НАЦІОНАЛЬНИЙ ТЕХНІЧНЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіти до комп'ютерних практикумів дисципліни

«Системне програмне забезпечення»

Прийняв
доцент кафедри ІПІ
Лісовиченко О.І.
"31" травня 2024р.

Виконав
Студент групи ІП-24
Піддубний Б.С.

Київ 2024

Комп'ютерний практикум №3

Тема: програмування розгалужених алгоритмів

Завдання:

1. Можливість введення користувачем значень x, y, t, a, b за необхідності.
2. Обчислювати значення функції за введеними значеннями.
3. Виводити на екран результат обчислень.
4. Якщо є ділення, то результат дозволяється виводити:
    a. Як дійсне число (наприклад: $\frac{5}{3} = 1.6666667$) – підвищена складність;
    b. окремо цілу частину та остачу (наприклад: $\frac{5}{3} = 1$ остача 2) – середня складність
    c. окремо цілу частину та остачу як дріб (наприклад: $\frac{5}{3} = 1\frac{2}{3}$) – середня складність
    d. Процедура переведення отриманого результату в рядок та виведення його на екран.
5. Програма повинна мати захист від некоректного введення вхідних даних (символи, переповнення, ділення на 0 і т.і.).

Варіант 22:

$$Z = \begin{cases} x - 1, \text{якщо } x < -1 \\ 0, \text{якщо } x = -1 \\ \dfrac{x^3 + 2x^2 + 11}{2x + 1}, \text{якщо } x > -1 \end{cases}$$

