Шабалкин Виталий Витальевич
БПИ-214
Вариант №14

Задача: Разработать программу, вычисляющую с помощью степенного ряда с точностью не хуже 0,1% значение функции гиперболического котангенса cth (x) = (e^x + e^(−x)) / (e^x − e^(−x)) для заданного параметра x.

Код на С:

```
#include <stdio.h>
#include <math.h>

long int fact (int n)
{
if (n <= 1)
{
return 1;
}
else
{
return n * fact (n - 1);
}
}

double bin (long int n, long int k)
{
return 1.0 * fact (n) / fact (k) / fact (n - k);
}

double bernoulli (long int n)
{
if (n <= 0)
{
return 1.0;
}
else
{
double sum = 0;
for (long k = 1; k <= n; k++)
{
sum += bin (n + 1, k + 1) * bernoulli (n - k);
}
return -1.0 / (n + 1) * sum;
}
}

int main ()
{
double x;
scanf ("%lf", &x);
double cur = 1.0 / x;
double perf = (exp (x) + exp (-x)) / (exp (x) - exp (-x));
double eps = perf / 1000;
int counter = 1;
do
{
cur = cur + pow (2, 2 * counter) * bernoulli (2 * counter) * pow (x, 2 * counter - 1) / fact (2 *
counter);
counter += 1;

}
while (fabs (cur - perf) > eps);
printf ("%lf", cur);
return 0;
}
```

Код на GAS:

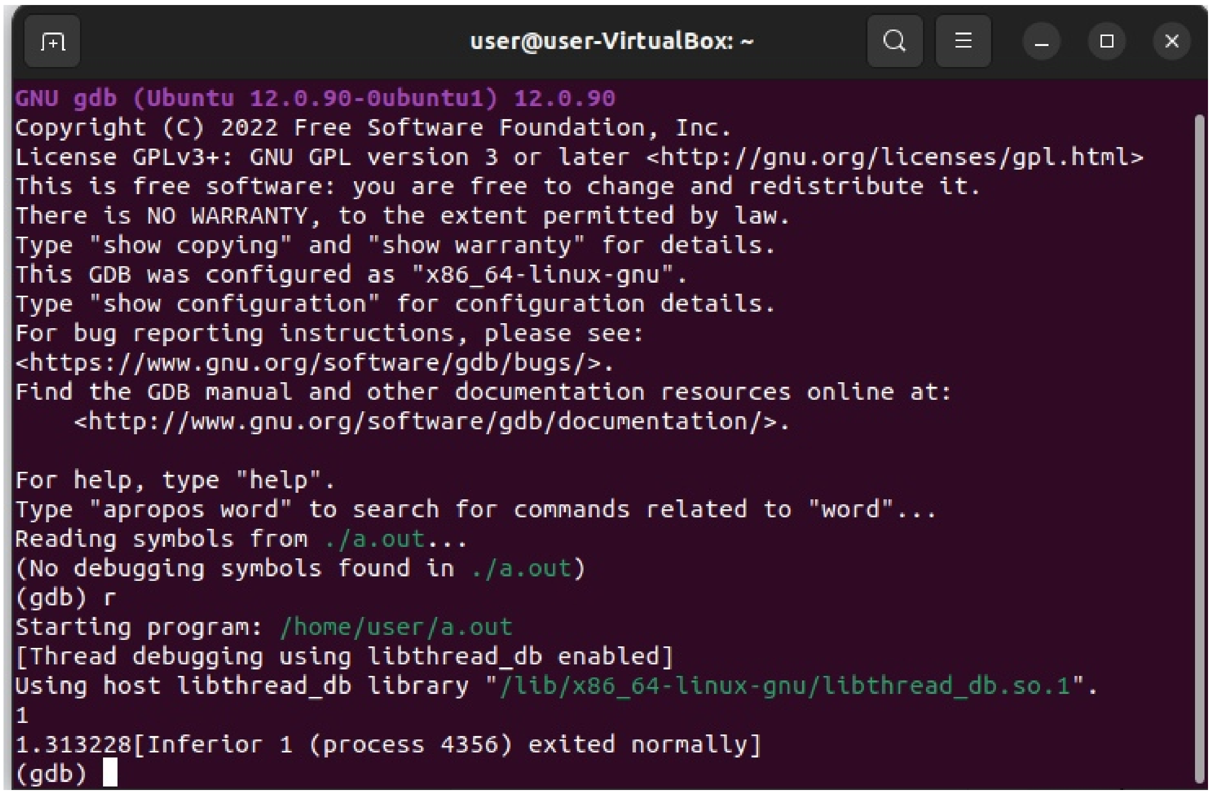
```
		.file	"code.c"
	.text
	.globl	fact
	.type	fact, @function
fact:
	endbr64
	pushq	%rbp
	movq	%rsp, %rbp
	pushq	%rbx
	subq	$24, %rsp
	movl	%edi, -20(%rbp)
	cmpl	$1, -20(%rbp)
	jg	.L2		#Если n > 1, то переход к блоку L2
	movl	$1, %eax
	jmp	.L3		#Переход к блоку L3
.L2:
	movl	-20(%rbp), %eax
	movslq	%eax, %rbx
	movl	-20(%rbp), %eax
	subl	$1, %eax
	movl	%eax, %edi
	call	fact	#Рекурентный вызов fact (n - 1)
	imulq	%rbx, %rax #n * fact(n-1)
.L3:
	movq	-8(%rbp), %rbx
	leave
	ret	#return
	.size	fact, .-fact
	.globl	bin
	.type	bin, @function
bin:
	endbr64
	pushq	%rbp
	movq	%rsp, %rbp
	subq	$32, %rsp
	movq	%rdi, -8(%rbp)		#Аргумент n
	movq	%rsi, -16(%rbp)			#Аргумент k
	movq	-8(%rbp), %rax
	movl	%eax, %edi
	call	fact			#fact(n)
	pxor	%xmm2, %xmm2
	cvtsi2sdq	%rax, %xmm2
	movsd	%xmm2, -24(%rbp)
	movq	-16(%rbp), %rax
	movl	%eax, %edi
	call	fact			#fact(k)
