятичном формате. Уточнение: оба входных числа имеют одинаковое количество цифр и оба положительны.

## Алгоритм, используемый тип данных

На вход подаются символьные строки. В цикле производится перевод числа в BCD-формат путём деления на два и записи остатка в результирующую строку для определенного числа. Перевод осуществляется для двух чисел. Далее в цикле последовательно проверяются биты. Если они не равны, то цикл прерывается и в зависимости от величины числа в результирующую переменную записывается результат. Если все числа одинаковы, то в переменную записывается 0.

Входные данные – массивы байтов без знака, выходные – байт со знаком.

## Листинг программы

Листинг представлен в прил. G.

# Задача №7

## Постановка задачи

Используя команды обработки цепочек, решите следующие задачи:

1. Заполните элементы массива значением, заданным пользователем.

2. Даны два массива. Количество элементов во втором массиве кратно числу элементов в первом. Заполните второй массив значениями элементов первого массива последовательно повторяя их.

3. Найдите в массиве первый элемент, значение которого совпадает со значением, заданным пользователем.

4. Найдите в массиве последний элемент, значение которого совпадает со значением, заданным пользователем.

5. Сравните два массива на совпадение (результат – Да или Нет).

6. Найдите первый слева элемент, значение которого в первом массиве не совпадает со значением элемента, стоящего на том же месте во втором массиве.

7. Найдите первый справа элемент, значение которого в первом массиве не совпадает со значением элемента, стоящего на том же месте во втором массиве.

8. Разработайте функцию поиска подстроки в строке (последовательности элементов одного массива (цепочки), повторяющейся во втором массивы) с использованием операций над цепочками. Результат – номер элемента, соответствующего первому элементу найденной подстроки, или признак, что значение не найдено (например, –1).

## Алгоритмы, используемый тип данных

1. На вход подаётся массив типа двойного слова без знака. При помощи цепочечной команды movs и префикса repe производится копирование в результирующий массив.
2. Вход – массив типа слова со знаком. При помощи тех же команд производится копирование, но перед копированием в регистр ecx помещается размер результирующего массива.
3. На вход подаётся массив типа слова без знака. При помощи цепочечной команды scas и префикса repnz производится поиск элемента. При этом для поиска слева-направо обнуляется флаг направления командой cld.
4. Вход – массив типа слова со знаком. Команды для поиска те же, кроме изменения направления. Для поиска справа-налево флагу направления присваивается единица командой std.
5. На вход – два массива типа слова без знака одинаковой длины. Для сравнения применяется команда cmpsw с префиксом repe, так как оба массива – слова. Если они эквивалентны, то выводится «Yes», иначе – «No».
6. Вход – два массива типа байта без знака одинаковой длины. Для выполнения задачи использовалась команда cmps с префиксом repe. Если строки не равны, то вычисляется индекс элемента. В противном случае выводится -1, как результат того, что строки равны.
7. Вход – два массива типа слова со знаком одинаковой длины. Для выполнения задачи используются те же команды, что и в 6 пункте, кроме того, что флаг направления задаётся единицей командой std.
8. Вход – два массива типа байта без знака. В цикле сначала находится индекс первого символа подстроки в строке. Далее, начиная с этого индекса и заканчивая длиной подстроки, производится на то, что выделенная подстрока является искомой. Если нет, то перейти на следующую итерацию и так до тех пор, пока не будет обработана вся строка. В противном случае цикл прерывается вычисляется индекс начала подстроки в строке.

## Листинг программы

Все вышеперечисленные программы представлены в прил. H.

# Задача №8

## Постановка задачи

Разработайте процедуры, реализующие

1. Шифрование (подстановочный шифр) текста, записанного в символьном формате с использованием 8-битовых кодов.

2. Расшифрование шифротекста, записанного в символьном формате с использованием 8 битовых кодов, полученного с помощью подстановочного шифра (предшествующее задание).

## Алгоритмы, используемый тип данных

1. На вход подаётся массив байтов без знака. В цикле обрабатывается каждый символ массива. При помощи команды xlat осуществляется подстановка. Затем в результирующий массив подставляется полученный символ. Так повторяется до тех пор, пока не обработается последний символ в массиве.
2. На вход подаётся массив байтов без знака. В цикле обрабатывается каждый символ зашифрованного текста. Так как в алфавите, при помощи которого происходило шифрование, смещение на 4 буквы, то из элемента массивы отнимается 4. Если буква является «a», «b», «c» или «d», то она заменяется на соответствующую букву в конце латинского алфавита. Так до тех пор, пока не обработается вся строка.