Текст програми:

```
.8086
.model small

.stack 100h

.data
   header_msg         db 10, '--- Piecewise Function Calculator ---', 10, 10,
13, '$'
   function_msg       db '| x-1,                 if x < -1', 10, 13
                      db '| 0,                   if x = -1', 10, 13
                      db '| (x^3+2x^2+11)/(2x+1), if x > -1', 10, 10, 13, '$'
   prompt_msg         db 'Enter your X (from -32 768 to 32 767): ', '$'
   result_msg         db 'Result is: ', '$'
   remainder_msg      db 'Remainder is: ', '$'
   overflow_err_msg   db 'Error: Overflow happened.', 10, 13, '$'
   non_int_err_msg    db 'Error: You have entered non-integer value.', 10,
13, '$'
   incorrect_input_msg db 'Error: Incorrect input number.', 10, 13, '$'
   new_line           db 10, 13, "$"

   buffer             db 7, ?, 7 dup (?)
   number             dw 0
   result             dw 0
   remainder          dw 0
   minus_sign         db 0

.code
   start:
                      mov  ax, DGROUP
                      mov  ds, ax

                      lea  dx, header_msg
                      call print_line
                      lea  dx, function_msg
                      call print_line

process_input:
                      call process_number
```

```
                call process_function

                lea  dx, new_line
                call print_line
                lea  dx, result_msg
                call print_line

                mov  bx, result
                call print_number

                lea  dx, new_line
                call print_line
                lea  dx, remainder_msg
                call print_line

                mov  bx, remainder
                call print_number
   ; terminate program
                mov  ax, 4c00h
                int  21h

print_line proc
                mov  ah, 09h
                int  21h
                ret
print_line endp

process_number proc
   ; process_input:
                xor  ax, ax
                xor  cx, cx
                xor  dx, dx
                mov  minus_sign, 0

   ; print prompt
                lea  dx, prompt_msg
                call print_line

   ; receive user input using DOS function 21h, interruption 0ah
                lea  dx, buffer
```

```
                mov  ah, 0ah
                int  21h

    ; Process numerical input (digits 0-9)
                xor  ax, ax
                mov  cl, buffer + 1          ; Set the loop counter to the
number of characters to process
                mov  si, 2             ; Set the SI index to the first character
of number
                cmp  buffer + 2, '-'
                jne  process_digit

    ; Handle negative sign (if present)
                inc  si                    ; Increment SI to point to the first
digit after the minus sign
                dec  cl                    ; Decrement CL to account for the
minus sign (one less digit to process)
                mov  minus_sign, 1

    ; Convert ASCII to integer and check for errors
    process_digit:

    ; Handle non-integer values
                mov  bl, buffer[si]
                cmp  bl, '0'
                jb   handle_non_int
                cmp  bl, '9'
                ja   handle_non_int

    ; Convert ASCII to integer
                sub  bl, '0'

    ; Add together converted digits and check for overflow
                mov  dx, 10
                imul dx
                jo   handle_overflow
                add  ax, bx
                jo   handle_special_case

                inc  si
```

```
                loop process_digit
                jmp  check_sign

    handle_overflow:
                lea  dx, new_line
                call print_line
                lea  dx, overflow_err_msg
                call print_line
                jmp  process_input

    handle_non_int:
                lea  dx, new_line
                call print_line
                lea  dx, non_int_err_msg
                call print_line
                jmp  process_input

    handle_special_case:
                cmp  ax, -32768
                jne  handle_overflow
                cmp  minus_sign, 1
                jne  handle_overflow
                neg  ax
                jmp  return_number

    check_sign:
                cmp  minus_sign, 1
                jne  return_number
                neg  ax

    return_number:
                mov  number, ax

                ret
process_number endp

process_function proc

                xor  ax, ax
                xor  bx, bx
```

```
                xor  dx, dx

                mov  ax, number
                test ax, ax
                jns  third_operation

                cmp  ax, -1
                je   second_operation

    ; if x < -1
    ; result = x - 1
    first_operation:
                sub  ax, 1
                ; jo   handle_special_case

                jmp  return_result

    ; if x = -1
    ; result = 0
    second_operation:
                mov  ax, 0

                jmp  return_result

    ; if x > -1
    ; result = (x^3 + 2x^2 + 11) / (2x + 1)
    third_operation:
                mov  ax, number          ; Load number into ax
                mov  dx, ax              ; Copy ax to dx
                mul  dx                  ; ax = ax * dx (ax = number *
number)
                jo   handle_overflow

                mov  cx, ax              ; Move result to cx
                mov  ax, number          ; Load number into ax again
                mul  cx                  ; ax = ax * cx (ax = number * (number
* number)), ax - first term of numerator
                jo   handle_overflow
```

```
                add  cx, cx             ; cx = cx + cx (cx = 2 * (number *
number)), cx - second term of numenator
                add  ax, cx               ; ax = ax + cx (ax = (number *
(number * number)) + (2 * (number * number)))
                jo   handle_overflow
                add  ax, 11             ; ax = ax + 11, ax - numerator
                jo   handle_overflow

                mov  dx, number           ; Load x into dx
                add  dx, dx             ; dx = dx + dx (dx = 2 * number),
dx - first term of denumerator
                add  dx, 1

                mov  cx, dx             ; Move denumenator to the cx
                xor  dx, dx             ; reset dx to perform multiply
                div  cx                 ; ax = ax / cx (result in ax, remainder
in dx)

                mov  remainder, dx
                mov  result, ax


   return_result:
                mov  result, ax
                ret

process_function endp

print_number proc
   ; mov  bx, result          ; number
                or   bx, bx
                jns  m1
                mov  al, '-'
                int  29h
                neg  bx

   m1:
                mov  ax, bx
                xor  cx, cx
                mov  bx, 10
```

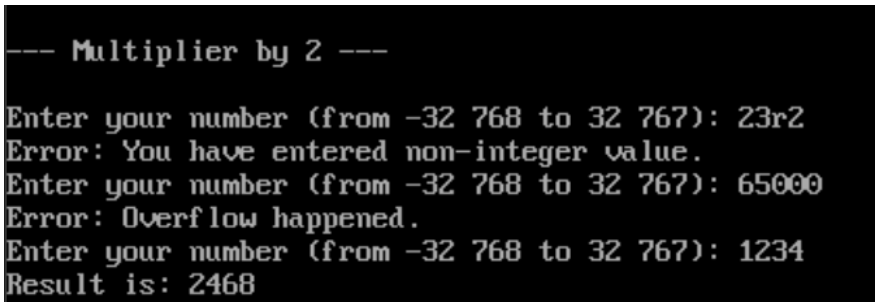
```
    m2:
                xor  dx, dx
                div  bx
                add  dl, '0'
                push dx
                inc  cx
                test ax, ax
                jnz  m2

    m3:
                pop  ax
                int  29h
                loop m3

                ret
print_number endp

end start
```

