Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра И5 «Информационные системы и программная инженерия»

**Практическая работа №5**по дисциплине «Информатика: Основы программирования»  
на тему «Функции»

Выполнил:  
Студент Альков В.С.  
Группа И407Б  
  
Преподаватель:  
Першин Д.В.

Санкт-Петербург  
2020 г.

1. Отсортировать массив М (50) и каждую строку матрицы  
   А (6х7) в порядке убывания

*Исходные данные:*

*L - const*, тип *int,* размер *M*.

*N, С* – *const*, тип *int*, размеры *b*, *N* - кол-во строк, *C*– столбцов.

*M[L]* – статический массив, тип *int*, размером L,

*b –* указатель на *int,* матрица,тип *int\*\*,* выделение памяти способом для динамической матрицы

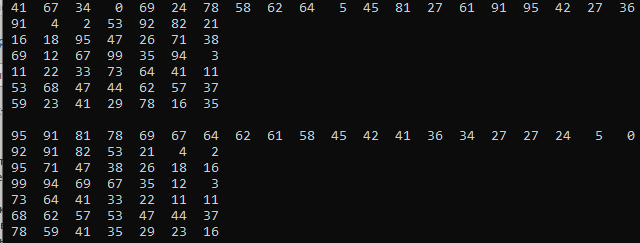
*Результирующие данные:*

печать массива M и матрицы b, тип *int*

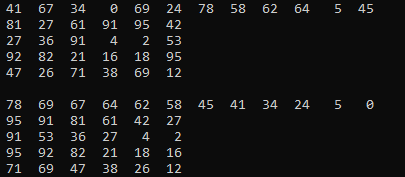
Таблица тестирования:

**Случайные числа**

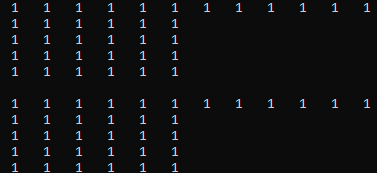
L=20 N=6 C=7



L=12 N=4 C=6



**Одинаковые числа**



Основной алгоритм:

начало

окончание

scan(M, L)

scan2(b, N, C)

sort(M,L)

sort2(b, N, C)

print(M, L)

print2(b, N, C)

Вспомогательные алгоритмы:

1. Алгоритм ввода одномерного массива

*Входные данные:* имя массива, размер.

*Результирующие данные:* заполненный массив

Схема алгоритма:

scan(a, n)

ввод ai

окончание

Цикл 1

i=1,n ш. 1

Цикл 1

Этот алгоритм описывается в программе функцией

*void scan(int\*, int)*

*Параметры:*

первый параметр *–* адрес первого элемента массива

второй параметр – количество элементов массива

Возвращаемое значение –отсутствует

*Вспомогательные переменные:*

*i –* тип *int*, индекс переданного массива

1. Алгоритм ввода двумерной динамической матрицы

*Входные данные:* имя матрицы, кол-во строк, кол-во столбцов.

*Результирующие данные:* заполненная матрица

Схема алгоритма:

scan2(a, n, m)

окончание

Цикл 1

i=1,n ш. 1

Цикл 1

scan(ai, m)

Этот алгоритм описывается в программе функцией

*void scan2(int\*\*, int, int)*

*Параметры:*

первый параметр *–* адрес первой строки матрицы

второй параметр – кол-во строк матриц

третий параметр – кол-во столбцов матрицы

Возвращаемое значение –отсутствует

*Вспомогательные переменные:*

*i –* тип *int*, индекс переданного массива

1. Алгоритм вывода одномерного массива

*Входные данные:* имя массива, размер.

*Результирующие данные:* нет

Схема алгоритма:

print(a, n)

вывод ai

окончание

Цикл 1

i=1,n ш. 1

Цикл 1

n<1

нет

да

Этот алгоритм описывается в программе функцией

*void print(int\*, int)*

*Параметры:*

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

*Возвращаемое значение* отсутствует.

*Вспомогательные переменные:*

*i –* тип *int*, индекс переданного массива

1. Алгоритм вывода двумерной динамической матрицы

*Входные данные:* кол-во строк, кол-во столбцов.