## Листинг программы

Листинг представлен в прил. I. Стоит отметить, что в алфавите были задействованы маленькие буквы латинского алфавита. С другими символами программа будет работать некорректно.

# Задача №9

## Постановка задачи

Разработайте процедуры, реализующие операции над множествами:

1. Объединение множеств (пример – реализация множеств в языке Pascal).

2. Пересечение множеств (пример – реализация множеств в языке Pascal).

3. Разность множеств (пример – реализация множеств в языке Pascal).

4. Определение принадлежности элемента множеству (пример – реализация множеств в языке Pascal).

## Алгоритмы, используемый тип данных

В данной задаче я использовал массив байтов размером 256, чтобы охватить все символы. Потому максимальное количество символов может быть 256. Так как каждый символ имеет свой код, то этот код является индексом в множестве. При таком массиве гарантировано отсутствие повторяющихся элементов.

1. Для операции объединения была использована команда or для каждого элемента множеств. Результат команды записывался в результирующее множество.
2. Для операции умножения была использована команда and для каждого элемента множеств.
3. Для операции разности была использована команда xor для каждого элемента множеств.
4. Для операции определения принадлежности производилась ссылка на элемент по коду символа. Затем использовалась команда and с единицей. Если результат единица, то элемент присутствует, иначе – отсутствует.

Хотелось бы сказать о другой вариации выполнения задачи на примере объединения множества. В ней использовались бы цепочечные команды. При помощи первого цикла производилась бы проверка того, находится ли элемент первого множества в результирующем множестве. Если нет, то добавляется в результат, если да, то следующая итерация. Во втором цикле производилось бы объединение множеств путём проверки наличия символа второго множества в первом и результирующем. В целом такой алгоритм оказывается неэффективным по сравнению с предыдущим из-за большого количества циклов и всяческих проверок.

## Листинг программы

Листинг программы, выполняющей первые три задачи представлен в прил. J.1. Листинг программы, отвечающей за четвертую задачу представлен в прил. J.2. Листинг альтернативной программы, реализующей объединение, представлен в прил. J.3.

# Задача №10

## Постановка задачи

Разработайте функции, вычисляющие

1. Результат умножения целого числа m на число k, являющееся n й степенью числа 2 (F(m, k) = m × k, где k = 2n). Примечание: входными данными для решения задачи могут быть m и k или m и n (при вводе в качестве параметров для вычисления функции пары m и k необходимо организовать проверку, является k ли степенью числа 2 – это усложнённый вариант задания).

2. Частное (целую часть) от деления целого числа m на число k, являющееся n й степенью числа 2 (F(m, k) = m / k, где k = 2n). Примечание: входными данными для решения задачи могут быть m и k или m и n (при вводе в качестве параметров для вычисления функции пары m и k необходимо организовать проверку, является k ли степенью числа 2 – это усложнённый вариант задания).

Уточнение: ниже будут приведены программы, работающие и с k, и с n.

## Алгоритмы, используемый тип данных

1. Для k:

На вход подаются два числа типа двойного слова без знака, m – умножаемое число и k – число, на которое надо умножить. Для k производится проверка. В два регистра копируются значение k, а затем из второго вычитается единица. После к ним применяется команда and. Если результат не равен нулю, то это число нам подходит. Затем происходит умножение числа m и k при помощи команды mul.

Для n:

На вход – m и n типа двойного слова без знака. Для операции умножения используется команда cal.

1. Для k:

Все те же данные, что и в предыдущем задании. Проверка та же. Деление производится путём использования команды div.

Для n:

Те же данные. Для операции деления используется команда shr.

Стоит сказать, что программа не обрабатывает переполнения и не работает с отрицательными числами. Потому при некоторых значения программа будет выдавать некорректны результат (например, при n, большем 255).

## Листинг программы

Все программы представлены в прил. K.

