

Федеральное государственное автономное образовательное  
учреждение высшего образования  
«Национальный исследовательский университет  
«Высшая школа экономики»

*Факультет компьютерных наук*

Ахаладзе Мария Мерабиевна, БПИ193(2)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К МИКРОПРОЕКТУ ПО ДИСЦИПЛИНЕ «АРХИТЕКТУРА  
ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»**

студента образовательной программы «Программная инженерия»  
по направлению подготовки 09.03.04 Программная инженерия

**Руководитель:**

профессор департамента  
программной инженерии  
Легалов А. И.

Москва, 2020 год

## Постановка задания

Разработать программу численного интегрирования функции  $y = a \cdot x + b \cdot x^3$  (задаётся целыми числами  $a, b$ ) в определённом диапазоне целых (задаётся так же) методом трапеций (шаг 1).

## Описание расчетных методов

Согласно [1], интеграл функции  $f(x)$  методом трапеций вычисляется по формуле (1):

$$\int_a^b f(x)dx \approx h \cdot \left( \frac{f_1 + f_N}{2} + f_2 + \dots + f_{N-1} \right), \quad (1)$$

где  $h$  – размер шага,

$$f_1 = f(a),$$

$$f_N = f(b).$$

По условию задания,  $h = 1$ . В таком случае задача сводится к тому, чтобы вычислить сумму значений функции в точках, принадлежащих промежутку  $(a; b)$  с шагом 1, и сложить ее со средним арифметическим  $f(a)$  и  $f(b)$ .

Частным случаем является равенство границ интегрирования, тогда без вычислений можно считать, согласно свойству 6 из [2], интеграл равным нулю.

Другим частным случаем является равенство  $a$  и  $b$  нулю, тогда функция, как и ее интеграл, тоже всегда равны нулю.

В общем случае задача решается следующим образом:

1. Определяется количество итераций цикла как  $b - a - 1$ , т.к. в цикле вычисляется сумма значений функции в точках, находящихся в диапазоне от  $a + 1$  до  $b - 1$  включительно. Если оно равно нулю, переход в пункт 4.
2. Стартовое значение аргумента равно  $b$ .
3. В цикле на каждой итерации повторяются следующие операции:
  - 3.1. Значение аргумента  $k$  уменьшается на 1.

- 3.2. К переменной `sum`, отведенной под сумму значений функции в точках интервала  $(a; b)$ , прибавляется  $f(k)$ .
4. В переменную `limitSum`, отведенную под сумму значений функции в крайних точках, прибавляются значения  $f(a)$  и  $f(b)$ .
5. В переменную `result`, отведенную под вычисление результирующего значения, добавляется `limitSum`, после чего ее значение делится на 2.
6. К переменной `result` прибавляется `sum`.

### Описание переменных

Входные данные хранятся в следующих переменных:

- значение параметра  $a$  – в `a`;
- значение параметра  $b$  – в `b`;
- значение нижнего предела интегрирования – в `lowerLimit`;
- значение верхнего предела интегрирования – в `upperLimit`.

Областью допустимых значений для параметров  $a$  и  $b$  являются целые числа, принадлежащие отрезку  $[-1000; 1000]$ .

Областью допустимых значений для пределов интегрирования `lowerLimit` и `upperLimit` являются целые числа, принадлежащие отрезку  $[-50; 50]$ , причем  $\text{upperLimit} \geq \text{lowerLimit}$ .

Границы допустимых значений хранятся в следующих переменных:

- минимально допустимое значение для  $a$  и  $b$  – в `minParameterValue` (равно -1000);
- максимально допустимое значение для  $a$  и  $b$  – в `maxParameterValue` (равно 1000);
- минимально допустимое значение для `lowerLimit` и `upperLimit` – в `minLimitValue` (равно -50);
- минимально допустимое значение для `lowerLimit` и `upperLimit` – в `maxLimitValue` (равно 50).

