**ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»**

**(СПбГУТ)**

**Кафедра безопасности информационных систем**

**ОТЧЁТ**

по практической работе работе №2 на тему:   
**«Анализ алгоритмов и программ»**

по дисциплине «Алгоритмы и структуры данных»

Выполнил: студент группы ИСТ-114,Константинов А.Д.

«21» сентября 2022 г. \_\_\_\_\_\_\_\_\_\_\_/А.Д. Константинов/

Принял: к.ф.-м.н., доцент, И.А. Моисеев

«05» октября 2022 г. \_\_\_\_\_\_\_\_\_\_\_/ И.А. Моисеев /

**Содержание:**

[Цель работы: 3](#_Toc114692552)

[Листинг кода: 3](#_Toc114692553)

[Результат работы программы: 6](#_Toc114692554)

[Вывод: 7](#_Toc114692555)

[Приложение: 8](#_Toc114692556)

Цель работы:

Найти локальный минимум (максимум) функции вида f(x) = на заданном интервале [a,b] с заданной точностью ε > 0 одним из способов:

1) Методом «деления отрезка» пополам;

2) Методом «золотого сечения»;

3) Методом «Фибоначчи».

Провести анализ разработанного алгоритма и программы и сравнить с аналогичным решением с помощью алгоритма «пассивного поиска».

# Листинг кода:

#include <iostream>  
#include <cmath>  
float func(float x) {  
 return pow(x,3)-x+exp(-x);  
}  
void PassiveSearch(float left,float right, float e)  
{  
 int steps = 0;  
 float minPoint = 0;  
 float minVal = 99999999;  
 for (float i = left; i <= right; i+=e) {  
 steps++;  
 float tmp = func(i);  
 if(minVal > tmp) {  
 minVal = tmp;  
 minPoint = i;  
 }  
 }  
 std::cout << "X:" << minPoint << " Y:" << func(minPoint) << std::endl;  
}  
void Task1\_SearchHalfPart(float a,float b,float e){  
 float x1;  
 float a1=0,b1 = 999999;  
 float a0 = a;  
 float b0 = b;  
  
 while(b1-a1 > e) {  
 float om = (b0 - a0) / 10;  
 float alp0 = (a0 + b0) / 2 - om;  
 float bet0 = alp0 + 2 \* om;  
 float fAlp = func(alp0);  
 float fBet = func(bet0);  
  
  
 if (fAlp <= fBet) {  
 x1 = alp0;  
 a1 = a0;  
 b1 = bet0;  
 } else {  
 a1 = alp0;  
 x1 = bet0;  
 b1 = b0;  
 }  
  
 a0 = a1;  
 b0 = b1;  
 }  
 float res = (a1 + b1) / 2;  
 std::cout << "X:" << res << " Y:" << func(res) << std::endl;  
}  
void Task1\_SearchGoldCut(float a,float b,float e){  
 float x1;  
 float a1=0,b1 = 999999;  
 float a0 = a;  
 float b0 = b;  
  
 while(b1-a1 > e) {  
 float om = b0 - a0;  
 float alp0 =a0+(2\*om)/(3+sqrt(5));  
 float bet0 = a0+(2\*om)/(1+sqrt(5));  
 float fAlp = func(alp0);  
 float fBet = func(bet0);  
  
  
 if (fAlp <= fBet) {  
 x1 = alp0;  
 a1 = a0;  
 b1 = bet0;  
 } else {  
 a1 = alp0;  
 x1 = bet0;  
 b1 = b0;  
 }  
  
 a0 = a1;  
 b0 = b1;  
 }  
 float res = (a1 + b1) / 2;  
 std::cout << "X:" << res << " Y:" << func(res) << std::endl;  
}  
float Fib(int n)  
{  
 if(n == 0) return 1;  
 if(n== 1) return 1;  
 return Fib(n-1) + Fib(n-2);  
}  
void Task1\_SearchFib(float a, float b, float e)  
{  
 int f0 = 1;  
 int f1=1;  
 float a0 = a;  
 float b0 = b;  
 float d0 = b0-a0;  
 int N = 1;  
  
 while(d0/Fib(N) >= e)  
 {  
 N++;  
 }  
  
  
 float x1=0;  
 float a1=0,b1 = 999999;  
 int k =0;  
 while(b1-a1 > e) {  
 float om = b0 - a0;  
 float alp0 = a0 + Fib(N-k-2)/Fib(N-k)\*om;  
 float bet0 = a0 + Fib(N-k-1)/Fib(N-k)\*om;  
 float fAlp = func(alp0);  
 float fBet = func(bet0);  
  
 if (fAlp <= fBet) {  
 x1 = alp0;  
 a1 = a0;  
 b1 = bet0;  
 } else {  
 a1 = alp0;  
 x1 = bet0;  
 b1 = b0;  
 }  
  