# Приложение А. Листинг задачи №1 (Основная)

int main()

{

const unsigned int n = 4; //Строки

const unsigned int m = 4; //Столбцы

static unsigned int A[n][m]; //Массив размера двойного слова

static unsigned char min\_row; //Индекс строки

static unsigned char max\_col; //Индекс столбца

unsigned int k = m \* 4; //Переменная для перехода на следующую строку

unsigned short divider\_row = 16; //Переменная для приведения в нормальный вид индекс строки

unsigned short divider\_col = 4; //Переменная для приведения в нормальный вид индекс столбца

for (int I = 0; I < n; ++i) {

for (int j = 0; j < m; ++j) {

cin >> A[i][j];

}

}

\_\_asm {

push eax

push esi

push edi

push ecx

push ebx

push edx

xor ecx, ecx

o reax, eax; o reax

xor ebx, ebx; maximum

xor edx, edx; temp

xor esi, esi

xor edi, edi

mov ecx, n

loopExternal:

push ecx

mov eax, A[esi]

mov edi, 4

mov ecx, m

dec ecx

loopInternal :

mov edx, A[edi + esi]

cmp eax, edx

jbe minL

mov eax, edx

mov ebx, edi

minL:

add edi, 4

loop loopInternal

push esi

xor esi, esi

mov ecx, n

loopFind :

mov edx, A[esi + ebx]

cmp eax, edx

jae label

jmp newLoop

label:

add esi, k

loop loopFind

o reax, eax

pop esi

mov eax, esi

div divider\_row

add al, 49

mov min\_row, al

o reax, eax

;mov min\_row, esi

mov eax, ebx

div divider\_col

add al, 49

mov max\_col, al

;mov max\_col, ebx

jmp labelExit

newLoop:

pop esi

pop ecx

add esi, k

loop loopExternal

mov min\_row, 48

mov max\_col, 48

labelExit:

pop eax

pop esi

pop edi

pop ecx

pop ebx

pop edx

}

cout << min\_row << “ “ << max\_col;

}

# Приложение B. Листинг задачи №1 (с дополнительными заданиями)

int main()

{

const unsigned int n = 4;

const unsigned int m = 6;

static signed int A[n][m];

static unsigned char min\_row = 1;

static unsigned char max\_col = 1;

unsigned short divider\_row = n \* m;

unsigned char divider\_col = 4;

static unsigned int offs = m \* 4;

for (int I = 0; I < n; ++i) {

for (int j = 0; j < m; ++j) {

cin >> A[i][j];

}

}

\_\_asm {

push eax

push esi

push edi

push ecx

push ebx

push edx

xor ecx, ecx

o reax, eax; o reax

xor ebx, ebx; maximum

xor edx, edx; temp

xor esi, esi

xor edi, edi

lea esi, A

mov ecx, n; 4

labelExternal:

push ecx

xor ebx, ebx

mov eax, [esi]

mov edi, 4

mov ecx, m

dec ecx

labelInternal:

mov edx, [edi + esi]

cmp eax, edx

jle minL

mov eax, edx

mov ebx, edi

minL :

add edi, 4

dec ecx

cmp ecx, 0

jne labelInternal

push esi

xor esi, esi

lea esi, A

mov ecx, n

labelFind:

mov edx, [esi + ebx]

cmp eax, edx

jge label

jmp newLoop

label:

add esi, offs

dec ecx

cmp ecx, 0

jne labelFind

o reax, eax

; mov min\_row, esi

mov eax, ebx

div divider\_col

add al, 49

mov max\_col, al

; mov max\_col, ebx

jmp labelExit

newLoop :

inc min\_row

pop esi

pop ecx

add esi, offs

dec ecx

cmp ecx, 0

jne labelExternal

mov min\_row, 0

mov max\_col, 48

labelExit:

pop eax

pop esi

pop edi

pop ecx

pop ebx

pop edx

}

cout << min\_row + 0 << “ “ << max\_col;

}

# Приложение С. Листинг задачи №2

int main()

{

const unsigned int n = 4; //Количество строк

unsigned int multiple = 10; //Множитель для перевода

static unsigned char str[n]; //Входная строка

static unsigned char res[CHAR\_MAX]; //Выходная строка

static signed int resultd; //Результат в виде двойного слова

static signed char resultb; //Результат в виде байта

static signed short resultw; //Результат в виде слова

//Ввод может быть как +123, -123 или просто 0123

for (int I = 0; I < n; ++i) {

cin >> str[i];