Промежуточные данные хранятся в следующих переменных:

- счетчик цикла – в `loopCounter`;

- значение аргумента для возведения в степень (в процедуре вычисления значения функции) – в temp;
- сумма значений функции в крайних точках – в limitSum;
- сумма значений функции в точках интервала (a; b) – в sum;
- делитель limitSum, равный 2 – в fivisor.

Выходными данными является вещественное число, хранящееся в переменной result – сумме площадей всех трапеций.

### **Реализация программы**

В программе используются следующие процедуры без параметров: ParameterAInput, ParameterBInput, LowerLimitInput, UpperLimitInput, CheckZero, IntegralCalculation, CheckParameterInput, CheckLimitInput, FunctionValue.

Процедуры ParameterAInput, ParameterBInput, LowerLimitInput, UpperLimitInput используются для ввода значений параметров a, b нижнего и верхнего пределов соответственно. После ввода значение записывается в регистр eax и вызывается соответствующая процедура проверки корректности значения. В процедуре UpperLimitInput перед этим еще происходит сравнение верхнего и нижнего пределов интегрирования, если первый меньше второго, то ввод считается некорректным и программа завершает выполнение.

Процедуры CheckParameterInput и CheckLimitInput проверяют, находится ли значение параметра и предела интегрирования соответственно в допустимых диапазонах. Регистр eax поочередно сравнивается с крайними допустимыми значениями параметра и предела интегрирования соответственно, в случае некорректности программа завершает выполнение.

Процедура CheckZero проверяет, равны ли нулю оба параметра функции (в таком случае интеграл тоже равен нулю, частный случай). Значения параметров a и b записываются поочередно в регистр eax, который сравнивается с нулем.

Процедура `FunctionValue` вычисляет значение функции с заданным аргументом. Его значение хранится в регистре `ebx`, значение функции сохраняется в него же.

Процедура `IntegralCalculation` вычисляет значение интеграла функции с помощью цикла (если итераций больше нуля), в котором вызывается `FunctionValue` для всех точек диапазона  $(a;b)$  и `ebx` прибавляется к переменной `sum`, и отдельного вызова `FunctionValue` для точек `a` и `b`, после чего полученные значения суммируются в переменную `limitSum`. После этого с помощью средств FPU к переменной `result` прибавляется `limitSum`, значение делится на `divisor` (равный двум) и прибавляется `sum`.

Значение `result` выводится с одним знаком после запятой, т.к. `sum` – целое число; в случае четности `limitSum` результат деления на 2 и `result` не имеет дробной части, а в случае нечетности – имеет дробную часть 0,5, поэтому большей точности и не требуется.

Программа была реализована с использованием информации (об инструкциях `loop`, `cdq`, работе с FPU, `esp`) из [3]. Листинг программы представлен в приложении А.

### Тестирование программы

Результат тестирования случая, когда `a` и `b` принадлежат промежутку  $(-1000; 1000)$  (неграничный случай), границы пределов принадлежат промежутку  $(-50; 50)$  (неграничный случай), а результат является дробным числом, приведен на рисунке 1.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 20
Input parameter b: 301
Input a lower integral limit: -1
Input an upper integral limit: 10
Integral value: 760094.5
```

*Рисунок 1 – Результат тестирования*

Результат тестирования случая, когда  $a = 1000$  (граничный случай),  $b = 1000$  (граничный случай), нижний предел равен  $-50$  (граничный случай), верхний предел равен  $50$  (граничный случай), приведен на рисунке 2.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1000
Input parameter b: 1000
Input a lower integral limit: -50
Input an upper integral limit: 50
Integral value: 100000.0
```

**Рисунок 2 – Результат тестирования**

Результат тестирования случая, когда  $a = -1000$  (граничный случай),  $b = -1000$  (граничный случай), приведен на рисунке 3.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: -1000
Input parameter b: -1000
Input a lower integral limit: 0
Input an upper integral limit: 50
Integral value: -1563175000.0
```

**Рисунок 3 – Результат тестирования**

Результат тестирования случая, когда  $a = -1000$  (граничный случай),  $b = -1000$  (граничный случай), нижний предел равен  $-50$  (граничный случай), верхний предел равен  $50$  (граничный случай), приведен на рисунке 4.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: -1000
Input parameter b: -1000
Input a lower integral limit: -50
Input an upper integral limit: 50
Integral value: -100000.0
```