 a0 = a1;  
 b0 = b1;  
 k++;  
 }  
 float res = (a1 + b1) / 2;  
 std::cout << "X:" << res << " Y:" << func(res) << std::endl;  
}  
int main() {  
 float left,right,e;  
// std::cout << "Current function y=x^3-x+e^(-x)" << std::endl;  
// std::cout << "Type left limit:";  
// std::cin >> left;  
// std::cout << "Type right limit:";  
// std::cin >> right;  
// std::cout << "Type accuracy:";  
// std::cin >> e;  
// std::cout<<std::endl;  
//  
// PassiveSearch(left,right,e);  
 e= 0.001;  
 std::cout<< "Task 2 Functions(accuracy = 0.001)" << std::endl;  
 std::cout << std::endl<<"Part 1: Searching extremum in range[-5,-3]" << std::endl;  
 left = -5;  
 right = -3;  
 std::cout << "Method 1: Classic search: "; PassiveSearch(left,right,e);  
 std::cout << "Method 2: Like-binary search: ";Task1\_SearchHalfPart(left,right,e);  
 std::cout << "Method 3: Gold-cut search: ";Task1\_SearchGoldCut(left,right,e);  
 std::cout << "Method 4: Fib search: ";Task1\_SearchFib(left,right,e);  
  
 std::cout << std::endl<<"Part 2: Search Max-points in range [-3,0]"; std::cout << std::endl;  
 left = -3;  
 right = 0;  
 std::cout << "Method 1: Classic search: "; PassiveSearch(left,right,e);  
 std::cout << "Method 2: Like-binary search: ";Task1\_SearchHalfPart(left,right,e);  
 std::cout << "Method 3: Gold-cut search: ";Task1\_SearchGoldCut(left,right,e);  
 std::cout << "Method 4: Fib search: ";Task1\_SearchFib(left,right,e);  
  
 std::cout <<std::endl << "Part 3: Search Min-points in range [0,3]"; std::cout << std::endl;  
 left = 0;  
 right = 3;  
 std::cout << "Method 1: Classic search: "; PassiveSearch(left,right,e);  
 std::cout << "Method 2: Like-binary search: ";Task1\_SearchHalfPart(left,right,e);  
 std::cout << "Method 3: Gold-cut search: ";Task1\_SearchGoldCut(left,right,e);  
 std::cout << "Method 4: Fib search: ";Task1\_SearchFib(left,right,e);  
  
 std::cout <<std::endl<< "End. Thanks for using our software" << std::endl;  
 return 0;  
}

# Результат работы программы:

"/Users/andrew/Projects/Algoritms/Algorithms and Data Structures/Second task/second/cmake-build-debug/second"

Task 2 Functions(accuracy = 0.001)

Part 1: Searching extremum in range[-5,-3]

Method 1: Classic search: X:-3.6791 Y:-6.50965

Method 2: Like-binary search: X:-3.67856 Y:-6.50965

Method 3: Gold-cut search: X:-3.67868 Y:-6.50965

Method 4: Fib search: X:-3.67918 Y:-6.50965

Part 2: Search Max-points in range [-3,0]

Method 1: Classic search: X:-3 Y:-3.91446

Method 2: Like-binary search: X:-2.99958 Y:-3.91196

Method 3: Gold-cut search: X:-2.99958 Y:-3.91198

Method 4: Fib search: X:-2.99964 Y:-3.91234

Part 3: Search Min-points in range [0,3]

Method 1: Classic search: X:0.705994 Y:0.139511

Method 2: Like-binary search: X:0.705522 Y:0.13951

Method 3: Gold-cut search: X:0.705585 Y:0.13951

Method 4: Fib search: X:0.704975 Y:0.139511

End. Thanks for using our software

Process finished with exit code 0

# Вывод:

В ходе выполнения практической работы была написана программа, которая находит локальные минимумы(максимумы) функции на трех отрезках – [-5,-3];[-3,0];[0,3]. Каждый промежуток решается при помощи четырех различных методов – пассивный поиск(последовательный перебор чисел), поиск делением отрезка пополам, поиск золотым сечение, поиск методом Фибоначчи. Самым тяжелым для исполнения является пассивный поиск – он выполняет 3000 операций для поиска подходящего числа, самый быстрый алгоритм - поиск методом Фибоначчи, он требует всего 18 итераций алгоритма, что является чрезвычайно хорошим результатом. По точности все алгоритмы выдают заданные значения – 0.001 – максимальное различие ответов. Вся программа поделена на 6 функций и работает на вывод в текстовом режиме. В результате выводит все расчеты и заканчивает свою работу.

# Приложение:

Тестовые файлы(a.txt;b.txt). и файл исходного кода main.cpp можно найти по адресу <https://github.com/StSiRe/Algoritms/tree/master/Algorithms%20and%20Data%20Structures/Second%20task/second>