}

\_asm {

push ecx

push eax

push esi

push edx

push ebx

o reax, eax

xor esi, esi

xor ecx, ecx

xor edx, edx

xor ebx, ebx

mov ecx, n

cmp str[0], ‘+’

jne labelNextCheck

dec ecx

inc esi

jmp loopNumber

labelNextCheck:

cmp str[0], ‘-‘

jne loopNumber

dec ecx

inc esi

loopNumber:

mov dl, str[esi]

inc esi

sub dl, 48

add al, dl

mul multiple

loop loopNumber

div multiple

cmp str[0], 45

jne labelResult

neg eax

labelResult:

mov resultd, eax

mov o rea, ax

mov o rea, al

pop ecx

pop eax

pop esi

pop edx

pop ebx

}

cout << “Double word: “ << resultd << ‘\n’;

cout << “Word: “ << o rea << ‘\n’;

cout << «Byte: « << resultb + 0;

}

# Приложение D. Листинг задачи №3

int main()

{

static unsigned int n; //Количество символов в выходной строке

static signed int numb; //Исходное число

unsigned int multiple = 10; //Делитель для перевода

static unsigned char res[CHAR\_MAX]; //Выходная строка

cin >> numb;

\_asm {

push ecx

push eax

push esi

push edx

push ebx

o reax, eax

xor esi, esi

xor ecx, ecx

xor edx, edx

xor ebx, ebx

mov eax, numb

cmp eax, 0

jge labelDiv

inc ebx

neg eax

labelDiv:

div multiple

mov res[esi], dl

add res[esi], 48

xor edx, edx

inc esi

cmp ax, 0

jne labelDiv

cmp ebx, 0

je labelAddPlus

mov res[esi], 45

jmp labelContinue

labelAddPlus:

mov res[esi], 43

labelContinue:

mov n, esi

pop ecx

pop eax

pop esi

pop edx

pop ebx

}

for (int I = n; I >= 0; --i) {

cout << res[i];

}

}

# Приложение Е. Листинг задачи №4

int main()

{

const unsigned int n = 1; //Количество цифр

static unsigned int m = n; //Количество цифр в результате

static unsigned char num1[n]; //Первое число

static unsigned char num2[n]; //Второе число

static unsigned char res[CHAR\_MAX]; //Результат

for (int I = 0; I < n; ++i) {

cin >> num1[i];

}

for (int I = 0; I < n; ++i) {

cin >> num2[i];

}

\_\_asm {

push eax

push ecx

push esi

o reax, eax

xor ecx, ecx

xor esi, esi

xor edi, edi

mov edi, n

mov ecx, n

o reax

loopAddition:

mov al, num1[edi]

add al, ah

add al, num2[edi]

;add al, ah

mov ah, 0

aaa

mov res[esi], al

add res[esi], 48

o reax

inc esi

loop loopAddition

cmp ah, 0

je labelExit

mov res[esi], ah

add res[esi], 48

inc esi

labelExit:

;mov n, esi

mov m, esi

pop eax

pop ecx

pop esi

}

for (int I = m – 1; I >= 0; --i) {

cout << res[i];

}

}

# Приложение F. Листинг задачи №5

int main()

{

const unsigned n = 5; //Количество цифр в числе

static unsigned int m = n; //Результирующее количество элементов

static unsigned char num1[n]; //Первое число

static unsigned char num2[n]; //Второе число

static unsigned char res[CHAR\_MAX]; //Результат

for (int I = 0; I < n; ++i) {

cin >> num1[i];

}

for (int I = 0; I < n; ++i) {

cin >> num2[i];

}

\_\_asm {

push eax

push ecx

push esi

push ebx

xor ebx, ebx

mov ecx, n

mov esi, n

dec esi

loopSub:

mov al, num1[esi]

sub al, ah

sub al, num2[esi]

mov ah, 0

aas

cmp ah, 255

jne labelOk

mov ah, 1

labelOk:

mov res[esi], al

add res[esi], 48

dec esi

loop loopSub

cmp ah, 0

je labelContinue

o reax, eax

mov ecx, n

inc m

mov esi, n

dec esi

mov al, res[esi]

loopNeg:

add al, ah

sub al, 58

neg al

mov ah, 1

inc esi

mov res[esi], al

add res[esi], 48

sub esi, 2

mov al, res[esi]

loop loopNeg

mov res[0], ‘-‘

labelContinue:

pop eax

pop ecx

pop esi

pop ebx

}

for (int I = 0; I < m; ++i) {

cout << res[i];

}

}

# Приложение G. Листинг задачи №6

int main()