*Результирующие данные:* нет

Схема алгоритма:

print2(a, n, m)

окончание

Цикл 1

i=1,n ш. 1

Цикл 1

print(ai, m)

Этот алгоритм описывается в программе функцией

*void print2(int\*\*, int, int)*

*Параметры:*

первый параметр – адрес первой строки матрицы

второй параметр – количество строк матрицы

третий параметр – количество столбцов матрицы

*Возвращаемое значение* отсутствует.

*Вспомогательные переменные:*

*i –* тип *int*, индекс переданного массива

1. Алгоритм выделения памяти для динамической матрицы, матрица хранится по строкам

Этот алгоритм описывается в программе функцией

*int\*\* create2(int\*\*, int, int)*

*Параметры:*

первый параметр – указатель, под который надо выделить память

второй параметр – количество строк

третий параметр – количество столбцов

*Возвращаемое значение –* адрес первого элемента в массиве указателей или NULL в случае отсутствия искомого значения.

*Вспомогательные переменные:*

*i –* тип *int*, индекс переданного массива, индекс строки

*j –* тип *int*, переменная для очистки памяти, выделенной под строки, в том случае если память не выделилась под строку.

1. Алгоритм сортировки одномерного массива по убыванию

*Входные данные:* имя массива, размер.

*Результирующие данные:* массив будет отсортирован по убыванию

Схема алгоритма:

sort(a, n)

tmp = aj

aj = aj+1

aj+1 = tmp

окончание

Цикл 1

i=n-1,1 ш. 1

Цикл 1

Цикл 2

j=1,i ш. 1

Цикл 2

aj < aj+1

нет

да

Этот алгоритм описывается в программе функцией

*void sort (int\*, int)*

Параметры:

первый параметр – адрес первого элемента массива

второй параметр – количество элементов массива

*Возвращаемое значение –* отсутствует

*Вспомогательные переменные:*

*i,j –* тип *int*, индексы переданного массива

1. Алгоритм сортировки строк матрицы по убыванию

*Входные данные:* имя массива, размер.

*Результирующие данные:* числа в строках будут отсортированы по убыванию*.*

Схема алгоритма:

sort2(a, n, m)

окончание

Цикл 1

i=1,n ш. 1

Цикл 1

sort(ai, m)

Этот алгоритм описывается в программе функцией

*void sort2 (int\*\*, int, int)*

Параметры:

первый параметр – адрес первой строки матрицы

второй параметр – кол-во строк матрицы

третий параметр – кол-во столбцов матрицы

*Возвращаемое значение –* отсутствует

*Вспомогательные переменные:*

*i,–* тип *int*, индекс массива a, отвечающий за строки

1. Алгоритм освобождения памяти, выделенной под динамическую матрицу

*Входные данные:* имя матрицы, кол-во строк.

*Результирующие данные:* выделенная под матрицу память будет очищена

Этот алгоритм описывается в программе функцией

*void free2(int\*\*, int)*

*Параметры:*

первый параметр – адрес первой строки матрицы.

второй параметр – количество строк

*Возвращаемое значение –* отсутствует

*Вспомогательные переменные:*

*i –* тип *int*, индекс переданного массива, индекс строки

Текст программы

#include <stdio.h>

#include <stdlib.h>

#define L 50

#define N 6

#define C 7

/\*объявление функций\*/

void scan(int\*, int);

void print(int\*, int);

void print2(int\*\*, int, int);

void scan2(int\*\*, int, int);

int\*\* create2(int\*\*, int, int);

void sort(int\*, int);

void sort2(int\*\*, int, int);

void free2(int\*\*, int);

int main()

{

int M[L], \*\*b;

/\*выделение памяти под b\*/

b=create2(b,N,C);

/\*ввод M\*/

scan(M,L);

/\*ввод b\*/

scan2(b,N,C);

/\*сортировка M\*/

sort(M,L);

/\*сортировка b\*/

sort2(b,N,C);

/\*печать M\*/

print(M,L);

/\*печать b\*/

print2(b,N,C);

/\*освобождение памяти, выделенной под b\*/

free2(b,N);

return 0;

}

/\*фунция ввода одномерного массива\*/

void scan(int\* a, int n)

{

int i;

for(i=0; i<n; i++)

scanf("%d", &a[i]);

//a[i] = rand()%100;

}

/\*фунция печати одномерного массива\*/

void print(int\*a, int n)

{

int i;

if (n<1)

return;

for(i=0; i<n; i++)

printf("%3d ", a[i]);

printf("\n");