**Рисунок 4 – Результат тестирования**

Результат тестирования случая, когда нижний и верхний предел одновременно равны  $-50$  (граничный случай) приведен на рисунке 5.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1
Input parameter b: 2
Input a lower integral limit: -50
Input an upper integral limit: -50
Integral value: 0.0
```

**Рисунок 5 – Результат тестирования**

Результат тестирования случая, когда нижний и верхний предел одновременно равны  $50$  (граничный случай) приведен на рисунке 6.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1
Input parameter b: 2
Input a lower integral limit: 50
Input an upper integral limit: 50
Integral value: 0.0
```

**Рисунок 6 – Результат тестирования**

Результат тестирования случая, когда  $a = 0$  ( $f(x) = bx^3$ ), приведен на рисунке 7.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 0
Input parameter b: 600
Input a lower integral limit: -15
Input an upper integral limit: 10
Integral value: -6112500.0
```

*Рисунок 7 – Результат тестирования*

Результат тестирования случая, когда  $b = 0$  ( $f(x) = a$ ), приведен на рисунке 8.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 500
Input parameter b: 0
Input a lower integral limit: -10
Input an upper integral limit: 40
Integral value: 25000.0
```

*Рисунок 8 – Результат тестирования*

Результат тестирования случая, когда пределы интегрирования равны, приведен на рисунке 9.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 2
Input parameter b: 3
Input a lower integral limit: 45
Input an upper integral limit: 45
Integral value: 0.0
```

*Рисунок 9 – Результат тестирования*

Результат тестирования случая, когда  $a=b=1000$ , нижний предел равен 0, верхний предел равен 50 (максимально возможное значение функции при данных ограничениях входных данных), приведен на рисунке 10.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1000
Input parameter b: 1000
Input a lower integral limit: 0
Input an upper integral limit: 50
Integral value: 1563175000
```

*Рисунок 10 – Результат тестирования*

Результат тестирования случая, когда  $a = b = -1000$ , нижний предел равен 0, верхний предел равен 50 (минимально возможное значение функции при данных ограничениях входных данных), приведен на рисунке 11.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: -1000
Input parameter b: -1000
Input a lower integral limit: 0
Input an upper integral limit: 50
Integral value: -1563175000.0
```

*Рисунок 11 – Результат тестирования*

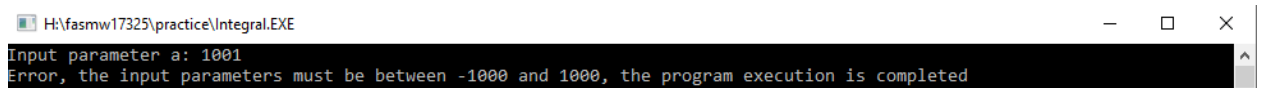
Результат тестирования случая, когда оба параметра равны нулю ( $f(x) = 0$ ), приведен на рисунке 12.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 0
Input parameter b: 0
Input a lower integral limit: 1
Input an upper integral limit: 2
Integral value: 0.0
```

*Рисунок 12 – Результат тестирования*

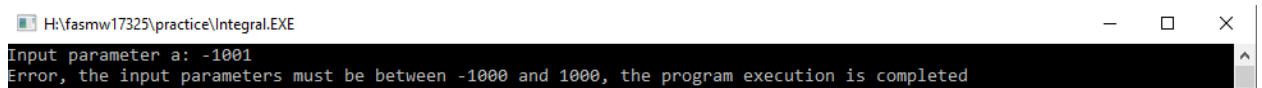
Результат тестирования случая, когда  $a > 1000$  (некорректный ввод), приведен на рисунке 13.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1001
Error, the input parameters must be between -1000 and 1000, the program execution is completed
```

*Рисунок 13 – Результат тестирования*

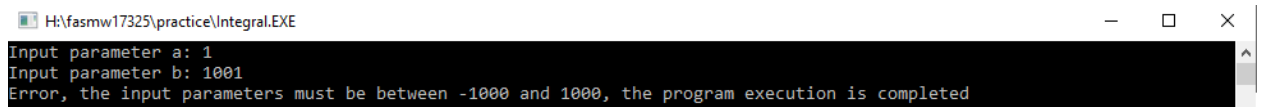
Результат тестирования случая, когда  $a < -1000$  (некорректный ввод), приведен на рисунке 14.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: -1001
Error, the input parameters must be between -1000 and 1000, the program execution is completed
```

