МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпропетровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Технічна Кібернетика»

**Лабораторна робота №6 частина 2  
з дисципліни «Архітектура комп’ютера»**

**на тему: «**Організація введення та виведення інформації в консолі на рівні Ассемблера. Обробка масивів чисел в мові Ассемблера**»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Нежуміра О.И.

Дніпро, 2019

**Тема.** Організація введення та виведення інформації в консолі на рівні Ассемблера. Обробка масивів чисел в мові Ассемблера.

**Мета.** Вивчити засоби мови Ассемблера для введення та виведення числової інформації. Набути практичних навичок з обробки двомірних числових масивів мовою Ассемблера.

# **Постановка задач згідно з загальним та індивідуальними завданнями.**

*Загальне завдання:*

Розробити програму відповідно до обраного рівню.

Вимоги до програми:

* всі вхідні дані вводиться з клавіатури;
* передбачити обробку помилки введення числа для всіх вхідних даних;
* передбачити перевірку розмірності на відповідність заданому діапазону;
* введений масив/матриця виводяться на консоль;
* результати роботи програми виводяться на консоль.

Вимоги до тексту програми:

* коментарі щодо призначення програми, її вхідних і вихідних даних;
* коментарі щодо призначення кожного блоку програми, дій окремих команд для пояснення алгоритму;
* самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних і міток.

*Індивідуальне завдання:*

Рівень А (відмінно)

Дано двовимірний масив розмірністю N (від 3 до 5) на M (від 3 до 5). Елемент масиву – ціле число зі знаком з діапазону слово. Сформувати масив розмірністю М з елементів стовпців заданого масиву за умовою індивідуального завдання з таблиці 3.

|  |  |
| --- | --- |
| Варіант | Завдання |
| 8 | З добутків елементів, які більше своїх сусідів. |

# **Тести і алгоритм (блок-схема) розв’язання задачі.**

Тести:

Таблиця 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва | Вхідні дані | | | | | | Очікуваний результат |
| Кількість рядків/стовпців не входить до проміжку[3;5] | N = 2 або M = 7  Та інші | | | | | | Повідомлення про помилку, поновлення вводу |
| Елемент не входить до проміжку[-32768;32767], або помилка вводу | 32768 або -111111 | | | | | | Повідомлення про помилку, поновлення вводу |
| Добуток елементів стовпчика, які більше за усіх своїх сусідів, не входить до проміжку[-32768;32767] | 1 | | 1 | | 32000 | | Повідомлення про помилку, завершення програми |
| 1 | | 1 | | 1 | |
| 1 | | 1 | | 32000 | |
|  | | | | | | | |
| Добуток елементів стовпчика, які більше за усіх своїх сусідів, не входить до проміжку[-32768;32767] .Приклад 2 | -32000 | -32001 | | -32000 | | -32000 | Повідомлення про помилку, завершення програми |
| -32001 | -32001 | | -32000 | | -32000 |
| -32000 | -32001 | | -32000 | | -32000 |
| -32001 | -32001 | | -32000 | | -32000 |
| -32000 | -32001 | | -32000 | | -32000 |

Алгоритми:

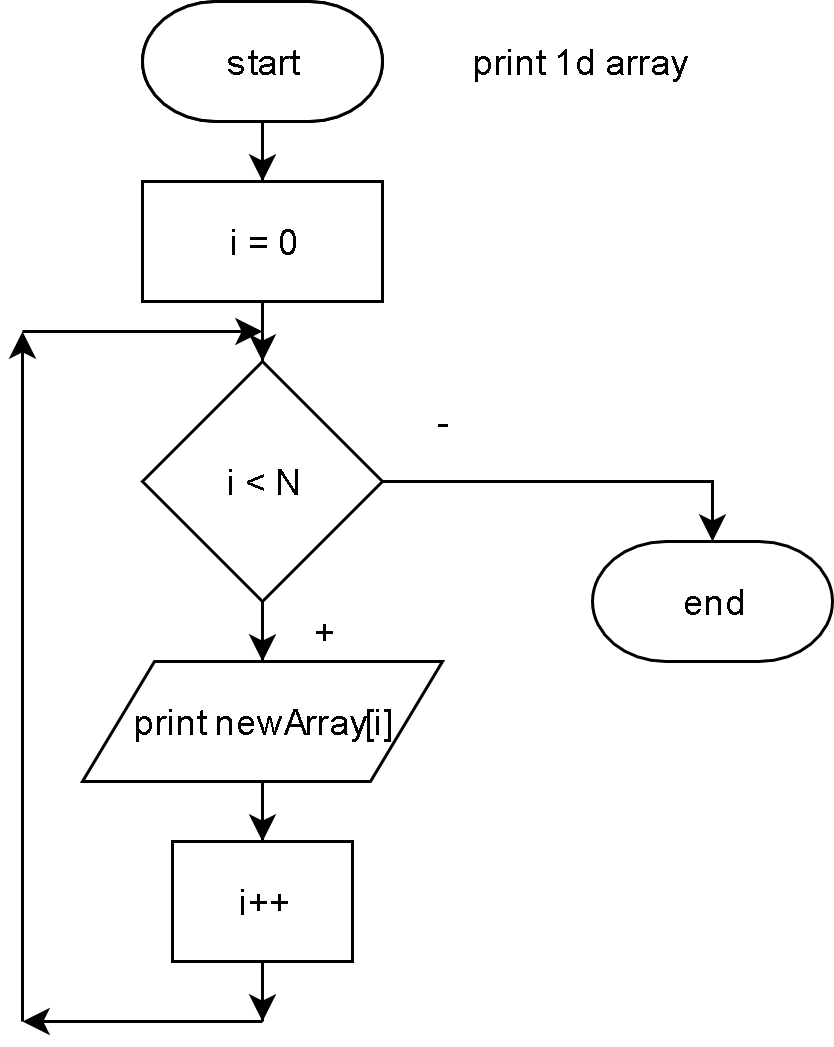


Рисунок 1

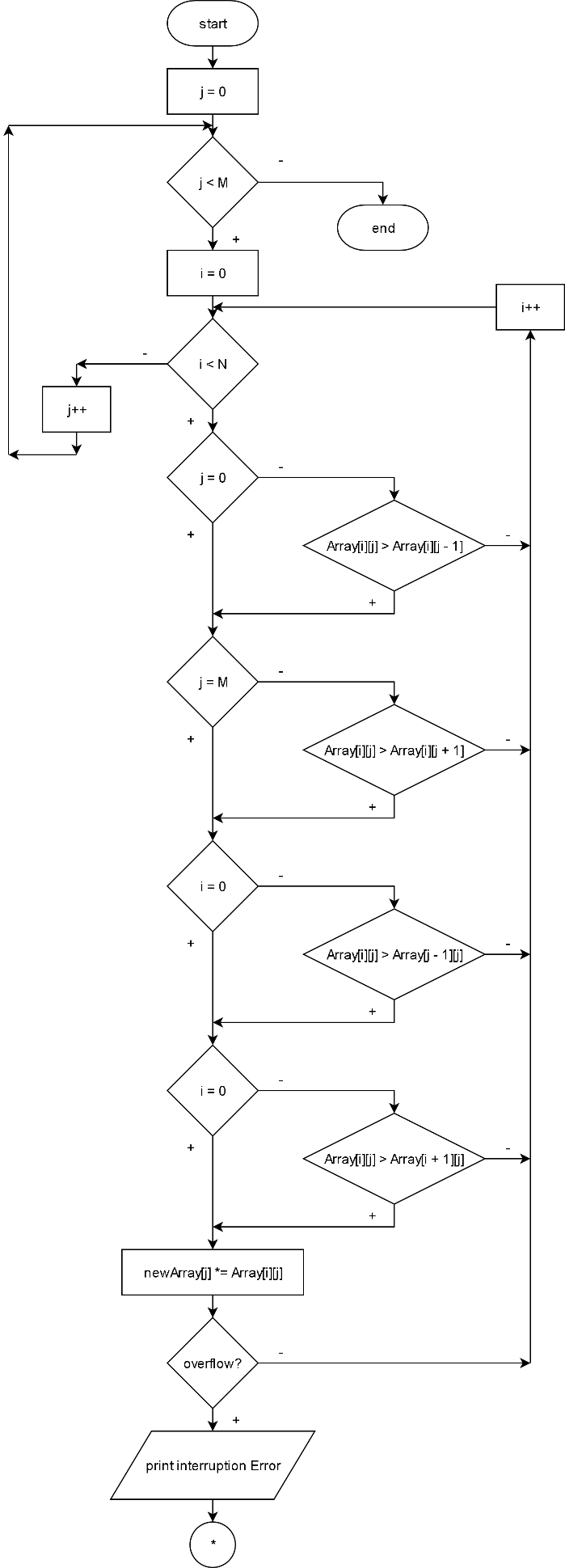


Рисунок 2

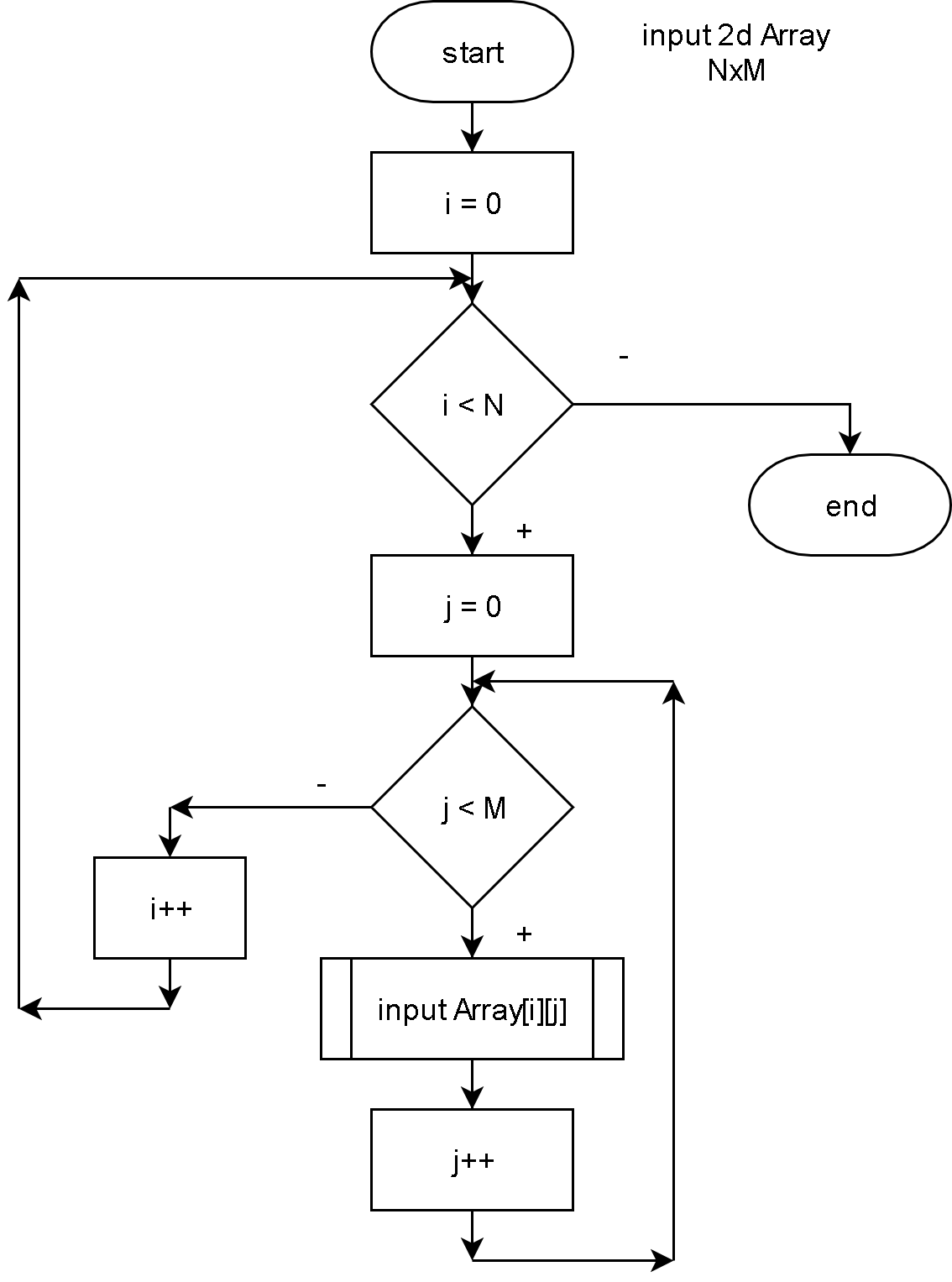


Рисунок 3

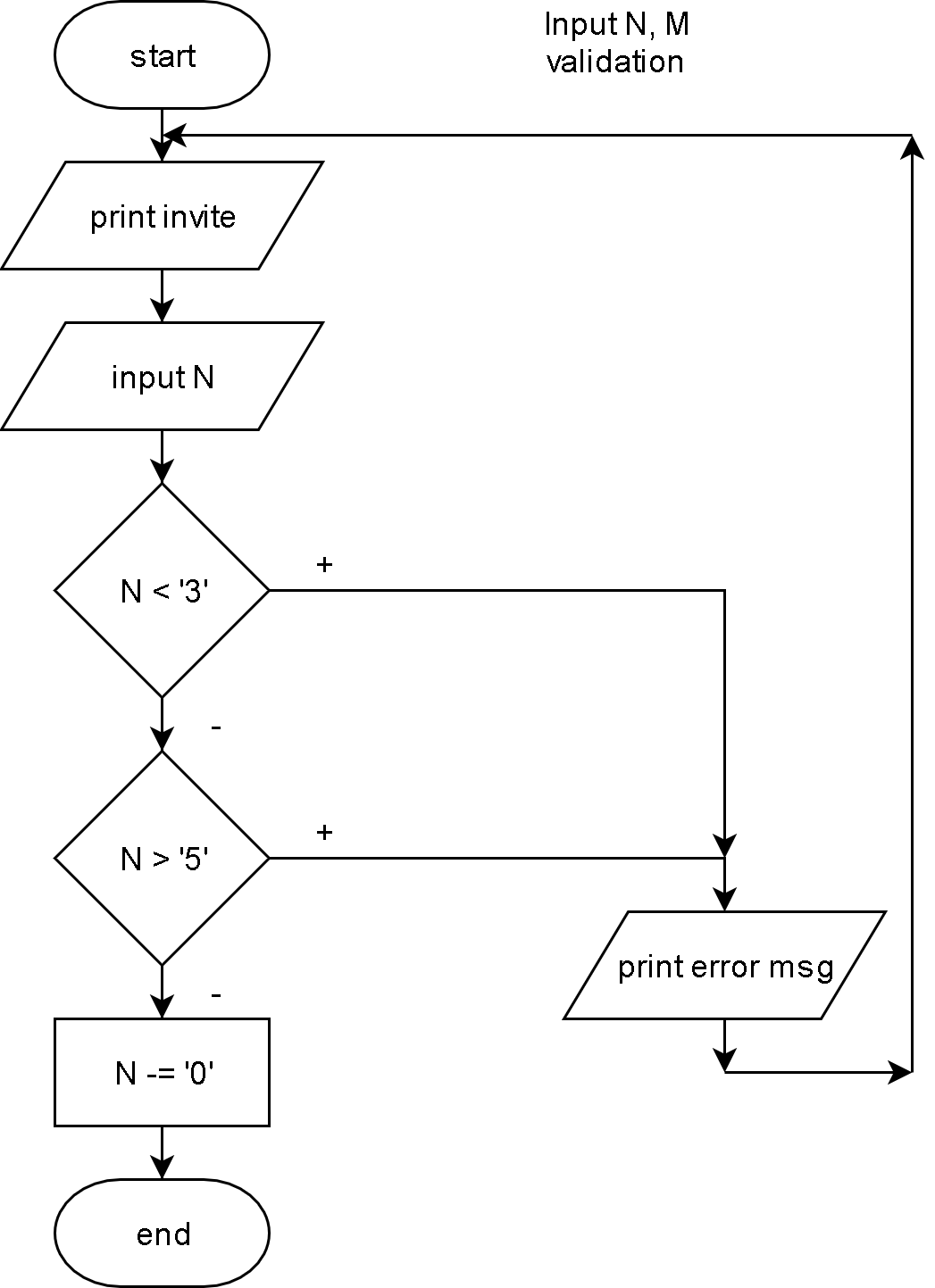


Рисунок 4

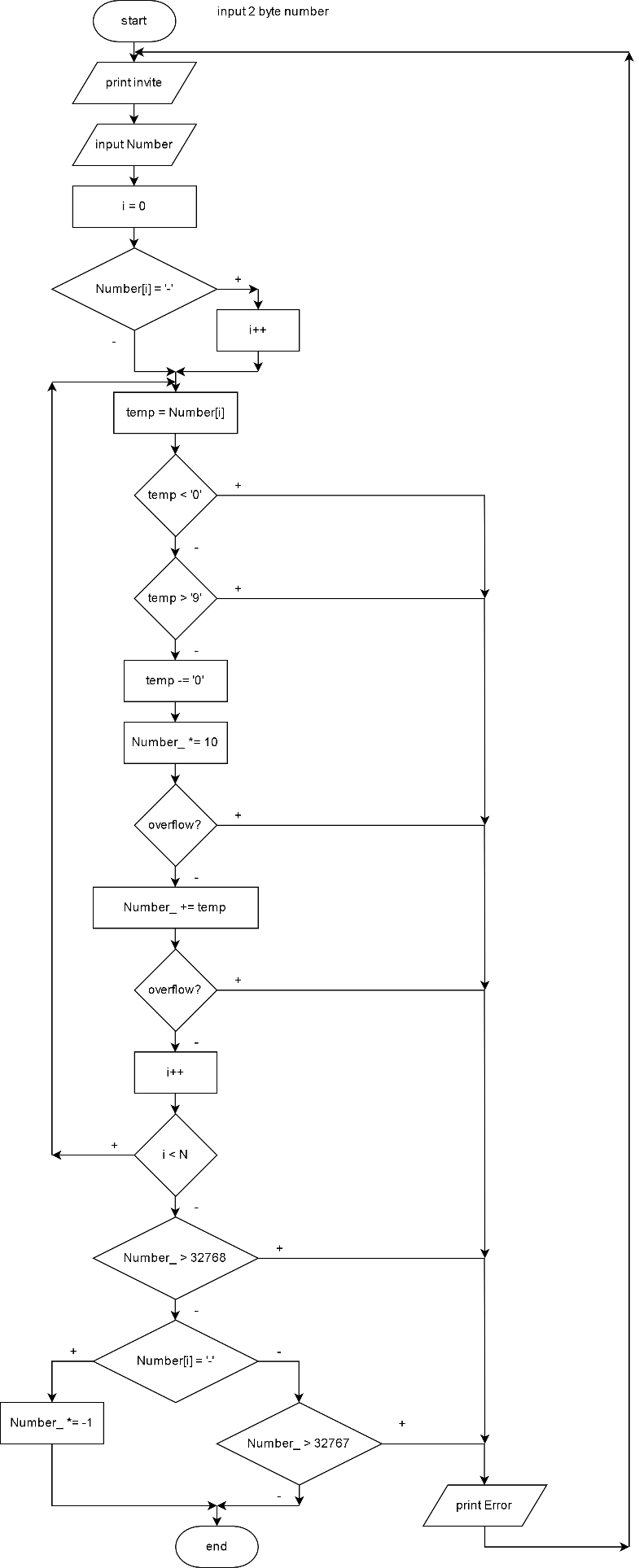


Рисунок 5

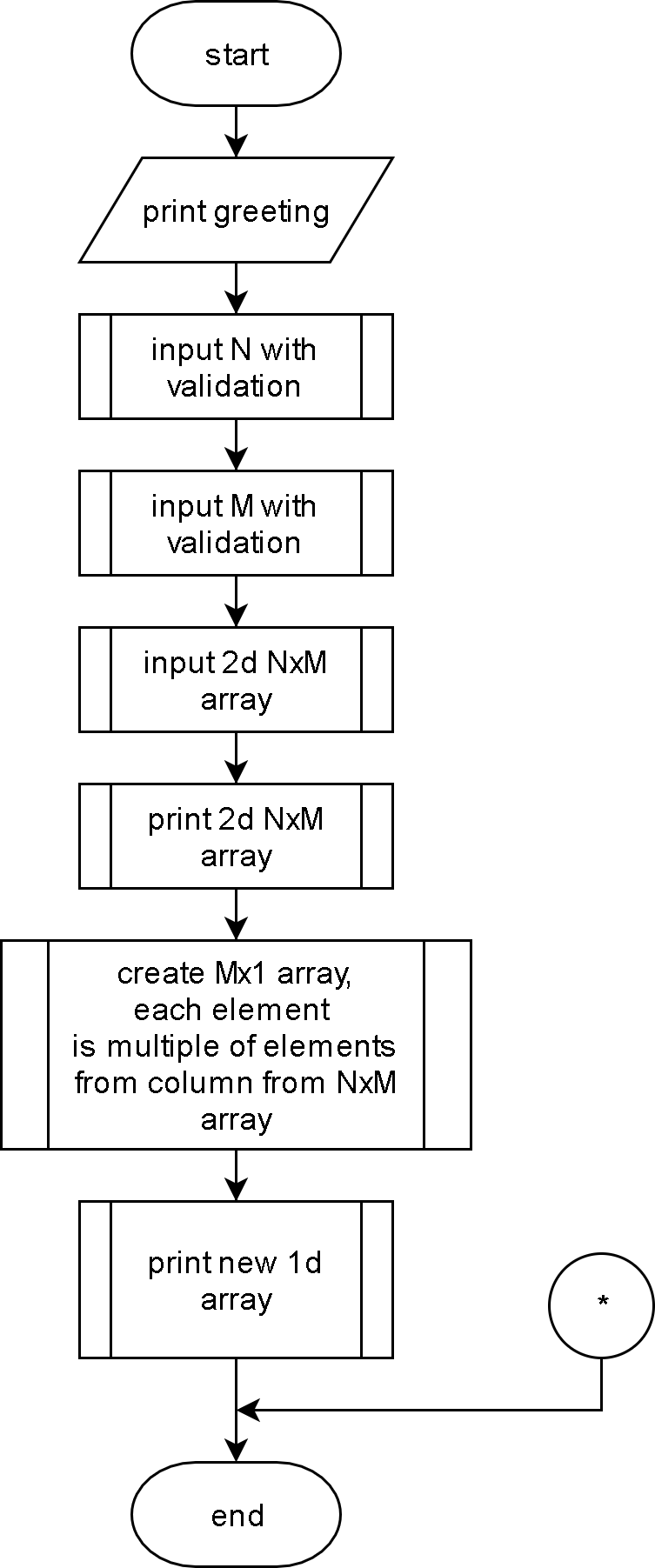


Рисунок 6

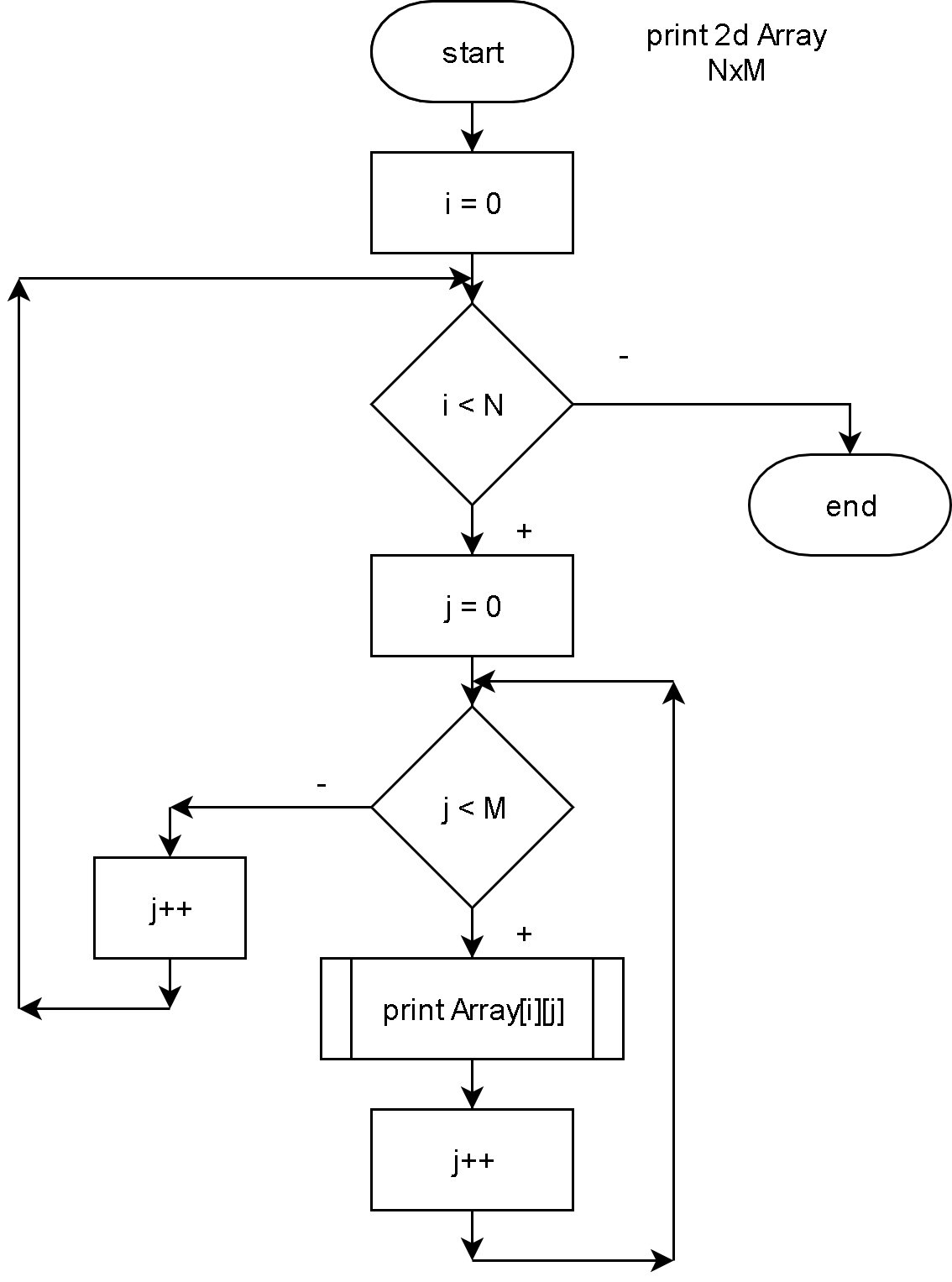


Рисунок 7

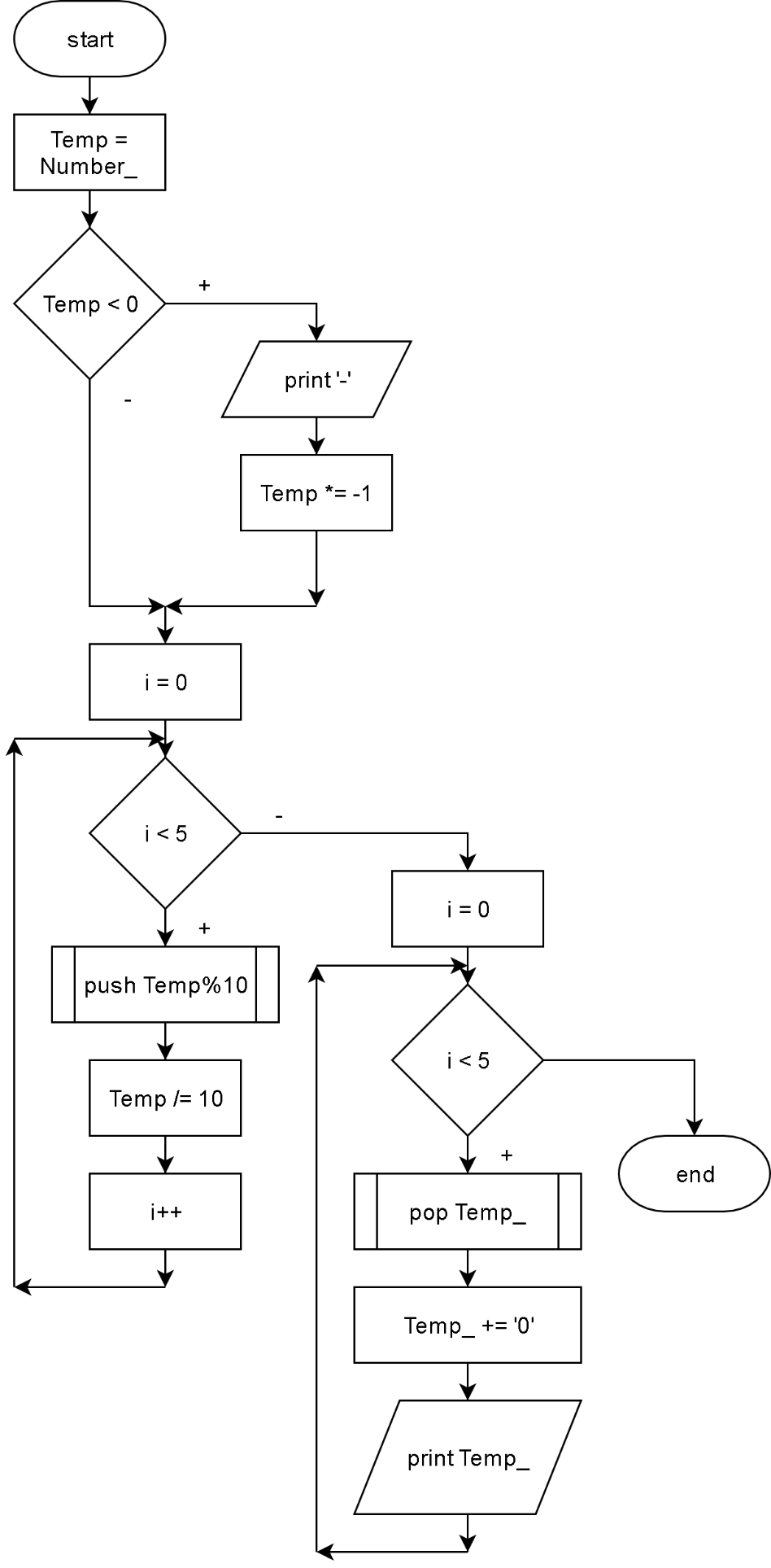


Рисунок 8

# **Лістинг програми.**

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 1

6p2\_v1.asm

1 0000 sseg segment stack 'stack'

2 0000 80\*(??) db 128 dup(?)

3 0080 sseg ends

4 ;-----------------------------------------------------------------------------

5 0000 dseg segment 'data'

6 0000 05\*(????) Array dw 5 dup(?);row1

7 000A 05\*(????) dw 5 dup(?);row2

8 0014 05\*(????) dw 5 dup(?);row3

9 001E 05\*(????) dw 5 dup(?);row4

10 0028 05\*(????) dw 5 dup(?);row5

11

12 0032 07 ?? 05\*(??) 24 24 buffer db 7, ?, 5 dup(?), '$', '$'

13 003B 0D 0A 24 msg\_endl db 0dh, 0ah, '$'

14 003E 49 6E 70 75 74 20 6E+ msg\_invite db "Input number: ", '$'

15 75 6D 62 65 72 3A 20+

16 24

17

18 004D 66 N db 'f';amount of rows [1;5]

19 004E 66 M db 'f';amount of columns [1;5]

20

21 004F ?? Nr db ?