{

const unsigned int n = 5; //Количество цифр в числе

const unsigned int m = n \* 4; //Количество цифр в BCD-числе

static unsigned char num1[n]; //Первое число

static unsigned char num2[n]; //Второе число

static unsigned char num1BCD[m]; //Первое BCD-число

static unsigned char num2BCD[m]; //Второе BCD-число

static unsigned char divider = 2; //Делитель для перевода

static unsigned char checker = 4; //Делитель для проверки

static signed char res = 0; //Результирующая переменная

for (int I = 0; I < n; ++i) {

cin >> num1[i];

}

for (int I = 0; I < n; ++i) {

cin >> num2[i];

}

\_\_asm {

push eax

push ecx

push esi

push edi

mov ecx, n

mov esi, n

mov edi, m

o reax

dec esi

loopTranslateNum1:

mov al, num1[esi]

labelInternalNum1:

div divider

mov num1BCD[edi], ah

add num1BCD[edi], 48

o reax

push eax

mov eax, edi

inc eax

div checker

cmp ah, 0

pop eax

jne labelInternalNum1

dec esi

loop loopTranslateNum1

mov esi, n

mov ecx, n

mov edi, m

o reax

dec esi

loopTranslateNum2:

mov al, num2[esi]

labelInternalNum2:

div divider

mov num2BCD[edi], ah

add num2BCD[edi], 48

o reax

push eax

mov eax, edi

inc eax

div checker

cmp ah, 0

pop eax

jne labelInternalNum2

dec esi

loop loopTranslateNum2

mov ecx, m

mov esi, -1

o reax, eax

loopCmp:

inc esi

mov al, num1BCD[esi]

cmp num2BCD[esi], al

jnbe labelMinus

mov res, 1

jnae labelRes

mov res, 0

loop loopCmp

jmp labelRes

labelMinus:

mov res, -1

labelRes:

pop eax

pop ecx

pop esi

pop edi

}

cout << res + 0;

}

# Приложение H. Листинг программ задачи №7

## Приложение H.1. Задача №7.1

int main()

{

const unsigned n = 5; //Количество элементов в массиве

static unsigned int mas1[n]; //Исходный массив

static unsigned int mas2[n]; //Массив, в который копируют

for (int I = 0; I < n; ++i) {

cin >> mas1[i];

}

\_\_asm {

push ecx

push edi

push esi

lea esi, mas1

lea edi, mas2

mov ecx, n

cld

repe movs mas2, mas1

pop ecx

pop edi

pop esi

}

for (int I = 0; I < n; ++i) {

cout << mas2[i] << ‘ ‘;

}

}

## Приложение H.2. Задача №7.2

int main()

{

const unsigned int n = 2; //Количество элементов для первого массива

const unsigned int m = n \* 5; //Количество элементов для второго массива

static signed short mas1[n]; //Первый массив

static signed short mas2[m]; //Второй массив

for (int I = 0; I < n; ++i) {

cin >> mas1[i];

}

\_\_asm {

push esi

push edi

push ecx

mov ecx, m

lea esi, mas1

lea edi, mas2

cld

repe movs mas2, mas1

pop esi

pop edi

pop ecx

}

for (int I = 0; I < m; ++i) {

cout << mas2[i] << ‘ ‘;

}

}

## Приложение H.3. Задача №7.3

int main()

{

const unsigned int n = 5; //количество чисел

static unsigned char mas[n]; //массив чисел

static signed char num; //искомое число

for (int I = 0; I < n; ++i) {

cin >> mas[i];

}

cin >> num;

\_\_asm {

push ecx

push esi

push edi

push eax

lea edi, mas

mov al, num

mov ecx, n

repnz scas mas

mov eax, n

je found

mov ecx, n

inc ecx

found:

sub eax, ecx

mov num, al

pop ecx

pop esi

pop edi

pop eax

}

cout << num + 0;

}

## Приложение H.4. Задача №7.4

int main()

{

const unsigned int n = 5; //Количество элементов в массиве

static signed char mas[n]; //Массив

static signed char num; //Искомое число

static signed int index; //Индекс числа

for (int I = 0; I < n; ++i) {

cin >> mas[i];

}

cin >> num;

\_\_asm {

push eax

push ecx

push edi

mov eax, n

lea edi, mas[eax – 1]

o reax, eax

mov al, num

mov ecx, n

std

repnz scas mas

je found

mov ecx, -2

found:

inc ecx

mov index, ecx

cld

pop eax

pop ecx

pop edi

}

cout << index;