*Рисунок 14 – Результат тестирования*

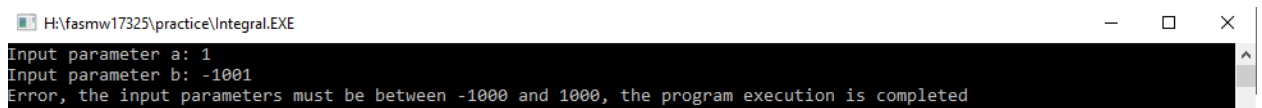
Результат тестирования случая, когда  $b > 1000$  (некорректный ввод), приведен на рисунке 15.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1
Input parameter b: 1001
Error, the input parameters must be between -1000 and 1000, the program execution is completed
```

*Рисунок 15 – Результат тестирования*

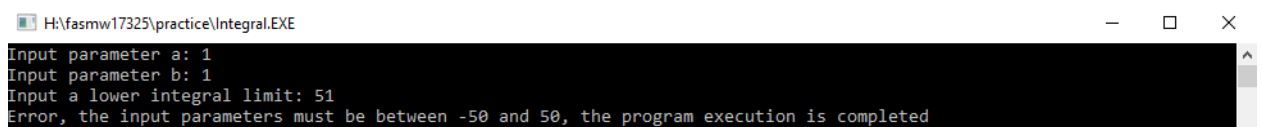
Результат тестирования случая, когда  $b < -1000$  (некорректный ввод), приведен на рисунке 16.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1
Input parameter b: -1001
Error, the input parameters must be between -1000 and 1000, the program execution is completed
```

*Рисунок 16 – Результат тестирования*

Результат тестирования случая, когда нижний предел больше 50 (некорректный ввод), приведен на рисунке 17.

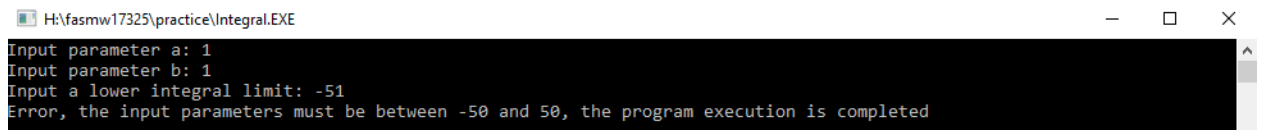


```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1
Input parameter b: 1
Input a lower integral limit: 51
Error, the input parameters must be between -50 and 50, the program execution is completed
```