Введені та отримані результати



Вміст .lst файлу:

```
Turbo Assembler	 Version 4.1	 05/31/24 10:46:52	 Page 1
test.asm



      1                         .8086
      2 0000                    .model small
      3
      4 0000                    .stack 100h
      5
      6 0000                    .data
      7 0000  0A 2D 2D 2D 20 50    69+    header_msg          db       10,  '-
-- Piecewise Function Calculator ---', 10, 10, 13, '$'
      8       65 63 65 77 69 73 65+
      9       20 46 75 6E 63 74 69+
     10             6F 6E 20 43 61 6C       63+
     11             75 6C 61 74 6F 72 20+
     12             2D 2D 2D 0A 0A 0D       24
     13         002A  7C 20 78 2D 31 2C     20+    function_msg        db
        '| x-1,                      if x <       -1', 10, 13
     14             20 20 20 20 20 20 20+
     15             20 20 20 20 20 20 20+
     16             20 20 20 69 66 20 78+
     17             20 3C 20 2D 31 0A       0D
     18         004D  7C 20 30 2C 20 20     20+                 db       '| 0,
             if x =       -1', 10, 13
     19             20 20 20 20 20 20 20+
     20             20 20 20 20 20 20 20+
     21             20 20 20 69 66 20 78+
     22             20 3D 20 2D 31 0A       0D
     23         0070  7C 20 28 78 5E 33     2B+                 db       '|
(x^3+2x^2+11)/(2x+1), if x >       -1', 10, 10, 13, '$'
     24             32 78 5E 32 2B 31 31+
     25             29 2F 28 32 78 2B 31+
     26             29 2C 20 69 66 20 78+
     27             20 3E 20 2D 31 0A       0A+
     28             0D 24
```

```
29       0095  45 6E 74 65 72 20      79+    prompt_msg          db
  'Enter your X (from -32       768 to 32 767):   ', '$'
30             6F 75 72 20 58 20 28+
31             66 72 6F 6D 20 2D      33+
32             32 20 37 36 38 20 74+
33             6F 20 33 32 20 37 36+
34             37 29 3A 20 24
35       00BD  52 65 73 75 6C 74      20+    result_msg          db
  'Result       is: ', '$'
36             69 73 3A 20 24
37       00C9  52 65 6D 61 69 6E      64+    remainder_msg       db
  'Remainder is: ', '$'
38             65 72 20 69 73 3A 20+
39             24
40       00D8  45 72 72 6F 72 3A      20+    overflow_err_msg          db
  'Error:       Overflow happened.', 10, 13, '$'
41             4F 76 65 72 66 6C 6F+
42             77 20 68 61 70 70 65+
43             6E 65 64 2E 0A 0D      24
44       00F4  45 72 72 6F 72 3A      20+    non_int_err_msg           db
  'Error:       You have entered non-integer value.', 10, 13, '$'
45             59 6F 75 20 68 61 76+
46             65 20 65 6E 74 65 72+
47             65 64 20 6E 6F 6E 2D+
48             69 6E 74 65 67 65 72+
49             20 76 61 6C 75 65 2E+
50             0A 0D 24
51       0121  45 72 72 6F 72 3A      20+    incorrect_input_msg       db
  'Error:       Incorrect input    number.', 10, 13, '$'
52             49 6E 63 6F 72 72 65+
53             63 74 20 69 6E 70 75+
54             74 20 6E 75 6D 62      65+
55             72 2E 0A 0D 24
56       0142  0A 0D 24                new_line       db       10,      13,
  "$"
57
```

```
   58       0145  07 ?? 07*(??)          buffer              db       7,   ?,
7     dup (?)
   59       014E  0000                    number              dw       0
   60       0150  0000              result              dw       0
   61       0152  0000              remainder     dw       0
   62       0154  00                minus_sign          db       0
   63
   64       0155                  .code
   65       0000                     start:
   66       0000  B8 0000s                                   mov  ax, DGROUP
   67       0003  8E D8                                     mov  ds, ax
   68
   69       0005  BA 0000r                                   lea                 dx,
header_msg
   70       0008  E8 0037                                   call print_line
   71       000B  BA 002Ar                                   lea                 dx,
function_msg
   72       000E  E8 0031                                   call print_line
   73
   74       0011                  process_input:
   75       0011  E8 0033                                   call
process_number
   76       0014  E8 00BE                                   call
process_function
   77
   78       0017  BA 0142r                                   lea  dx, new_line
   79       001A  E8 0025                                   call print_line
   80       001D  BA 00BDr                                        lea         dx,
result_msg
   81       0020  E8 001F                                   call print_line
   82
   83       0023  8B 1E 0150r                               mov  bx, result
   84       0027  E8 0100                                   call print_number
   85
   86       002A  BA 0142r                                   lea  dx, new_line
```

```
   87       002D  E8 0012                              call print_line
   88       0030  BA 00C9r                             lea                dx,
remainder_msg
   89       0033  E8 000C                              call print_line