22 0050 ?? Mr db ?;for proper comparing

23

24

25

26 0051 0D 0A 57 72 6F 6E 67+ msg\_error db 0dh, 0ah, "Wrong input!", 0dh, 0ah, '$'

27 20 69 6E 70 75 74 21+

28 0D 0A 24

29

30 0062 49 6E 70 75 74 20 4E+ msg\_invN db "Input N: ", '$'

31 3A 20 24

32 006C 49 6E 70 75 74 20 4D+ msg\_invM db "Input M: ", '$'

33 3A 20 24

34

35 0076 05\*(0001) newArray dw 5 dup(1);array with elements according to each row +

36 of 'Array',

37 ;each element is multiple of elements that are bigger

38 ;than their neighbours(virtical and horizontal)

39

40 0080 41 70 70 6C 69 63 61+ msg\_interrupt db "Application interrupted because of result overflow!", 0dh, 0ah, '$'

41 74 69 6F 6E 20 69 6E+

42 74 65 72 72 75 70 74+

43 65 64 20 62 65 63 61+

44 75 73 65 20 6F 66 20+

45 72 65 73 75 6C 74 20+

46 6F 76 65 72 66 6C 6F+

47 77 21 0D 0A 24

48

49 00B6 20 20 54 68 69 73 20+ msg\_salute0 db " This application takes N,M from range [3;5],", 0dh, 0ah, +

50 61 70 70 6C 69 63 61+ '$'

51 74 69 6F 6E 20 74 61+

52 6B 65 73 20 4E 2C 4D+

53 20 66 72 6F 6D 20 72+

54 61 6E 67 65 20 5B 33+

55 3B 35 5D 2C 0D 0A 24

56 00E7 20 20 67 65 74 73 20+ msg\_salute1 db " gets input for each element of NxM array in range +

57 69 6E 70 75 74 20 66+ [-32768;32767],", 0dh, 0ah, '$'

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 2

6p2\_v1.asm

58 6F 72 20 65 61 63 68+

59 20 65 6C 65 6D 65 6E+

60 74 20 6F 66 20 4E 78+

61 4D 20 61 72 72 61 79+

62 20 69 6E 20 72 61 6E+

63 67 65 20 5B 2D 33 32+

64 37 36 38 3B 33 32 37+

65 36 37 5D 2C 0D 0A 24

66 012D 20 20 61 6E 64 20 63+ msg\_salute2 db " and checks each column for elements which are larger", +

67 68 65 63 6B 73 20 65+ 0dh, 0ah, '$'

68 61 63 68 20 63 6F 6C+

69 75 6D 6E 20 66 6F 72+

70 20 65 6C 65 6D 65 6E+

71 74 73 20 77 68 69 63+

72 68 20 61 72 65 20 6C+

73 61 72 67 65 72 0D 0A+

74 24

75 0166 20 20 74 68 61 6E 20+ msg\_salute3 db " than their vertical and horizontal neighbours,", 0dh, +

76 74 68 65 69 72 20 76+ 0ah, '$'

77 65 72 74 69 63 61 6C+

78 20 61 6E 64 20 68 6F+

79 72 69 7A 6F 6E 74 61+

80 6C 20 6E 65 69 67 68+

81 62 6F 75 72 73 2C 0D+

82 0A 24

83 0199 20 20 63 72 65 61 74+ msg\_salute4 db " creates Mx1 array elements of which", 0dh, 0ah, '$'

84 65 73 20 4D 78 31 20+

85 61 72 72 61 79 20 65+

86 6C 65 6D 65 6E 74 73+

87 20 6F 66 20 77 68 69+

88 63 68 0D 0A 24

89 01C1 20 20 61 72 65 20 6D+ msg\_salute5 db " are multiples of chosen elements for each column(or (1) +

90 75 6C 74 69 70 6C 65+ if none).", 0dh, 0ah, 0dh, 0ah, '$'

91 73 20 6F 66 20 63 68+

92 6F 73 65 6E 20 65 6C+

93 65 6D 65 6E 74 73 20+

94 66 6F 72 20 65 61 63+

95 68 20 63 6F 6C 75 6D+

96 6E 28 6F 72 20 28 31+

97 29 20 69 66 20 6E 6F+

98 6E 65 29 2E 0D 0A 0D+

99 0A 24

100

101 0209 0D 0A 52 65 73 75 6C+ msg\_result db 0dh, 0ah, "Resulting array:", 0dh, 0ah, '$'

102 74 69 6E 67 20 61 72+

103 72 61 79 3A 0D 0A 24

104

105 021E 41 5B 24 msg\_r1 db "A[", '$'

106 0221 5D 3D 20 24 msg\_r2 db "]= ", '$'

107 0225 dseg ends

108 ;-----------------------------------------------------------------------------

109 0000 cseg segment 'code'

110 assume cs:cseg, ds:dseg, ss:sseg

111

112 0000 B8 0000s start: mov ax, dseg

113 0003 8E D8 mov ds, ax

114 ;main+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 3

6p2\_v1.asm

115

116 0005 BA 00B6r lea dx, msg\_salute0

117 0008 B4 09 mov ah, 09h

118 000A CD 21 int 21h

119

120 000C BA 00E7r lea dx, msg\_salute1

121 000F CD 21 int 21h

122

123 0011 BA 012Dr lea dx, msg\_salute2

124 0014 CD 21 int 21h

125

126 0016 BA 0166r lea dx, msg\_salute3

127 0019 CD 21 int 21h

128

129 001B BA 0199r lea dx, msg\_salute4

130 001E CD 21 int 21h

131

132 0020 BA 01C1r lea dx, msg\_salute5

133 0023 CD 21 int 21h

134

135 0025 EB 08 90 jmp inp\_N

136 0028 ErrorN:

137 0028 BA 0051r lea dx, msg\_error

138 002B B4 09 mov ah, 09h

139 002D CD 21 int 21h

140

141 002F inp\_N:

142 002F BA 0062r lea dx, msg\_invN

143 0032 B4 09 mov ah, 09h

144 0034 CD 21 int 21h;invite

145

146 0036 B4 01 mov ah, 01h

147 0038 CD 21 int 21h;al = symbol Input

148

149 003A 50 push ax;09h puts '$' to al

150 003B BA 003Br lea dx, msg\_endl

151 003E B4 09 mov ah, 09h

152 0040 CD 21 int 21h;endl

153 0042 58 pop ax;restore

154

155 0043 3C 33 cmp al, '3'

156 0045 72 E1 jb ErrorN;if <3

157 0047 3C 35 cmp al, '5'

158 0049 77 DD ja ErrorN;if >5

159

160 004B 2C 30 sub al, '0';symbol to digit

161

162 004D A2 004Dr mov N, al;storing N

163 0050 FE C8 dec al

164 0052 B3 0A mov bl, 10

165 0054 F6 E3 mul bl;al = n\*10

166 0056 A2 004Fr mov Nr, al

167

168 0059 EB 08 90 jmp inp\_M

169 005C ErrorM:

170 005C BA 0051r lea dx, msg\_error

171 005F B4 09 mov ah, 09h

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 4

6p2\_v1.asm

172 0061 CD 21 int 21h

173

174 0063 inp\_M:

175 0063 BA 006Cr lea dx, msg\_invM

176 0066 B4 09 mov ah, 09h

177 0068 CD 21 int 21h;invite

178

179 006A B4 01 mov ah, 01h

180 006C CD 21 int 21h;al = symbol Input

181

182 006E 50 push ax;09h puts '$' to al

183 006F BA 003Br lea dx, msg\_endl

184 0072 B4 09 mov ah, 09h

185 0074 CD 21 int 21h;endl

186 0076 58 pop ax;restore

187

188 0077 3C 33 cmp al, '3'

189 0079 72 E1 jb ErrorM;if <3

190 007B 3C 35 cmp al, '5'

191 007D 77 DD ja ErrorM;if >5

192

193 007F 2C 30 sub al, '0';symbol to digit

194

195 0081 A2 004Er mov M, al;storing M

196 0084 FE C8 dec al

197 0086 B3 02 mov bl, 2

198 0088 F6 E3 mul bl;al = M\*2

199 008A A2 0050r mov Mr, al

200

201 008D 8A 16 004Dr mov dl, N

202 0091 80 C2 30 add dl, '0';digit to symbol

203 0094 B4 02 mov ah, 02h;prints N

204 0096 CD 21 int 21h

205

206 0098 B2 78 mov dl, 'x'

207 009A CD 21 int 21h;prints 'x'