*Рисунок 17 – Результат тестирования*



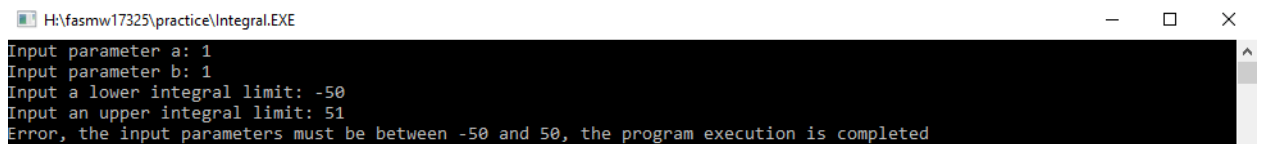
Результат тестирования случая, когда нижний предел меньше -50 (некорректный ввод), приведен на рисунке 18.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1
Input parameter b: 1
Input a lower integral limit: -51
Error, the input parameters must be between -50 and 50, the program execution is completed
```

*Рисунок 18 – Результат тестирования*

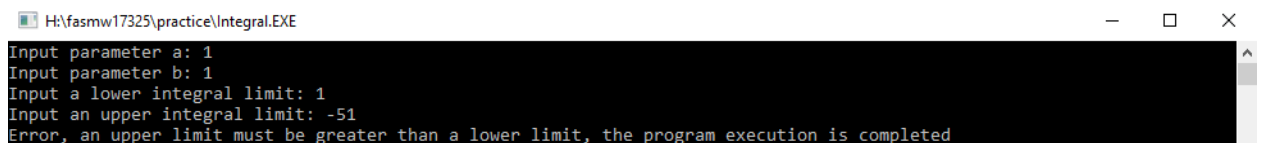
Результат тестирования случая, когда верхний предел больше 50 (некорректный ввод), приведен на рисунке 19.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1
Input parameter b: 1
Input a lower integral limit: -50
Input an upper integral limit: 51
Error, the input parameters must be between -50 and 50, the program execution is completed
```

*Рисунок 19 – Результат тестирования*

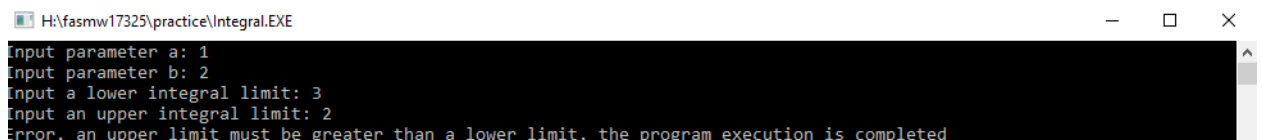
Результат тестирования случая, когда верхний предел меньше -50 (некорректный ввод), приведен на рисунке 20.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1
Input parameter b: 1
Input a lower integral limit: 1
Input an upper integral limit: -51
Error, an upper limit must be greater than a lower limit, the program execution is completed
```

*Рисунок 20 – Результат тестирования*

Результат тестирования случая, когда верхний предел больше нижнего, приведен на рисунке 21.