   90
   91       0036  8B 1E 0152r                          mov  bx, remainder
   92       003A  E8 00ED                              call print_number
   93                              ; terminate program
   94       003D  B8 4C00                              mov  ax, 4c00h
   95       0040  CD 21                                int  21h
   96
   97       0042                  print_line     proc
   98       0042  B4 09                                mov  ah, 09h
   99       0044  CD 21                                int  21h
  100       0046  C3                                      ret
  101       0047                  print_line     endp
  102
  103       0047                  process_number proc
  104                                 ; process_input:
  105       0047  33 C0                                xor  ax, ax
  106       0049  33 C9                                xor  cx, cx
  107       004B  33 D2                                xor  dx, dx
  108       004D  C6 06 0154r 00                       mov  minus_sign, 0
  109
  110                                 ; print prompt
  111       0052  BA 0095r                             lea                dx,
prompt_msg
  112       0055  E8 FFEA                              call print_line
  113
  114                                 ; receive user    input     using     DOS
      function 21h, interruption 0ah
```

```
115      0058  BA 0145r                    lea  dx, buffer
116      005B  B4 0A                       mov  ah, 0ah
117      005D  CD 21                       int  21h
118
119                          ; Process numerical input (digits 0-9)
120      005F  33 C0                       xor  ax, ax
121      0061  8A 0E 0146r                 mov  cl, buffer + 1
; Set the loop counter to thenumber of   +
122                          characters     to process
123      0065  BE 0002                     mov  si, 2
     ; Set       the SI index to    the first character +
124                          ofnumber
125      0068  80 3E 0147r 2D              cmp  buffer + 2, '-'
126      006D  75 08                       jne  process_digit
127
128                          ; Handle negative sign       (if present)
129      006F  46                               inc  si              ;
Increment SI to point to the first digit+
130                          after the minus sign
131      0070  FE C9                       dec  cl
; Decrement CL to account for     the minus   +
132                          sign (one less digit to process)
133      0072  C6 06 0154r 01              mov  minus_sign, 1
134
135                          ;  Convert  ASCII  to  integer  and
   check for errors
136      0077                process_digit:
137
138                          ; Handle non-integer values
139      0077  8A 9C 0145r                 mov  bl, buffer[si]
140      007B  80 FB 30                    cmp  bl, '0'
141      007E  72 28                       jb   handle_non_int
142      0080  80 FB 39                    cmp  bl, '9'
143      0083  77 23                       ja   handle_non_int
144
```

```
145                             ; Convert ASCII to integer
146     0085  80 EB 30                    sub  bl, '0'
147
148                             ; Add together  converted  digits  and
check for overflow
149     0088  BA 000A                     mov  dx, 10
150     008B  F7 EA                       imul dx
151     008D  70 0A                       jo
handle_overflow
152     008F  03 C3                       add  ax, bx
153     0091  70 24                       jo
handle_special_case
154
155     0093  46                        inc  si
156     0094  E2 E1                       loop process_digit
157     0096  EB 30 90                    jmp  check_sign
158
159     0099                            handle_overflow:
160     0099  BA 0142r                    lea  dx, new_line
161     009C  E8 FFA3                     call print_line
162     009F  BA 00D8r                    lea              dx,
overflow_err_msg
163     00A2  E8 FF9D                     call print_line
164     00A5  E9 FF69                     jmp  process_input
165
166     00A8                            handle_non_int:
167     00A8  BA 0142r                    lea  dx, new_line
168     00AB  E8 FF94                     call print_line
169     00AE  BA 00F4r                    lea              dx,
non_int_err_msg
170     00B1  E8 FF8E                     call print_line
171     00B4  E9 FF5A                     jmp  process_input
```

```
172
173  00B7                    handle_special_case:
174  00B7  3D 8000                        cmp  ax, -32768
175  00BA  75 DD                          jne
handle_overflow
176  00BC  80 3E 0154r 01                 cmp  minus_sign, 1
177  00C1  75 D6                          jne
handle_overflow
178  00C3  F7 D8                          neg  ax
179  00C5  EB 0A 90                       jmp
return_number
180
181  00C8                    check_sign:
182  00C8  80 3E 0154r 01                 cmp  minus_sign, 1
183  00CD  75 02                          jne  return_number