208

209 009C 8A 16 004Er mov dl, M

210 00A0 80 C2 30 add dl, '0';digit to symbol

211 00A3 CD 21 int 21h;prints M

212

213 00A5 BA 003Br lea dx, msg\_endl

214 00A8 B4 09 mov ah, 09h

215 00AA CD 21 int 21h;endl

216

217

218 00AC BB 0000r lea bx, Array

219 00AF 8A 16 004Dr mov dl, N

220 00B3 8A 36 004Er mov dh, M

221 00B7 BF 0051r lea di, msg\_error

222 00BA E8 01A5 call inp\_2d\_ar

223

224 00BD 52 push dx

225 00BE BA 003Br lea dx, msg\_endl

226 00C1 B4 09 mov ah, 09h

227 00C3 CD 21 int 21h

228 00C5 5A pop dx

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 5

6p2\_v1.asm

229

230 00C6 BF 003Br lea di, msg\_endl

231 00C9 E8 00AC call print\_2d\_ar

232

233

234

235 00CC 33 DB xor bx, bx;i = 0

236 00CE 8A 0E 004Er mov cl, M;i from 0 to M

237 00D2 32 ED xor ch, ch

238 00D4 new\_loop\_row:

239 00D4 33 F6 xor si, si;j = 0

240 00D6 B8 0001 mov ax, 1;current element of 'newArray'

241

242 00D9 51 push cx

243 00DA 8A 0E 004Dr mov cl, N;j from 0 to N

244 00DE new\_loop\_column:

245

246 00DE 8B 90 0000r mov dx, Array[bx][si];for checking and multiplying

247

248 00E2 80 FB 00 cmp bl, 0;if left

249 00E5 74 06 je left

250 00E7 3B 90 FFFEr cmp dx, Array[bx][si - 2];left

251 00EB 7E 2C jle break\_new\_loop\_column

252 00ED left:

253

254 00ED 3A 1E 0050r cmp bl, Mr;if right

255 00F1 74 06 je right

256 00F3 3B 90 0002r cmp dx, Array[bx][si + 2];right

257 00F7 7E 20 jle break\_new\_loop\_column

258 00F9 right:

259

260 00F9 83 FE 00 cmp si, 0;if up

261 00FC 74 06 je up

262 00FE 3B 90 FFF6r cmp dx, Array[bx - 10][si];up

263 0102 7E 15 jle break\_new\_loop\_column

264 0104 up:

265

266 0104 50 push ax

267 0105 A0 004Fr mov al, Nr

268 0108 32 E4 xor ah, ah

269

270 010A 3B F0 cmp si, ax;if down

271 010C 58 pop ax

272 010D 74 06 je down

273 010F 3B 90 000Ar cmp dx, Array[bx + 10][si];down

274 0113 7E 04 jle break\_new\_loop\_column

275 0115 down:

276

277 0115 F7 EA imul dx;current\_new\_element \* good\_old\_element

278 0117 72 12 jc carry\_interrupt;if result > 2bytes

279

280 0119 break\_new\_loop\_column:

281 0119 83 C6 0A add si, 10;j++

282 011C E2 C0 loop new\_loop\_column

283 011E 59 pop cx

284

285 011F 89 87 0076r mov newArray[bx], ax

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 6

6p2\_v1.asm

286

287 0123 83 C3 02 add bx, 2;i++

288 0126 E2 AC loop new\_loop\_row

289

290 0128 EB 0B 90 jmp print\_1d

291 012B carry\_interrupt:

292 012B BA 0080r lea dx, msg\_interrupt

293 012E B4 09 mov ah, 09h

294 0130 CD 21 int 21h

295

296 0132 EB 40 90 jmp final

297

298

299 0135 print\_1d:

300

301 0135 BA 0209r lea dx, msg\_result

302 0138 B4 09 mov ah, 09h

303 013A CD 21 int 21h

304

305 013C 32 ED xor ch, ch

306 013E 8A 0E 004Er mov cl, M

307 0142 33 F6 xor si, si;i from 0 to M

308 0144 B3 02 mov bl, 2

309

310 0146 print\_1d\_loop:

311

312 0146 BA 021Er lea dx, msg\_r1

313 0149 B4 09 mov ah, 09h

314 014B CD 21 int 21h

315

316 014D 8B C6 mov ax, si

317 014F F6 F3 div bl;2i/2 = i = al

318 0151 8A D0 mov dl, al

319 0153 80 C2 31 add dl, '1';digit+1 to symbol

320 0156 B4 02 mov ah, 02h

321 0158 CD 21 int 21h;prints i

322

323 015A BA 0221r lea dx, msg\_r2

324 015D B4 09 mov ah, 09h

325 015F CD 21 int 21h

326

327 0161 8B 84 0076r mov ax, newArray[si]

328 0165 E8 0048 call num\_print\_word

329

330 0168 BA 003Br lea dx, msg\_endl

331 016B B4 09 mov ah, 09h

332 016D CD 21 int 21h;endl

333

334 016F 83 C6 02 add si, 2;i++

335 0172 E2 D2 loop print\_1d\_loop

336

337 0174 final:

338 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

339 0174 B4 4C mov ah, 4Ch

340 0176 CD 21 int 21h

341 ;2d\_array\_processing++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

342 ;in:

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 7

6p2\_v1.asm

343 ;bx = pointer to array

344 ;N = dl

345 ;M = dh

346

347 ;out: -

348

349 ;bx = array pointer(later poinet to row)

350 ;si = pointer to column

351 ;di - \*msg\_endl

352

353 ;uses ax, bx, cx, dx, si, di

354

355 0178 print\_2d\_ar PROC near

356

357 0178 50 push ax

358 0179 53 push bx

359 017A 51 push cx

360 017B 52 push dx

361 017C 56 push si

362 017D 57 push di;storing

363

364 017E 8A CA mov cl, dl

365 0180 32 ED xor ch, ch;i = N

366 0182 loop\_i:

367 0182 33 F6 xor si, si

368 0184 51 push cx;store outer counter to use cx for inner counter

369

370 0185 8A CE mov cl, dh

371 0187 32 ED xor ch, ch;j = M

372

373 0189 loop\_j:

374 0189 52 push dx

375 018A 8B 00 mov ax, [bx][si];array[i][j]

376 018C E8 0021 call num\_print\_word;prints array[i][j]

377

378 018F B2 20 mov dl, ' '

379 0191 B4 02 mov ah, 02h

380 0193 CD 21 int 21h

381 0195 5A pop dx

382

383 0196 83 C6 02 add si, 2;j++(2bytes)

384 0199 E2 EE loop loop\_j

385

386 019B 59 pop cx;restore outer clock after inner clock fadeout

387

388 019C 52 push dx

389 019D 8B D7 mov dx, di

390 019F B4 09 mov ah, 09h

391 01A1 CD 21 int 21h

392 01A3 5A pop dx

393

394 01A4 83 C3 0A add bx, 10;i++, 10 = 5(actualsize)\*2(bytes)

395 01A7 E2 D9 loop loop\_i

396

397 01A9 5F pop di

398 01AA 5E pop si

399 01AB 5A pop dx

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 8

6p2\_v1.asm

400 01AC 59 pop cx

401 01AD 5B pop bx

402 01AE 58 pop ax;restoring

403

404 01AF C3 RET

405 01B0 print\_2d\_ar ENDP

406 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

407 ;word\_print\_proc++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

408

409 ;prints signed number from ax

410 ;uses ax, bx, cx, dx

411

412 ;in: ax - number

413

414 ;out: -

415

416 01B0 num\_print\_word PROC near

417

418 01B0 50 push ax

419 01B1 53 push bx

420 01B2 51 push cx

421 01B3 52 push dx;storing

422

423 01B4 B9 0005 mov cx, 5;max 5 digits in 2byte number

424 01B7 BB 000A mov bx, 10;for dividing

425

426 01BA 3D 0000 cmp ax, 0

427 01BD 7D 0A jge additive\_stack

428 01BF 50 push ax;int 21h, ah = 02h, returns last printed char to al

429

430 01C0 B2 2D mov dl, '-'

431 01C2 B4 02 mov ah, 02h

432 01C4 CD 21 int 21h;prints '-'

433

434 01C6 58 pop ax

435

436 01C7 F7 D8 neg ax

437 01C9 additive\_stack:

438 01C9 33 D2 xor dx, dx;dx:ax = ax

439 01CB F7 F3 div bx;ax = dx:ax/10, dx = dx:ax % 10

440 01CD 52 push dx;storing dx with first digit from right

441

442 01CE E2 F9 loop additive\_stack

443

444 01D0 B9 0005 mov cx, 5

445

446 01D3 print:

447

448 01D3 5A pop dx;dl= symbol to print

449 01D4 80 C2 30 add dl, 30h;ascii

450 01D7 B4 02 mov ah, 02h

451 01D9 CD 21 int 21h;prints i-th digit from left

452

453 01DB E2 F6 loop print

454

455 01DD 5A pop dx

456 01DE 59 pop cx

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 9

6p2\_v1.asm

457 01DF 5B pop bx

458 01E0 58 pop ax;restoring

459

460 01E1 C3 RET

461 01E2 num\_print\_word ENDP

462 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

463 ;num\_input\_word+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

464

465 ;inputs symbolic to buffer

466 ;and transforms it into

467 ;2byte signed numeric value in ax

468

469 ;si - adress of buffer(9 bytes)

470 ;1st - max size(7 to check if input is correct [-32768;32767] c Z)

471 ;2nd - actual size(if length = 7, input.fail)

472 ;3rd - + or - sign, or digit

473 ;4th-7th - actual digits left to rigt

474 ;8th, 9th - '$', if after inputs its not $entinel, input.fail

475

476 ;uses:

477 ;si, ax, bx, cx, dx

478 ;si - pointer to string(later to Nth digit)

479 ;ax - number

480 ;bx -

481 ;cx - counter

482

483 ;in:

484 ;\*buffer = dx

485 ;\*msg\_endl = \*(buffer) + 9

486 ;\*msg\_invite = \*(msg\_endl) + 3 = \*(buffer) + 12

487 ;\*msg\_error = di

488

489 ;out:

490 ;ax = number

491

492 01E2 num\_input\_word PROC near

493 01E2 53 push bx

494 01E3 51 push cx

495 01E4 52 push dx

496 01E5 56 push si

497 01E6 57 push di;storing

498

499 01E7 32 ED xor ch, ch;cx = cl

500 01E9 32 FF xor bh, bh;bx = bl

501

502 01EB EB 09 90 jmp input

503 01EE error\_inw:

504 01EE 52 push dx

505 01EF 8B D7 mov dx, di

506 01F1 B4 09 mov ah, 09h

507 01F3 CD 21 int 21h;error

508 01F5 5A pop dx

509

510 01F6 input:

511 01F6 8B F2 mov si, dx;si points to buffer

512

513 01F8 52 push dx;store

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 10

6p2\_v1.asm

514 01F9 83 C2 09 add dx, 09h

515 01FC B4 09 mov ah, 09h

516 ;int 21h;std::endl

517

518 01FE 83 C2 03 add dx, 03h

519 0201 CD 21 int 21h;prints message

520 0203 5A pop dx;restore

521 0204 B4 0A mov ah, 0ah

522 0206 CD 21 int 21h;input string to buffer

523

524 0208 33 C0 xor ax, ax;number = 0

525 020A 8A 4C 01 mov cl, [si + 1];i = actual length of buffer

526 020D 83 C6 02 add si, 2;si = pointer to first actual element of string

527

528 0210 80 3C 2D cmp byte ptr [si], '-';checking if first element is SIGN(-)

529 0213 75 02 jne number\_assembly

530 0215 46 inc si;first actual element of element after '-'

531 0216 49 dec cx;one iteration less

532 0217 number\_assembly:

533

534 0217 8A 1C mov bl, [si];just for checking

535

536 0219 80 FB 30 cmp bl, '0'

537 021C 72 D0 jb error\_inw

538 021E 80 FB 39 cmp bl, '9'

539 0221 77 CB ja error\_inw

540 0223 80 EB 30 sub bl, '0';making digit from symbol

541

542 0226 52 push dx;store

543 0227 57 push di

544 0228 BF 000A mov di, 10

545 022B F7 E7 mul di;moving current state of number to left to add +

546 new digit

547 022D 5F pop di

548 022E 5A pop dx;restore

549

550 022F 72 BD jc error\_inw;if larger than 2 bytes

551 0231 03 C3 add ax, bx

552 0233 72 B9 jc error\_inw

553 0235 46 inc si;next digit

554 0236 E2 DF loop number\_assembly

555

556 0238 3D 8000 cmp ax, 32768

557 023B 77 B1 ja error\_inw

558 023D 8B DA mov bx, dx

559 023F 83 C3 02 add bx, 2

560 0242 80 3F 2D cmp byte ptr [bx], '-'

561 0245 75 05 jne pos\_

562 0247 F7 D8 neg ax;n \*= -1;

563 0249 EB 06 90 jmp end\_

564 024C pos\_:

565 024C 3D 7FFF cmp ax, 32767

566 024F 77 9D ja error\_inw

567 0251 end\_:

568

569 0251 52 push dx

570 0252 50 push ax

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 11

6p2\_v1.asm

571 0253 83 C2 09 add dx, 9

572 0256 B4 09 mov ah, 09h

573 0258 CD 21 int 21h;endl

574 025A 58 pop ax

575 025B 5A pop dx

576

577 025C 5F pop di

578 025D 5E pop si

579 025E 5A pop dx

580 025F 59 pop cx

581 0260 5B pop bx;restoring

582

583 0261 C3 RET

584 0262 num\_input\_word ENDP

585 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

586 ;2d\_array\_inp++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

587 ;in:

588 ;bx = \*Array

589 ;di = \*msg\_endl

590 ;N = dl

591 ;M = dh

592

593 ;out: -

594

595 ;bx = array pointer(later pointer to row)

596 ;si = pointer to column

597

598

599 ;uses ax, bx, cx, dx, si, di

600

601 0262 inp\_2d\_ar PROC near

602

603 0262 50 push ax

604 0263 53 push bx

605 0264 51 push cx

606 0265 52 push dx

607 0266 56 push si

608 0267 57 push di;storing

609

610 0268 8B C3 mov ax, bx;ax = \*Array

611 026A 8A CA mov cl, dl

612 026C 32 ED xor ch, ch;i = N

613 026E loop\_i1:

614 026E 33 F6 xor si, si

615 0270 51 push cx;store outer counter to use cx for inner counter

616

617 0271 8A CE mov cl, dh

618 0273 32 ED xor ch, ch;j = M

619

620 0275 loop\_j1:

621

622 0275 52 push dx

623 0276 53 push bx

624 0277 50 push ax

625 0278 B2 5B mov dl, '['

626 027A B4 02 mov ah, 02h

627 027C CD 21 int 21h

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 12

6p2\_v1.asm

628

629 027E 8B C3 mov ax, bx

630 0280 5B pop bx;bx = \*Array

631 0281 53 push bx;storing \*Array

632 0282 2B C3 sub ax, bx

633 0284 B3 0A mov bl, 10

634 0286 F6 F3 div bl;al = i

635

636 0288 8A D0 mov dl, al

637 028A 80 C2 31 add dl, '1';digit+1 to symbol

638 028D B4 02 mov ah, 02h

639 028F CD 21 int 21h

640

641 0291 B2 5D mov dl, ']'

642 0293 CD 21 int 21h

643

644 0295 B2 5B mov dl, '['

645 0297 CD 21 int 21h

646

647 0299 8B C6 mov ax, si

648 029B B3 02 mov bl, 2

649 029D F6 F3 div bl;al = j

650

651 029F 8A D0 mov dl, al

652 02A1 80 C2 31 add dl, '1';digit+1 to symbol

653 02A4 B4 02 mov ah, 02h

654 02A6 CD 21 int 21h

655

656 02A8 B2 5D mov dl, ']'

657 02AA CD 21 int 21h

658 02AC 58 pop ax

659 02AD 5B pop bx

660 02AE 50 push ax

661

662 02AF B2 2E mov dl, '.'