```
H:\fasmw17325\practice\Integral.EXE
Input parameter a: 1
Input parameter b: 2
Input a lower integral limit: 3
Input an upper integral limit: 2
Error, an upper limit must be greater than a lower limit, the program execution is completed
```

*Рисунок 21 – Результат тестирования*

## Библиографический список

- [1] Учебное пособие по курсу "Численные методы в оптике" // Университет ИТМО URL: [http://aco.ifmo.ru/el\\_books/numerical\\_methods/lectures/glava2\\_2.html](http://aco.ifmo.ru/el_books/numerical_methods/lectures/glava2_2.html) (дата обращения: 29.10.2020).
- [2] Свойства интегралов // Томский политехнический университет URL: [https://portal.tpu.ru/SHARED/k/KONVAL/Sites/Russian\\_sites/Calc1-ru/12/05.htm](https://portal.tpu.ru/SHARED/k/KONVAL/Sites/Russian_sites/Calc1-ru/12/05.htm) (дата обращения: 29.10.2020).
- [3] Programmer's Manual // flat assembler URL: <https://flatassembler.net/docs.php?article=manual> (дата обращения: 29.10.2020).

## Приложение А

### Листинг программы

; Ахаладзе Мария, БПИ193  
; Разработать программу численного интегрирования функции  $y=a+b*x^3$   
(задаётся целыми числами a,b)  
; в определённом диапазоне целых (задаётся так же) методом трапеций  
(шаг 1)

format PE console

entry start

include 'win32a.inc'

;-----

section '.data' data readable writable

stringA db 'Input parameter a: ', 0

stringB db 'Input parameter b: ', 0

stringLowerLimit db 'Input a lower integral limit: ', 0

stringUpperLimit db 'Input an upper integral limit: ', 0

stringScanningInteger db '%d', 0

stringOutput db 'Integral value:%1 %.1f', 0

stringInputError db 'Error, the input parameters must be between %d and  
%d, the program execution is completed', 10, 0

stringLimitError db 'Error, an upper limit must be greater than a lower  
limit, the program execution is completed', 10, 0

lowerLimit	dd 0	; Нижний предел интегрирования
upperLimit	dd 0	; Верхний предел интегрирования
loopCounter	dd 0	; Счетчик цикла
temp	dd 0	; Временная переменная для вычислений
значения функции $ax + bx^3$		
sum	dd 0	; Сумма значений функции $ax + bx^3$ в
точках интервала (lowerLimit; upperLimit)		
limitSum	dd 0	; Сумма значений функции $ax + bx^3$ в
lowerLimit и upperLimit		
divisor	dq 2.0	; Делитель для вычисления значения $(f1$
$+ fN) / 2$		
result	dq 0.0	; Результат вычисления
a	dd 0.0	; Параметр a функции
b	dd 0.0	; Параметр b функции
maxParameterValue	dd 1000	; Максимально допустимое
значение параметра		
minParameterValue	dd -1000	; Минимально допустимое
значение параметра		
maxLimitValue	dd 50	; Максимально допустимое значение
предела интегрирования		
minLimitValue	dd -50	; Минимально допустимое значение
предела интегрирования		

;------

-----

-----

section '.code' code readable executable

start:

call ParameterAInput ; Ввод параметра a

call ParameterBInput ; Ввод параметра b

call LowerLimitInput ; Ввод нижнего предела интегрирования

call UpperLimitInput ; Ввод верхнего предела интегрирования

call CheckZero ; Проверка случая, когда a и b равны нулю

call IntegralCalculation ; Вычисление значения интеграла

push dword[result+4] ; Вывод значения интеграла

push dword[result]

push stringOutput

call [printf]

finish: ; Метка окончания программы

call [getch]

push 0

call [ExitProcess]

```

;-----
-----

proc ParameterAInput      ; Ввод параметра a

    push stringA          ; Вносим в стек строку, уведомляющую о вводе
параметра a
    call [printf]         ; Выводим строку
    add esp, 4            ; Удаление аргумента со стека

    push a                ; Вносим в стек переменную-параметр a
    push stringScanningInteger ; Вносим в стек строку, в которую вводится
значение a
    call [scanf]          ; Считываем значение
    add esp, 8            ; Удаление аргументов со стека

    mov eax, [a]          ; Заносим значение a в регистр для проверки
корректности значения
    call CheckParameterInput ; Проверка корректности значения
параметра, записанного в eax

    ret

endp

;-----
-----

proc ParameterBInput      ; Ввод параметра b

```

push stringB ; Вносим в стек строку, уведомляющую о вводе параметра b

call [printf] ; Выводим строку

add esp, 4 ; Удаление аргумента со стека

push b ; Вносим в стек переменную-параметр b

push stringScanningInteger ; Вносим в стек строку, в которую вводится значение b

call [scanf] ; Считываем значение

add esp, 8 ; Удаление аргументов со стека

mov eax, [b] ; Заносим значение b в регистр для проверки корректности значения

call CheckParameterInput ; Проверка корректности значения параметра, записанного в eax

ret

endp

-----

-----

-----

proc LowerLimitInput ; Ввод нижнего предела интегрирования

push stringLowerLimit ; Вносим в стек строку, уведомляющую о вводе нижнего предела интегрирования

call [printf] ; Выводим строку

add esp, 4 ; Удаление аргумента со стека

push lowerLimit ; Вносим в стек переменную-нижний предел  
интегрирования

push stringScanningInteger ; Вносим в стек строку, в которую вводится  
значение

call [scanf] ; Считываем значение

add esp, 8 ; Удаление аргументов со стека

mov eax, [lowerLimit] ; Заносим значение lowerLimit в регистр для  
проверки корректности значения

call CheckLimitInput ; Проверка корректности значения предела  
интегрирования, записанного в eax

ret

endp

;------

-----

-----

proc UpperLimitInput ; Ввод верхнего предела интегрирования

push stringUpperLimit ; Вносим в стек строку, уведомляющую о  
вводе верхнего предела интегрирования

call [printf] ; Выводим строку

add esp, 4 ; Удаление аргумента со стека

push upperLimit ; Вносим в стек переменную-верхний предел  
интегрирования

push stringScanningInteger ; Вносим в стек строку, в которую вводится  
значение

call [scanf] ; Считываем значение

add esp, 8 ; Удаление аргументов со стека