}

## Приложение H.5. Задача №7.5

int main()

{

const unsigned int n = 3; //Количество элементов

static unsigned short mas1[n]; //первый массив

static unsigned short mas2[n]; //Второй массив

static signed char res[3]; //Результат

for (int I = 0; I < n; ++i) {

cin >> mas1[i];

}

for (int I = 0; I < n; ++i) {

cin >> mas2[i];

}

\_\_asm {

push ecx

push esi

push edi

lea edi, mas1

lea esi, mas2

mov ecx, n

cld

repe cmpsw

je ok

mov res[0], ‘N’

mov res[1], ‘O’

jmp quit

ok:

mov res[0], ‘Y’

mov res[1], ‘E’

mov res[2], ‘S’

quit:

pop ecx

pop esi

pop edi

}

for (int I = 0; I < 3; ++i) {

cout << res[i];

}

}

## Приложение H.6. Задача №7.6

int main()

{

const unsigned int n = 4; //Количество элементов

static unsigned char mas1[n]; //Первый массив

static unsigned char mas2[n]; //Второй массив

static signed int index; //Индекс

for (int I = 0; I < n; ++i) {

cin >> mas1[i];

}

for (int I = 0; I < n; ++i) {

cin >> mas2[i];

}

\_\_asm {

push ecx

push esi

push edi

push ebx

mov ebx, n

lea esi, mas1

lea edi, mas2

mov ecx, n

cld

repe cmps

jne notEqual

mov ecx, n

inc ecx

notEqual:

sub ebx, ecx

mov index, ebx

pop ecx

pop esi

pop edi

pop ebx

}

cout << index;

}

## Приложение H.7. Задача №7.7

int main()

{

const unsigned int n = 5; //Количество элементов

static signed short mas1[n]; //Первый массив

static signed short mas2[n]; //Второй массив

static signed int index; //Индекс

for (int I = 0; I < n; ++i) {

cin >> mas1[i];

}

for (int I = 0; I < n; ++i) {

cin >> mas2[i];

}

\_\_asm {

push edi

push esi

push ecx

mov ecx, n

lea edi, mas1[ecx\*2 – 2]

lea esi, mas2[ecx\*2 – 2]

std

repe cmpsw

jne notEqual

mov ecx, -2

notEqual:

inc ecx

mov index, ecx

cld

pop edi

pop esi

pop ecx

}

cout << index;

}

## Приложение H.8. Задача №7.8

int main()

{

const unsigned int n = 5; //Размер строки

const unsigned int m = 2; //Размер подстроки

static unsigned char str[n]; //Строка

static unsigned char substr[m]; //Подстрока

static signed int res; //Результат

for (int I = 0; I < n; ++i) {

cin >> str[i];

}

for (int I = 0; I < m; ++i) {

cin >> substr[i];

}

\_\_asm {

push ecx

push eax

push edi

push esi

mov al, substr[0]

lea edi, str

mov ecx, n

cld

repeat:

repnz scas str

jne notFound

sub ecx, n

neg ecx

dec ecx

lea edi, str[ecx]

lea esi, substr

push ecx

mov ecx, m

cld

repe cmpsb

je found

pop ecx

inc ecx

cmp ecx, n

jae notFound

lea edi, str[ecx]

sub ecx, n

neg ecx

;inc ecx

cld

jmp repeat

notFound:

mov res, -1

jmp quit

found:

pop ecx

inc ecx

mov res, ecx

quit:

cld

pop eax

pop ecx

pop edi

pop esi

}

cout << res + 0;

}

# Приложение I. Листинг к задаче № 8

int main()

{

const unsigned int n = 5; //Количество букв

static unsigned char source[n]; //Исходный текст

static unsigned char table[26]{ ‘e’, ‘f’, ‘g’, ‘h’, ‘I’, ‘j’, ‘k’, ‘l’, ‘m’, ‘n’, ‘o’, ‘p’, ‘q’,

‘r’, ‘s’, ‘t’, ‘u’, ‘v’, ‘w’, ‘x’, ‘y’, ‘z’, ‘a’, ‘b’, ‘c’, ‘d’ }; //Алфавит

static unsigned char result[n]; //Результат шифровки

static unsigned char resultDec[n]; //Результат дешифровки

for (int I = 0; I < n; ++i) {

cin >> source[i];