}

/\*фунция ввода двумерной динамической матрицы\*/

void scan2(int\*\* a, int n, int m)

{

int i;

for(i=0; i<n; i++)

/\*используем ф-ию scan, чтобы ввести каждую строку\*/

scan(a[i], m);

}

/\*фунция вывода двумерной динамической матрицы\*/

void print2(int\*\*a, int n, int m)

{

int i;

for(i=0; i<n; i++)

/\*используем ф-ию print, чтобы ввести каждую строку\*/

print(a[i], m);

}

/\*фунция выделения памяти под двумерную матрицу\*/

int\*\* create2(int\*\*a, int n, int m)

{

int i,j;

/\*выделяем память под указатели на строки\*/

a = malloc(n\*sizeof(int\*));

/\*если память не выделилась, возвращаем NULL\*/

if (a==NULL)

return NULL;

for(i=0; i<n; i++)

{

/\*выделяем память под каждую строку,

если не выделилась, очищаем память

под уже выделенные строки и возвращаем NULL\*/

a[i] = malloc(m\*sizeof(int));

if (a[i]==NULL)

{

for(j=0;j<i;j++)

free(a[j]);

free(a);

return NULL;

};

};

return a;

}

/\*функция сортировки одномерного

массива пузырьком по убыванию\*/

void sort(int\*a, int n)

{

int i,j, tmp;

for(i=n-1; i>0; i--)

for(j=0; j<i; j++)

if(a[j]<a[j+1])

{

tmp = a[j];

a[j] = a[j+1];

a[j+1] = tmp;

};

}

/\*функция сортировки строк динамической

матрицы по убыванию\*/

void sort2(int\*\*a, int n, int m)

{

int i;

for(i=0; i<n; i++)

/\*сортируем каждую строку\*/

sort(a[i], m);

}

/\*функия особождения памяти, выделенной под матрицу\*/

void free2(int\*\*a, int n)

{

int i;

for(i=0;i<n;i++)

/\*очищаем каждую строку\*/

free(a[i]);

/\*очищаем указатели на строки\*/

free(a);

}

1. Удалить из матрицы А (7х5) все строки, сумма элементов которых четная, а из матрицы В (6х8) – строки, сумма элементов которых нечетная, передвинув на их место следующие строки без нарушения порядка их следования.

*Исходные данные:*

*n1, m1, n2, m2* – размеры матриц *a* и *b*, *n1* и *n2* – кол-во строк, *m1* и *m2* – кол-во столбцов, тип *int*.

*a, b* – указатели на матрицы, тип *int\*\**, память выделена методом динамической матрицы

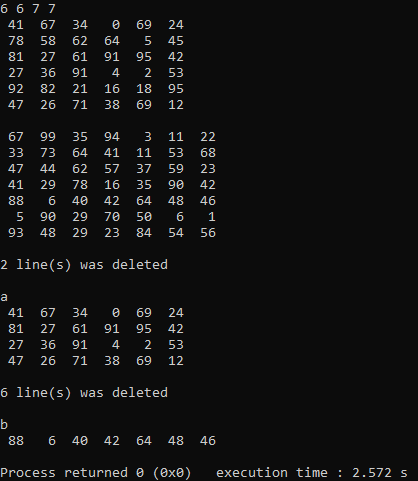
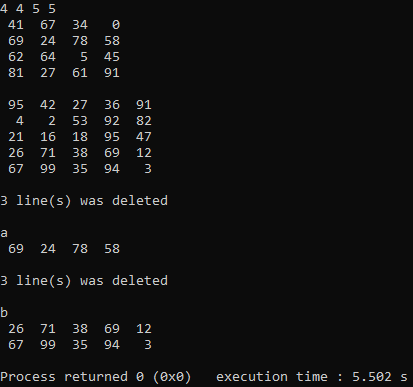
*Результирующие данные:*

матрицы *a* и *b*, тип *int*

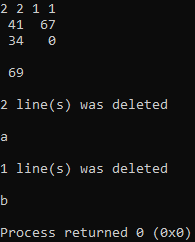
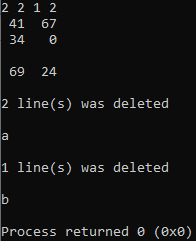
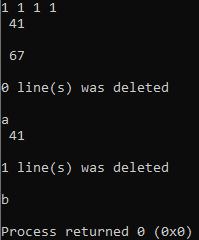
*Вспомогательные переменные:*