663 02B1 B4 02 mov ah, 02h

664 02B3 CD 21 int 21h

665

666

667 02B5 5A pop dx;dx = \*Array

668 02B6 52 push dx;storing \*Array

669 02B7 83 C2 32 add dx, 50

670

671 02BA E8 FF25 call num\_input\_word;ax = input

672 02BD 89 00 mov word ptr [bx][si], ax;Array[i][j] = input

673 02BF 58 pop ax;ax = \*Array(restored)

674 02C0 5A pop dx

675

676 02C1 83 C6 02 add si, 2;j++(2bytes)

677 02C4 E2 AF loop loop\_j1

678

679 02C6 59 pop cx;restore outer clock after inner clock fadeout

680

681 02C7 83 C3 0A add bx, 10;i++, 10 = 5(actualsize)\*2(bytes)

682 02CA E2 A2 loop loop\_i1

683

684 02CC 5F pop di

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 13

6p2\_v1.asm

685 02CD 5E pop si

686 02CE 5A pop dx

687 02CF 59 pop cx

688 02D0 5B pop bx

689 02D1 58 pop ax;restoring

690

691 02D2 C3 RET

692 02D3 inp\_2d\_ar ENDP

693 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

694 02D3 cseg ends

695 end start

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 14

Symbol Table

Symbol Name Type Value

??DATE Text "12/15/19"

??FILENAME Text "6p2\_v1 "

??TIME Text "15:27:31"

??VERSION Number 0300

@CPU Text 0101H

@CURSEG Text CSEG

@FILENAME Text 6P2\_V1

@WORDSIZE Text 2

ADDITIVE\_STACK Near CSEG:01C9

ARRAY Word DSEG:0000

BREAK\_NEW\_LOOP\_COLUMN Near CSEG:0119

BUFFER Byte DSEG:0032

CARRY\_INTERRUPT Near CSEG:012B

DOWN Near CSEG:0115

END\_ Near CSEG:0251

ERRORM Near CSEG:005C

ERRORN Near CSEG:0028

ERROR\_INW Near CSEG:01EE

FINAL Near CSEG:0174

INPUT Near CSEG:01F6

INP\_2D\_AR Near CSEG:0262

INP\_M Near CSEG:0063

INP\_N Near CSEG:002F

LEFT Near CSEG:00ED

LOOP\_I Near CSEG:0182

LOOP\_I1 Near CSEG:026E

LOOP\_J Near CSEG:0189

LOOP\_J1 Near CSEG:0275

M Byte DSEG:004E

MR Byte DSEG:0050

MSG\_ENDL Byte DSEG:003B

MSG\_ERROR Byte DSEG:0051

MSG\_INTERRUPT Byte DSEG:0080

MSG\_INVITE Byte DSEG:003E

MSG\_INVM Byte DSEG:006C

MSG\_INVN Byte DSEG:0062

MSG\_R1 Byte DSEG:021E

MSG\_R2 Byte DSEG:0221

MSG\_RESULT Byte DSEG:0209

MSG\_SALUTE0 Byte DSEG:00B6

MSG\_SALUTE1 Byte DSEG:00E7

MSG\_SALUTE2 Byte DSEG:012D

MSG\_SALUTE3 Byte DSEG:0166

MSG\_SALUTE4 Byte DSEG:0199

MSG\_SALUTE5 Byte DSEG:01C1

N Byte DSEG:004D

NEWARRAY Word DSEG:0076

NEW\_LOOP\_COLUMN Near CSEG:00DE

NEW\_LOOP\_ROW Near CSEG:00D4

NR Byte DSEG:004F

NUMBER\_ASSEMBLY Near CSEG:0217

NUM\_INPUT\_WORD Near CSEG:01E2

NUM\_PRINT\_WORD Near CSEG:01B0

POS\_ Near CSEG:024C

Turbo Assembler Version 3.0 12/15/19 15:27:31 Page 15

Symbol Table

PRINT Near CSEG:01D3

PRINT\_1D Near CSEG:0135

PRINT\_1D\_LOOP Near CSEG:0146

PRINT\_2D\_AR Near CSEG:0178

RIGHT Near CSEG:00F9

START Near CSEG:0000

UP Near CSEG:0104

Groups & Segments Bit Size Align Combine Class

CSEG 16 02D3 Para none CODE

DSEG 16 0225 Para none DATA

SSEG 16 0080 Para Stack STACK

# **Результати виконання програми та їх аналіз.**

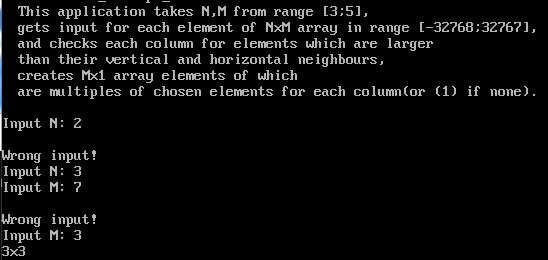


Рисунок 9

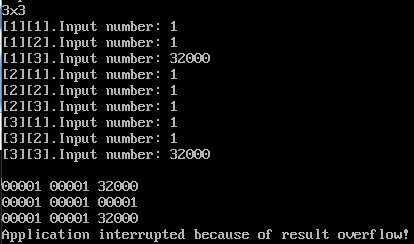


Рисунок 10

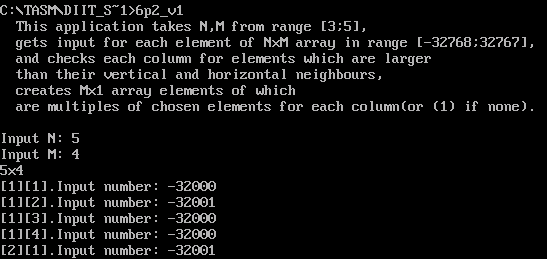


Рисунок 11

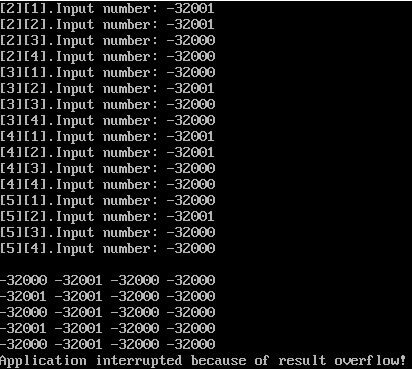


Рисунок 12

Результати відповідають очікуванням, програма працює корректно.

# **Висновки щодо перетворення символьних даних в числові та числових даних в символьні, роботи з матрицями в мові Ассемблера.**

Для введення та виведення десяткових чисел із декількох цифр треба обробляти кожен розряд числа.

Щоб ввести число треба перевірити чи е воно від’емним, потім циклічно додавати поточну цифру та множити число на 10(щоб здвинути на один розряд і з’явилося місце для наступної цифри), за потреби вкінці домножити на -1. При відповідних діях(множенні на 10, додаванні цифри), належно перевіряти чи не вийшло число за межі потрібного діапазону.

Для виведення перевірити знак, якщо мінус, то вивести на дісплей ‘-’ та домножити число на –1, циклічно ділимо на 10, зберігаємо залишок, доки результат від ділення не стане рівним нулю, виводимо залишки у зворотньому порядку. Я не став зупиняти збереження залишки через те, що таким чином запис числа буде відрізнятися тільки на один символ і при виведенні массиву у вигляді матриці це діє як свого роду вірівнювання.

Символи чисел в ASCII починаються з 30h = 48d, тож при введені треба віднімати 30h(або код ‘0’), а при введены додавати.

Звертатися до елементів матриця треба через вкладений цикл з лічильноком, можна помістити у один регістр(bx, si, di), у обидва рядки 0, тоді треба буде звертатися так:

Mov cx, N

Xor bx, bx

Loop\_i:

Push cx

Mov cx, M

Xor si, si

Loop\_j:

;Array[bx][si] або Array[bx + si]…

Add si, A;замість А розмір елемента у байтах

Add bx, B;замість B кількість стовпців ЗАРЕЗЕРВОВАНОЇ матриці помножена на розмір елемента

Loop Loop\_j

Pop cx

Loop Loop\_i

Або можна помістити у перший регістр адресу початку зарезервованої матриці:

Mov cx, N

Lea bx, Array

Loop\_i:

Push cx

Mov cx, M

Xor si, si

Loop\_j:

;[bx][si] або Array[bx + si]…

Add si, A;замість А розмір елемента у байтах

Add bx, B;замість B кількість стовпців ЗАРЕЗЕРВОВАНОЇ матриці помножена на розмір елемента

Loop Loop\_j

Pop cx

Loop Loop\_i

Перший спосіб трохи кращий для розуміння коду, другий доцільно використовувати у процедурах, через те, що в них не можна використовуати лейбли, тож замість Array, потрібно б було використовувати ще один регістр.