	pxor	%xmm0, %xmm0
	cvtsi2sdq	%rax, %xmm0
	movsd	-24(%rbp), %xmm2
	divsd	%xmm0, %xmm2			#fact(n)/fact(k)
	movsd	%xmm2, -24(%rbp)
	movq	-8(%rbp), %rax
	movl	%eax, %edx
	movq	-16(%rbp), %rax
	movl	%eax, %ecx
	movl	%edx, %eax
	subl	%ecx, %eax		#(n-k)
	movl	%eax, %edi
	call	fact			#fact(n-k)
	pxor	%xmm0, %xmm0
	cvtsi2sdq	%rax, %xmm0
	movsd	-24(%rbp), %xmm1
	divsd	%xmm0, %xmm1			#fact(n)/fact(k)/fact(n-k)
	movq	%xmm1, %rax
	movq	%rax, %xmm0
	leave
	ret	#return fact(n)/fact(k)/fact(n-k)
	.size	bin, .-bin
	.globl	bernoulli
	.type	bernoulli, @function
bernoulli:
	endbr64
	pushq	%rbp
	movq	%rsp, %rbp
	subq	$32, %rsp
	movq	%rdi, -24(%rbp)
	cmpq	$0, -24(%rbp)
```

```
        jg      .L7                     #Если n > 0, то переход к блоку L7
        movsd .LC0(%rip), %xmm0
        jmp     .L8                     #Переход к блоку L8
.L7:
        pxor    %xmm0, %xmm0
        movsd %xmm0, -16(%rbp)          #sum = 0
        movq    $1, -8(%rbp)            #k = 1 (for)
        jmp     .L9
.L10:
        movq    -8(%rbp), %rax
        leaq    1(%rax), %rdx
        movq    -24(%rbp), %rax
        addq    $1, %rax                #n + 1, k + 1
        movq    %rdx, %rsi
        movq    %rax, %rdi
        call    bin                     #bin (n + 1, k + 1)
        movsd %xmm0, -32(%rbp)
        movq    -24(%rbp), %rax
        subq    -8(%rbp), %rax                  #n - k
        movq    %rax, %rdi
        call    bernoulli               #bernoulli (n - k)
        mulsd   -32(%rbp), %xmm0        #bin (n + 1, k + 1) * bernoulli (n - k)
        movsd -16(%rbp), %xmm1
        addsd   %xmm1, %xmm0                    #sum += bin (n + 1, k + 1) * bernoulli (n - k)
        movsd %xmm0, -16(%rbp)
        addq    $1, -8(%rbp)            #k++
.L9:
        movq    -8(%rbp), %rax
        cmpq    -24(%rbp), %rax
        jle     .L10                    #Если k<=n, то выплняется блок L10
        movq    -24(%rbp), %rax
        addq    $1, %rax                #n+1