*z –* кол-во удаленных строк, тип *int*

Таблица тестирования:



Если все строки матрицы должны быть удалены



Основной алгоритм:

начало

окончание

scan2(a, n1, m1)

scan2(b, n2, m2)

z = delete\_lines(a, n1, m1, 0)

print2(a, n1, m1)

print2(b, n2, m2)

n1=scanN()

m1=scanN()

n2=scanN()

m2=scanN()

z = delete\_lines(b, n2, m2, 1)

n1 = n1 - z

n2 = n2 - z

Вспомогательные алгоритмы:

1. Алгоритм ввода одномерного массива и описывающая его функция те же, что и в предыдущей задаче
2. Алгоритм ввода двумерной матрицы и описывающая его функция те же, что и в предыдущей задаче
3. Алгоритм вывода одномерного массива и описывающая его функция те же, что и в предыдущей задаче
4. Алгоритм вывода двумерной матрицы и описывающая его функция те же, что и в предыдущей задаче
5. Алгоритм выделения памяти для двумерной матрицы и описывающая его функция те же, что и в предыдущей задаче
6. Алгоритм освобождения памяти, выделенной под двумерную матрицу и описывающая его функция те же, что и в предыдущей задаче
7. Алгоритм ввода натурального числа

*Входные данные:* нет.

*Результирующие данные:* нет

Схема алгоритма:

scanN

n

Цикл 1

Цикл 1

пока n<1

ввод n

Этот алгоритм описывается в программе функцией

*void scanN ()*

Параметры: нет

*Возвращаемое значение –* натуральное число, введенное пользователем

*Вспомогательные переменные:*

*n –* тип *int*, вводимая переменная

1. Алгоритм удаления строки матрицы

*Входные данные:* имя матрицы, индекс строки, кол-во строк матрицы.

*Результирующие данные*: удаление строки со смещение последующих

строк на ее место.

Схема алгоритма:

delete\_line(a, n, m)

окончание

Цикл 1

i=n, m-1 ш.1

Цикл 1

ai = ai+1

Этот алгоритм описывается в программе функцией

*void delete\_line(int\*\*, int, int)*

*Параметры:*

первый параметр – адрес матрицы

второй параметр – индекс строки, которую надо удалить

третий параметр – кол-во строк матрицы

*Возвращаемое значение –* отсутствует

*Вспомогательные переменные:*

*i* – инедкс строки, тип *int*

1. Алгоритм суммы строки

*Входные данные:* имя матрицы, индекс строки, кол-во строк матрицы.

*Результирующие данные*: удаление строки со смещение последующих строк вперед.

Схема алгоритма:

sum\_line(a, n)

s

Цикл 1

i=1, n ш.1

Цикл 1

s = s+ai

s = 0

Этот алгоритм описывается в программе функцией

*int sum\_line(int\*, int)*

*Параметры:*

первый параметр – адрес первого эл-та строки

второй параметр – кол-во столбцов матрицы

*Возвращаемое значение –* сумма эл-ов строки

*Вспомогательные переменные:*

*i* – индекс столбца, тип *int*

1. Алгоритм удаления строк, сумма эл-ов которых четная или нечетная

*Входные данные:* имя матрицы, индекс строки, кол-во строк матрицы.

*Результирующие данные*: удаление строки со смещением последующих строк на ее место.

Схема алгоритма:

delete\_lines(a,n,m,p)

c

Цикл 1

i=1, n ш.1

Цикл 1

c=c+1

i=i-1

n=n-1

c = 0

s делится на p

s=sum\_line(ai, m)

delete\_line(a, i, m)

нет

да

Этот алгоритм описывается в программе функцией

*int delete\_lines(int\*\*, int, int, int)*

*Параметры:*

первый параметр – адрес первой строки матрицы

второй параметр – кол-во строк матрицы

третий параметр – кол-во столбцов матрицы

четвертый параметр – параметр в зависимости от которого будет определяться какие строки удалять, четные (0), нечетные (1)