```
mov eax, [upperLimit]    ; Заносим значение upperLimit в регистр
cmp eax, [lowerLimit]    ; Сравниваем eax со значением нижнего
предела интегрирования
```

```
jl Error                ; Верхний предел интегрирования должен быть не
меньше нижнего
```

```
call CheckLimitInput     ; Проверка корректности значения предела
интегрирования, записанного в eax
ret
```

Error:

```
push stringLimitError    ; Вносим в стек строку, уведомляющую об
ошибке
```

```
call [printf]           ; Выводим строку
```

```
jmp finish              ; Переходим к метке окончания программы
```

```
endp
```

```
;-----
```

```
-----
```

```
-----
```

```
proc CheckParameterInput    ; Проверка, что введенное значение
параметра находится в допустимом диапазоне
```

```
cmp eax, [maxParameterValue] ; Сравниваем eax с максимальным
допустимым значением параметра
```

```
jg ParameterValueError      ; Переход к метке об ошибке, если eax
больше
```

cmp eax, [minParameterValue] ; Сравниваем eax с минимальным  
допустимым значением параметра

jl ParameterValueError ; Переход к метке об ошибке, если eax  
меньше

ret

ParameterValueError:

push [maxParameterValue] ; Помещаем в стек максимально  
допустимое значение параметра

push [minParameterValue] ; Помещаем в стек минимально  
допустимое значение парметра

push stringInputError ; Помещаем в стек строку, уведомляющую  
об ошибке

call [printf] ; Выводим строку

jmp finish ; Переходим к метке окончания программы

endp

-----

-----

-----

proc CheckLimitInput ; Проверка, что введенное значения предела  
интегрирования находится в допустимом диапазоне

cmp eax, [maxLimitValue] ; Сравниваем eax с максимальным  
допустимым значением предела интегрирования

jg LimitValueError ; Переход к метке об ошибке, если eax больше

cmp eax, [minLimitValue] ; Сравниваем eax с минимальным  
допустимым значением предела интегрирования

jl LimitValueError ; Переход к метке об ошибке, если eax меньше

ret

LimitValueError:

push [maxLimitValue] ; Помещаем в стек максимально допустимое  
значение предела интегрирования

push [minLimitValue] ; Помещаем в стек минимально допустимое  
значение предела интегрирования

push stringInputError ; Помещаем в стек строку, уведомляющую об  
ошибке

call [printf] ; Выводим строку

jmp finish ; Переходим к метке окончания программы

endp

;------

-----

-----

proc CheckZero ; Проверка частного случая, когда  $f(x) = 0$

mov eax, [a] ; Заносим в стек значение параметра a

cmp eax, 0 ; Сравниваем с нулем

jne exitCheck ; Если равенство не выполняется, функция не  
имеет вид  $f(x) = 0$

mov eax, [b] ; Заносим в стек значение параметра b

cmp eax, 0 ; Сравниваем с нулем  
jne exitCheck ; Если равенство не выполняется, функция не  
имеет вид  $f(x) = 0$

push 0 ; Функция имеет вид  $f(x) = 0$ , значит, интеграл равен  
нулю

push stringOutput ; Вносим в стек строку, выводящую результат  
call [printf] ; Выводим значение интеграла

jmp finish ; Переходим к метке окончания программы

exitCheck:

ret

endp

;------

-----

-----

proc FunctionValue ; Вычисление значения функции  $ax + bx^3$ , где  
значение  $x$  записано в ebx

mov [temp], ebx ; Помещаем ebx в значение temp для того, чтобы  
корректно посчитать возведение в степень

imul ebx, [temp] ; Умножаем ebx на значение temp (получается  
значение выражения вида  $x * x$ )