        pxor    %xmm1, %xmm1
        cvtsi2sdq       %rax, %xmm1
        movsd .LC2(%rip), %xmm0
        divsd   %xmm1, %xmm0                    #-1/(n+1)
        mulsd   -16(%rbp), %xmm0        #-1/(n+1)*sum
.L8:
        movq    %xmm0, %rax
        movq    %rax, %xmm0
        leave
        ret     #return в зависимости от того какой блок выполнился в выводке будет
лежать разное значение
        .size   bernoulli, .-bernoulli
        .section        .rodata
.LC3:
        .string "%lf"
        .text
        .globl  main
        .type   main, @function
main:
        endbr64
        pushq   %rbp
        movq    %rsp, %rbp
        pushq   %rbx
        subq    $72, %rsp
        movq    %fs:40, %rax
        movq    %rax, -24(%rbp)
        xorl    %eax, %eax
        leaq    -56(%rbp), %rax
        movq    %rax, %rsi
        leaq    .LC3(%rip), %rax
        movq    %rax, %rdi
        movl    $0, %eax
        call    __isoc99_scanf@PLT#scanf ("%lf", &x)
        movsd -56(%rbp), %xmm1
        movsd .LC0(%rip), %xmm0
        divsd   %xmm1, %xmm0
        movsd %xmm0, -48(%rbp)
        movq    -56(%rbp), %rax
        movq    %rax, %xmm0
        call    exp@PLT
        movsd %xmm0, -72(%rbp)     #cur = 1.0 / x
        movsd -56(%rbp), %xmm0
        movq    .LC4(%rip), %xmm1
        movapd          %xmm0, %xmm3
        xorpd   %xmm1, %xmm3
        movq    %xmm3, %rax
        movq    %rax, %xmm0
        call    exp@PLT
        movapd          %xmm0, %xmm2
        addsd   -72(%rbp), %xmm2     #exp (x) + exp (-x)
        movsd %xmm2, -72(%rbp)
        movq    -56(%rbp), %rax
        movq    %rax, %xmm0
        call    exp@PLT
        movq    %xmm0, %rbx
        movsd -56(%rbp), %xmm0
        movq    .LC4(%rip), %xmm1
        movapd          %xmm0, %xmm4
        xorpd   %xmm1, %xmm4
        movq    %xmm4, %rax
        movq    %rax, %xmm0
        call    exp@PLT
        movq    %rbx, %xmm1
        subsd   %xmm0, %xmm1                #exp (x) + exp (-x)
```