}

\_\_asm {

push eax

push ebx

push ecx

push esi

xor esi, esi

o reax, eax

xor ebx, ebx

mov ecx, n

lea ebx, table

loopCipher:

mov al, source[esi]

sub al, 97

xlat

mov result[esi], al

inc esi

loop loopCipher

xor esi, esi

mov ecx, n

loopDecipher:

mov al, result[esi]

sub al, 4

cmp al, 97

jae labelDec

sub al, 97

add al, 123

labelDec:

mov resultDec[esi], al

inc esi

loop loopDecipher

push eax

push ebx

push ecx

push esi

}

for (int I = 0; I < n; ++i) {

cout << result[i];

}

cout << ‘\n’;

for (int I = 0; I < n; ++i) {

cout << resultDec[i];

}

}

# Приложение J. Листинги к задаче № 9

## Приложение J.1. Задача №9.1-3

int main()

{

const unsigned int n = 256; //Количество возможных вариантов

const unsigned int m = 2; //Количество элементов в первом множестве

const unsigned int k = 4; //Количество элементов во втором множестве

static unsigned resn; //Количество элементов в результате

static signed char sign; //Знак операции

static unsigned char mas1[m]; //Первый исходный массив

static unsigned char mas2[k]; //Второй исходный массив

static unsigned char set1[n]; //Первое множество

static unsigned char set2[n]; //Второе множество

static unsigned char resset[n]; //Результирующее множество

//Для выполнения операции объединения после ввода первого множества нажать на «+»

//Для выполнения операции пересечения после ввода первого множества нажать на «\*»

//Для выполнения операции разности после ввода первого множества нажать на «-«

for (int I = 0; I < m; ++i) {

cin >> mas1[i];

}

cin >> sign;

for (int I = 0; I < k; ++i) {

cin >> mas2[i];

}

\_\_asm {

push esi

push edi

push eax

push ecx

o reax, eax

xor esi, esi

xor edi, edi

lea esi, mas1

lea edi, set1

mov ecx, m

call copyMas

lea esi, mas2

lea edi, set2

mov ecx, k

call copyMas

mov al, sign

mov ecx, n

xor esi, esi

xor edi, edi

cmp al, ‘+’

jne labelNextOpMul

loopPlus:

mov al, set1[esi]

or al, set2[esi]

cmp al, 0

je nextPlus

mov eax, esi

mov resset[edi], al

o reax

nextPlus:

inc esi

loop loopPlus

mov resn, edi

jmp endProg

labelNextOpMul:

cmp al, ‘\*’

jne labelNextOpMinus

loopMul:

mov al, set1[esi]

and al, set2[esi]

cmp al, 0

je nextMul

mov eax, esi

mov resset[edi], al

o reax

nextMul:

inc esi

loop loopMul

mov resn, edi

jmp endProg

labelNextOpMinus:

cmp al, ‘-‘

jne labelNextOpIn

loopMinus:

mov al, set1[esi]

push eax

xor al, set2[esi]

cmp al, 0

je nextMinus

pop eax

cmp al, 0

je nextMinus

mov eax, esi

mov resset[edi], al

o reax

nextMinus:

inc esi

loop loopMinus

mov resn, edi

jmp endProg

labelNextOpIn:

jmp endProg

copyMas:

loopInsert:

mov al, [esi]

mov [edi + eax], 1

inc esi

loop loopInsert

ret

endProg:

pop esi

pop edi

pop eax

pop ecx

}

for (int I = 0; I < resn; ++i) {

cout << resset[i] << ‘ ‘;

}

}

## Приложение J.2. Задача №9.4

int main()

{

const unsigned int n = 256; //Количество всех возможных элементов

const unsigned int m = 5; //Количество элементов в первом множестве

static unsigned char mas[m]; //Первое множество

static unsigned char bits[n]; //Множество со всеми элементами

static unsigned char element; //Искомый элемент

static bool res; //Результат

//Для начала вводится искомый элемент, а потом множество

cin >> element;

for (int I = 0; I < m; ++i) {

cin >> mas[i];

}

\_\_asm {

push esi

push ecx

push eax

o reax, eax

xor esi, esi

mov ecx, m

loopInsert:

mov al, mas[esi]

mov bits[eax], 1

inc esi

loop loopInsert

o reax, eax

mov al, element

mov al, bits[eax]

and al, 1

mov res, al

pop esi

pop eax

pop ecx

}

if (res) {

cout << “Yes”;

}

else {

cout << “No”;