184  00CF  F7 D8                          neg  ax
185
186  00D1                    return_number:
187  00D1  A3 014Er                       mov  number, ax
188
189  00D4  C3                                  ret
190  00D5                process_number endp
191
192  00D5                process_function proc
193
194  00D5  33 C0                          xor  ax, ax
195  00D7  33 DB                          xor  bx, bx
196  00D9  33 D2                          xor  dx, dx
197
198  00DB  A1 014Er                       mov  ax, number
199  00DE  85 C0                          test ax, ax
200  00E0  79 11                          jns  third_operation
201
202  00E2  3D FFFF                        cmp  ax, -1
```

```
  203      00E5  74 06                          je
second_operation
  204
  205                          ; if x< -1
  206                          ; result = x -     1
  207      00E7                first_operation:
  208      00E7  2D 0001                        sub  ax, 1
  209                                  ; jo  handle_special_case
  210
  211      00EA  EB 3A 90                       jmp  return_result
  212
  213                          ; if x= -1
  214                          ; result = 0
  215      00ED                second_operation:
  216      00ED  B8 0000                        mov  ax, 0
  217
  218      00F0  EB 34 90                       jmp  return_result
  219
  220                          ; if x> -1
  221                          ; result = (x^3 + 2x^2 + 11) /      (2x +
1)
  222      00F3                third_operation:
  223      00F3  A1 014Er                       mov  ax, number
       ; Load number  into ax
  224      00F6  8B D0                          mov  dx, ax
; Copy ax to dx
  225      00F8  F7 E2                          mul  dx
; ax = ax * dx (ax = number *      number)
  226      00FA  70 9D                          jo
handle_overflow
  227
  228      00FC  8B C8                          mov  cx, ax
; Move result      to cx
```

```
   229      00FE  A1 014Er                          mov  ax, number
        ; Load number  into ax      again
   230      0101  F7 E1                            mul  cx
; ax = ax * cx (ax = number *      (number    *   +
   231                           number)), ax - first term of numerator
   232      0103  70 94                            jo
handle_overflow
   233
   234      0105  03 C9                            add  cx, cx
; cx = cx + cx (cx = 2 * (number *           +
   235                           number)), cx - second term     of
numenator
   236      0107  03 C1                            add  ax, cx
; ax = ax + cx (ax = (number * (number *  +
   237                           number)) +    (2 * (number * number)))
   238      0109  70 8E                            jo
handle_overflow
   239      010B  05 000B                          add  ax, 11
; ax = ax + 11, ax - numerator
   240      010E  70 89                            jo
handle_overflow
   241
   242      0110  8B 16 014Er                      mov  dx, number
        ; Load x into    dx
   243      0114  03 D2                            add  dx, dx
; dx = dx + dx (dx = 2 * number), dx -    +
   244                           first term      of denumerator
   245      0116  83 C2 01                         add  dx, 1
   246
   247      0119  8B CA                            mov  cx, dx
; Move denumenator to       the cx
   248      011B  33 D2                            xor  dx, dx
; reset dx to       perform     multiply
   249      011D  F7 F1                            div  cx
; ax = ax / cx (result in ax,   remainder in+
```

```
                        dx)

011F  89 16 0152r                   mov  remainder, dx
0123  A3 0150r                      mov  result, ax


0126                    return_result:
0126  A3 0150r                      mov  result, ax
0129  C3                        ret

012A                process_function endp

012A                print_number proc
                        ; mov       bx, result          ; number
012A  0B DB                         or   bx, bx
012C  79 06                         jns  m1
012E  B0 2D                         mov  al, '-'
0130  CD 29                         int  29h
0132  F7 DB                         neg  bx

0134                    m1:
0134  8B C3                         mov  ax, bx
0136  33 C9                         xor  cx, cx
0138  BB 000A                       mov  bx, 10

013B                    m2:
013B  33 D2                         xor  dx, dx
013D  F7 F3                         div  bx
013F  80 C2 30                      add  dl, '0'
0142  52                        push dx
0143  41                        inc  cx
0144  85 C0                         test ax, ax
0146  75 F3                         jnz  m2

0148                    m3:
0148  58                        pop  ax
```

```
286     0149  CD 29                       int  29h
287     014B  E2 FB                       loop m3
288
289     014D  C3                        ret
290     014E                print_number endp
291
292                         end start
```