```
	movsd -72(%rbp), %xmm0
	divsd	%xmm1, %xmm0		#(exp (x) + exp (-x)) / (exp (x) - exp (-x))
	movsd %xmm0, -40(%rbp)
	movsd -40(%rbp), %xmm0	#perf = (exp (x) + exp (-x)) / (exp (x) - exp (-x))
	movsd .LC5(%rip), %xmm1
	divsd	%xmm1, %xmm0
	movsd %xmm0, -32(%rbp)	#eps = perf / 1000
	movl	$1, -60(%rbp)		#counter = 1
.L12:
	movl	-60(%rbp), %eax
	addl	%eax, %eax
	pxor	%xmm0, %xmm0
	cvtsi2sdl	%eax, %xmm0
	movq	.LC6(%rip), %rax
	movapd	%xmm0, %xmm1
	movq	%rax, %xmm0
	call	pow@PLT		#pow (2, 2 * counter)
	movsd %xmm0, -72(%rbp)
	movl	-60(%rbp), %eax
	addl	%eax, %eax
	cltq
	movq	%rax, %rdi
	call	bernoulli	#bernoulli (2 * counter)
	movapd	%xmm0, %xmm5
	mulsd	-72(%rbp), %xmm5	#pow (2, 2 * counter) * bernoulli (2 * counter)
	movsd %xmm5, -72(%rbp)
	movl	-60(%rbp), %eax
	addl	%eax, %eax
	subl	$1, %eax
	pxor	%xmm0, %xmm0
	cvtsi2sdl	%eax, %xmm0
	movq	-56(%rbp), %rax
	movapd	%xmm0, %xmm1
	movq	%rax, %xmm0
	call	pow@PLT		#pow (x, 2 * counter - 1)
	mulsd	-72(%rbp), %xmm0	#pow (2, 2 * counter) * bernoulli (2 * counter) * pow (x,
2 * counter - 1)
	movsd %xmm0, -72(%rbp)
	movl	-60(%rbp), %eax
	addl	%eax, %eax
	movl	%eax, %edi
	call	fact			#fact (2 * counter)
	pxor	%xmm1, %xmm1
	cvtsi2sdq	%rax, %xmm1
	movsd -72(%rbp), %xmm0
	divsd	%xmm1, %xmm0		#pow (2, 2 * counter) * bernoulli (2 * counter) *
pow (x, 2 * counter - 1) / fact (2 * counter)
	movsd -48(%rbp), %xmm1
```

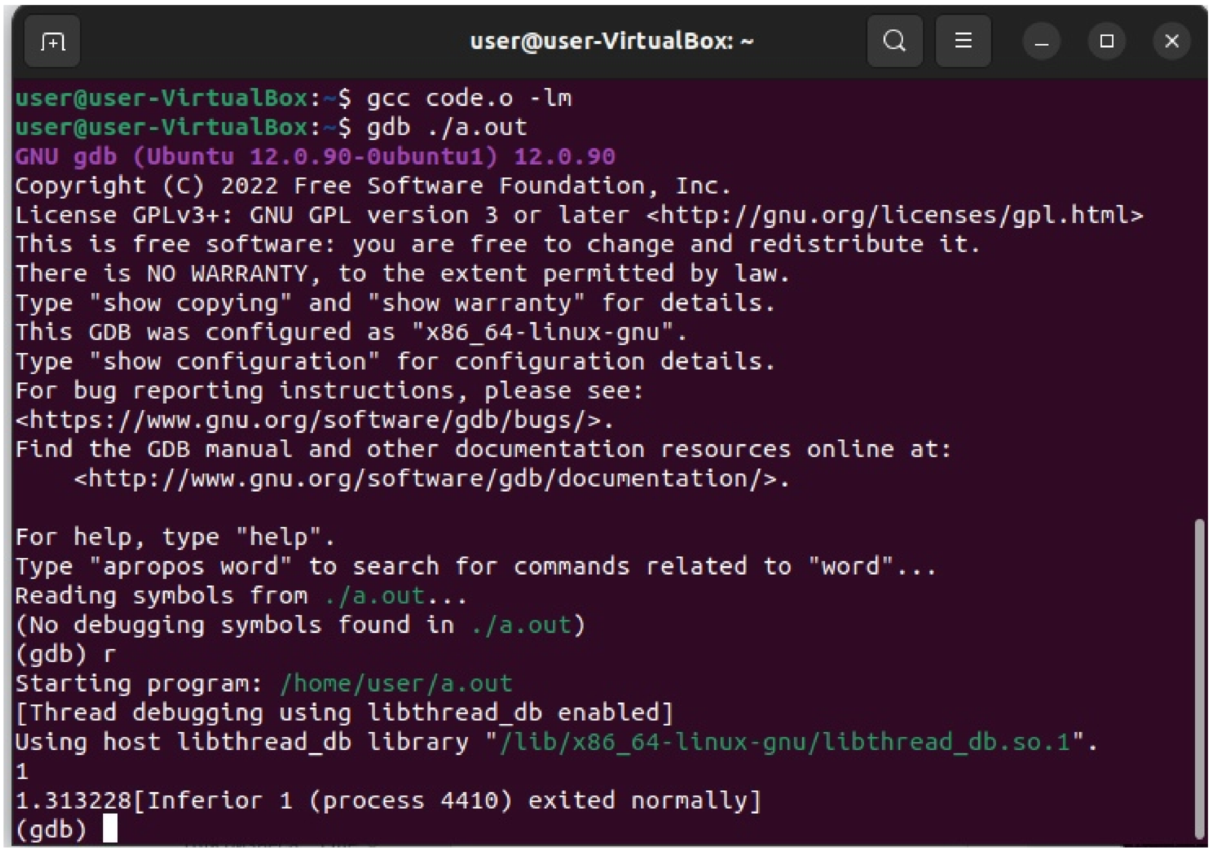
```
        addsd  %xmm1, %xmm0                 #cur + pow (2, 2 * counter) * bernoulli (2 *
counter) * pow (x, 2 * counter - 1) / fact (2 * counter)
        movsd %xmm0, -48(%rbp)      #cur = cur + pow (2, 2 * counter) * bernoulli (2 *
counter) * pow (x, 2 * counter - 1) / fact (2 * counter)
        addl   $1, -60(%rbp)        #counter += 1
        movsd -48(%rbp), %xmm0
        subsd  -40(%rbp), %xmm0
        movq   .LC7(%rip), %xmm1
        andpd  %xmm1, %xmm0
        comisd -32(%rbp), %xmm0
        ja     .L12                 #если fabs (cur - perf) > eps, то выполняется блок