}

}

## Приложение J.3. Задача №9. Альтернативное решение

int main()

{

const unsigned int n = 3;

const unsigned int m = 3;

static unsigned char mas1[n];

static unsigned char mas2[m];

static unsigned char resmas[n + m];

static unsigned char sign;

static unsigned int k;

for (int I = 0; I < n; ++i) {

cin >> mas1[i];

}

cin >> sign;

for (int I = 0; I < m; ++i) {

cin >> mas2[i];

}

\_\_asm {

push eax

push ecx

push edi

push esi

push ebx

xor ebx, ebx

mov al, sign

xor esi, esi

cmp al, 43

jne otherOpMul

mov ecx, n

loopCheckMas1:

cld

mov al, mas1[esi]

lea edi, resmas

push ecx

mov ecx, ebx

inc ecx

repnz scas resmas

je labelNextMas1

mov resmas[ebx], al

inc ebx

labelNextMas1:

pop ecx

inc esi

loop loopCheckMas1

xor esi, esi

mov ecx, m

loopCheckMas2:

cld

mov al, mas2[esi]

lea edi, resmas

push ecx

mov ecx, ebx

inc ecx

repnz scas resmas

je labelNextMas2

mov resmas[ebx], al

inc ebx

labelNextMas2:

pop ecx

inc esi

loop loopCheckMas2

mov k, ebx

pop eax

pop ecx

pop edi

pop esi

}

for (int I = 0; I < k; ++i) {

cout << resmas[i] << ‘ ‘;

}

}

# Приложение K. Листинг программ задач №10

## Приложение K.1. Задача №10.1. m и k входные

int main()

{

static unsigned int m;

static unsigned int k;

static unsigned int res; //Результат

static bool ok = true; //Флаг

cin >> m;

cin >> k;

\_\_asm {

push ebx

push eax

push ecx

mov eax, k

call checkK

cmp ebx, 0

je endComp

mov eax, m

call computeMul

mov res, eax

jmp endProg

endComp:

mov ok, 0

jmp endProg

checkK:

mov ebx, eax

dec eax

and eax, ebx

xor ebx, ebx

cmp eax, 0

jne endProc

inc ebx

endProc:

ret

computeMul:

mov ecx, k

mul ecx

ret

endProg:

pop ebx

pop eax

pop ecx

}

if (ok) {

cout << res;

}

else {

cout << "Wrong k!";

}

}

## Приложение K.2. Задача №10.1. m и n входные

int main()

{

static unsigned int m;

static unsigned int n;

static unsigned int res; //Результат

cin >> m;

cin >> n;

\_\_asm {

push ebx

push eax

push ecx

mov eax, m

call computeMul

mov res, eax

jmp endProg

computeMul:

mov ecx, n

sal eax, cl

ret

endProg:

pop ebx

pop eax

pop ecx

}

cout << res;

}

## Приложение K.3. Задача №10.2. m и k входные

int main()

{

static unsigned int m;

static unsigned int k;

static unsigned int res;

static bool ok = true;

cin >> m;

cin >> k;

\_\_asm {

push ebx

push eax

push ecx

push edx

xor eax, eax

xor ebx, ebx

xor edx, edx

mov eax, k

call checkK

cmp ebx, 0

je endComp

mov eax, m

call computeMul

mov res, eax

jmp endProg

endComp:

mov ok, 0

jmp endProg

checkK:

mov ebx, eax

dec eax

and eax, ebx

xor ebx, ebx

cmp eax, 0

jne endProc

inc ebx

endProc:

ret

computeMul:

div k

ret

endProg:

pop ebx

pop eax

pop ecx

pop edx

}

if (ok) {

cout << res;

}

else {

cout << "Wrong k!";

}

}

## Приложение K.4. Задача №10.2. m и n входные

int main()

{

static unsigned int m;

static unsigned int n;

static unsigned int res;

cin >> m;

cin >> n;

\_\_asm {

push ebx

push eax

push ecx

push edx

xor eax, eax

xor ebx, ebx

xor edx, edx

mov eax, m

call computeMul

mov res, eax

jmp endProg

checkK:

mov ebx, eax

dec eax

and eax, ebx

xor ebx, ebx

cmp eax, 0

jne endProc

inc ebx

endProc:

ret

computeMul:

mov ecx, n

shr eax, cl

ret

endProg:

pop ebx

pop eax

pop ecx

pop edx

}

cout << res;

}