Symbol Table

Symbol Name	Type	Value
??DATE	Text	"05/31/24"
??FILENAME	Text	"test "
??TIME	Text	"10:46:52"
??VERSION	Number	040A
@32BIT	Text	0
@CODE	Text	_TEXT
@CODESIZE	Text	0
@CPU	Text	0101H
@CURSEG	Text	_TEXT
@DATA	Text	DGROUP
@DATASIZE	Text	0
@FILENAME	Text	TEST
@INTERFACE	Text	000H
@MODEL	Text	2
@STACK	Text	DGROUP
@WORDSIZE	Text	2
BUFFER	Byte	DGROUP:0145
CHECK_SIGN	Near	_TEXT:00C8
FIRST_OPERATION	Near	_TEXT:00E7
FUNCTION_MSG	Byte	DGROUP:002A
HANDLE_NON_INT	Near	_TEXT:00A8
HANDLE_OVERFLOW	Near	_TEXT:0099
HANDLE_SPECIAL_CASE	Near	_TEXT:00B7
HEADER_MSG	Byte	DGROUP:0000
INCORRECT_INPUT_MSG	Byte	DGROUP:0121
M1	Near	_TEXT:0134
M2	Near	_TEXT:013B
M3	Near	_TEXT:0148
MINUS_SIGN	Byte	DGROUP:0154
NEW_LINE	Byte	DGROUP:0142
NON_INT_ERR_MSG	Byte	DGROUP:00F4
NUMBER	Word	DGROUP:014E

```
OVERFLOW_ERR_MSG . . . . . Byte  DGROUP:00D8
PRINT_LINE . . . . . Near  _TEXT:0042
PRINT_NUMBER . . . . . Near  _TEXT:012A
PROCESS_DIGIT . . . . . Near  _TEXT:0077
PROCESS_FUNCTION . . . . . Near  _TEXT:00D5
PROCESS_INPUT . . . . . Near  _TEXT:0011
PROCESS_NUMBER . . . . . Near  _TEXT:0047
PROMPT_MSG . . . . . Byte  DGROUP:0095
REMAINDER . . . . . Word  DGROUP:0152
REMAINDER_MSG . . . . . Byte  DGROUP:00C9
RESULT . . . . . Word  DGROUP:0150
RESULT_MSG . . . . . Byte  DGROUP:00BD
RETURN_NUMBER . . . . . Near  _TEXT:00D1
RETURN_RESULT . . . . . Near  _TEXT:0126
SECOND_OPERATION . . . . . Near  _TEXT:00ED
START . . . . . Near  _TEXT:0000
THIRD_OPERATION . . . . . Near  _TEXT:00F3
```

Symbol Table

Groups & Segments	Bit	Size	Align	Combine	Class
DGROUP		Group			
STACK	16	0100	Para	Stack	STACK
_DATA	16	0155	Word	Public	DATA
_TEXT	16	014E	Word	Public	CODE

Вміст .map файлу:

```
 Start  Stop   Length Name              Class

 00000H 0014DH 0014EH _TEXT            CODE
 00150H 002A4H 00155H _DATA            DATA
 002B0H 003AFH 00100H STACK             STACK

Program entry point at 0000:0000
```

Висновок:

Під час виконання комп’ютерного практикуму я розробив програму, що обраховує функцію за введеними користувачем параметрами та виводить результат на екран. Програма має можливість виведення десяткових чисел у вигляді остачі від ділення. Особливу увагу було приділено захисту від некоректного введення вхідних даних. Програма оброблює нецифрові символи, переповнення та некоректно введене число. Ця робота демонструє ефективне використання процедур для обміну даними і обробки помилок, що є важливими аспектами розробки програмного забезпечення. Завдяки цьому, програма може бути легко модифікована або розширена для виконання інших завдань або обробки інших типів даних.