L12 (do-while)
        movq   -48(%rbp), %rax
        movq   %rax, %xmm0
        leaq   .LC3(%rip), %rax
        movq   %rax, %rdi
        movl   $1, %eax
        call   printf@PLT           #printf ("%lf", cur)
        movl   $0, %eax
        movq   -24(%rbp), %rdx
        subq   %fs:40, %rdx
        je     .L14                 #переход к L14
        call   __stack_chk_fail@PLT
.L14:
        movq   -8(%rbp), %rbx
        leave
        ret
        .size  main, .-main
        .section        .rodata
        .align 8
.LC0:
        .long  0
        .long  1072693248
        .align 8
.LC2:
        .long  0
        .long  -1074790400
        .align 16
.LC4:
        .long  0
        .long  -2147483648
        .long  0
        .long  0
        .align 8
.LC5:
        .long  0
        .long  1083129856
        .align 8
```

```
.LC6:
	.long	0
	.long	1073741824
	.align 16
.LC7:
	.long	-1
	.long	2147483647
	.long	0
	.long	0
	.ident	"GCC: (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0"
	.section	.note.GNU-stack,"",@progbits
	.section	.note.gnu.property,"a"
	.align 8
	.long	1f - 0f
	.long	4f - 1f
	.long	5
0:
	.string	"GNU"
1:
	.align 8
	.long	0xc0000002
	.long	3f - 2f
2:
	.long	0x3
3:
	.align 8
4:
```



```
GNU gdb (Ubuntu 12.0.90-0ubuntu1) 12.0.90
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(No debugging symbols found in ./a.out)
(gdb) r
Starting program: /home/user/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
1
1.313228[Inferior 1 (process 4356) exited normally]
(gdb)
```



```
user@user-VirtualBox:~$ gcc code.o -lm
user@user-VirtualBox:~$ gdb ./a.out
GNU gdb (Ubuntu 12.0.90-0ubuntu1) 12.0.90
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(No debugging symbols found in ./a.out)
(gdb) r
Starting program: /home/user/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
1
1.313228[Inferior 1 (process 4410) exited normally]
(gdb)
```

Были использованы функции и локальные переменные