imul ebx, [temp] ; Умножаем ebx на значение temp (получается  
значение выражения вида  $(x * x) * x$ )

imul ebx, [b] ; Умножаем ebx на значение параметра b (получается  
значение выражения вида  $b * x^3$ )

add ebx, [a] ; Складываем ebx со значением параметра a  
(получается значение выражения вида  $a + b * x^3$ )

ret

endp

;-----

-----  
-----

proc IntegralCalculation ; Вычисление значения интеграла методом  
трапеций

mov ecx, [upperLimit] ; Помещаем значение нижнего предела  
интегрирования в ecx

cmp ecx, [lowerLimit] ; Сравниваем ecx со значением верхнего  
предела интегрирования

je exitCalculation ; Если пределы интегрирования равны, то  
интеграл равен нулю и нет необходимости вычислять значение кодом ниже

sub ecx, [lowerLimit] ; Количество итераций вычислим как разность  
верхнего и нижнего пределов интегрирования,

sub ecx, 1 ; после чего вычтем 1 (т.к. в цикле считаются  
значения функции строго внутри границ)

cmp ecx, 0 ; Сравнение ecx с нулем

je calculateLimitValues ; Если ecx = 0, то в цикле 0 итераций, нужно  
сразу перейти к вычислению значений функции в граничных точках

mov ebx, [upperLimit] ; Помещаем значение верхнего предела  
интегрирования в ebx, чтобы приравнять через него upperLimit и loopCounter

mov [loopCounter], ebx

calculationLoop: ; Цикл, в котором вычисляется сумма значений функции на промежутке (lowerLimit; upperLimit)

xor ebx, ebx ; Очистка ebx

sub [loopCounter], 1 ; С каждой итерацией берем аргументом функции число, на 1 меньшее предыдущего (начиная с upperLimit - 1)

mov ebx, [loopCounter] ; Помещаем это значение в ebx

call FunctionValue ; Вызываем процедуру для вычисления значения функции, аргумент которой равен loopCounter, значение записывается в ebx

add [sum], ebx

loop calculationLoop

calculateLimitValues:

mov ebx, [lowerLimit] ; Помещаем в ebx значение нижнего предела интегрирования

call FunctionValue ; Вызываем процедуру для вычисления значения функции, аргумент которой равен lowerLimit, значение записывается в ebx

add [limitSum], ebx ; Прибавляем ebx к сумме значений функции в граничных точках limitSum

xor ebx, ebx

mov ebx, [upperLimit] ; Помещаем в ebx значение верхнего предела интегрирования

call FunctionValue ; Вызываем процедуру для вычисления значения функции, аргумент которой равен upperLimit, значение записывается в ebx

add [limitSum], ebx ; Прибавляем ebx к сумме значений функции в граничных точках limitSum

mov ebx, 2 ; Помещаем в ebx 2 для того, чтобы далее поделить на 2 значение limitSum

mov [temp], ebx ; Помещаем ebx в значение temp

FINIT ; Инициализация сопроцессора

FLD [result] ; Загружаем значение result типа qword (изначально равное нулю)

FIADD [limitSum] ; Производим целочисленное сложение с limitSum, т.к. limitSum – целое число

FDIV [divisor] ; Делим значение limitSum на 2

FIADD [sum] ; Производим целочисленное сложение с sum, т.к. sum – целое число

FSTP [result] ; Запись в result

exitCalculation: ; Выход из процедуры вычисления интеграла

ret

endp

;

-----

-----

```
section '.idata' import data readable
```

```
library kernel, 'kernel32.dll',\  
    msvcrt, 'msvcrt.dll',\  
    user32, 'USER32.DLL'
```

```
import kernel,\  
    ExitProcess, 'ExitProcess'
```

```
import msvcrt,\  
    printf, 'printf',\  
    scanf, 'scanf',\  
    getch, '_getch'
```