*Возвращаемое значение –* кол-во удаленных строк

*Вспомогательные переменные:*

*i* – индекс строки, тип *int*

*s* – сумма эл-тов строки, тип *int*

*c –* счетчик удаленных строк, тип *int*

Текст программы

#include <stdio.h>

#include <stdlib.h>

/\*объявление функций\*/

int scanN();

void scan(int\*, int);

void print(int\*, int);

void print2(int\*\*, int, int);

void scan2(int\*\*, int, int);

int\*\* create2(int\*\*, int, int);

void free2(int\*\*, int);

void delete\_line(int\*\*, int, int);

int delete\_lines(int\*\*, int, int, int);

int sum\_line(int\*, int);

int main()

{

/\*объявление переменных и ввод некторых с помощью ф-ии scanN\*/

int \*\*a, \*\*b, z, i, n1=scanN(),m1=scanN(),n2=scanN(),m2=scanN();

/\*выделение памяти под матрицы\*/

a = create2(a, n1, m1);

b = create2(b, n2, m2);

/\*ввод матриц\*/

scan2(a, n1, m1);

scan2(b, n2, m2);

/\*печать матриц\*/

print2(a, n1, m1);

printf("\n");

print2(b, n2, m2);

printf("\n");

/\*удаление строк с четной суммой эл-тов из первой матрицы\*/

z=delete\_lines(a, n1, m1, 0);

printf("%d line(s) was deleted\n\na\n",z);

/\*уменьшаем общее кол-во строк первой матрицы на кол-во удаленных,

то есть фиксируем, что строк стало меньше, чтобы не освобождать память

одного указателя несколько раз\*/

n1-=z;

/\*печать матрицы\*/

print2(a, n1, m1);

/\*удаление строк с нечетной суммой эл-тов из второй матрицы\*/

z=delete\_lines(b, n2, m2, 1);

printf("\n%d line(s) was deleted\n\nb\n",z);

n2-=z;

/\*печать матрицы\*/

print2(b, n2, m2);

free2(a, n1);

free2(b, n2);

return 0;

}

/\*ф-ия ввода натурального число\*/

int scanN()

{

int n;

do

scanf("%d",&n);

while (n<1);

return n;

}

/\*фунция ввода одномерного массива\*/

void scan(int\* a, int n)

{

int i;

for(i=0; i<n; i++)

scanf("%d", &a[i]);

//a[i] =rand()%100;

}

/\*фунция печати одномерного массива\*/

void print(int\*a, int n)

{

int i;

if (n<1)

return;

for(i=0; i<n; i++)

printf("%3d ", a[i]);

printf("\n");

}

/\*фунция ввода двумерной динамической матрицы\*/

void scan2(int\*\* a, int n, int m)

{

int i;

for(i=0; i<n; i++)

/\*используем ф-ию scan, чтобы ввести каждую строку\*/

scan(a[i], m);

}

/\*фунция вывода двумерной динамической матрицы\*/

void print2(int\*\*a, int n, int m)

{

int i;

for(i=0; i<n; i++)

/\*используем ф-ию print, чтобы ввести каждую строку\*/

print(a[i], m);

}

/\*фунция выделения памяти под двумерную матрицу\*/

int\*\* create2(int\*\*a, int n, int m)

{

int i,j;

/\*выделяем память под указатели на строки\*/

a = malloc(n\*sizeof(int\*));

/\*если память не выделилась, возвращаем NULL\*/

if (a==NULL)

return NULL;

for(i=0; i<n; i++)

{

/\*выделяем память под каждую строку,

если не выделилась, очищаем память

под уже выделенные строки и возвращаем NULL\*/

a[i] = malloc(m\*sizeof(int));

if (a[i]==NULL)

{

for(j=0;j<i;j++)

free(a[j]);

free(a);

return NULL;

};

};

/\*возращаем указатель\*/

return a;

}

/\*функия особождения памяти, выделенной под матрицу\*/

void free2(int\*\*a, int n)

{

int i;

for(i=0;i<n;i++)

/\*очищаем каждую строку\*/

free(a[i]);

/\*очищаем указатели на строки\*/

free(a);

}

/\*ф-ия удаления определенной строки по ее индексу\*/

void delete\_line(int\*\*a, int n, int m)

{

int i;

free(a[n]);

for(i=n; i<m-1; i++)

a[i] = a[i+1];

}

/\*функция нахождения суммы эл-тов определенной строки по индексу\*/

int sum\_line(int\* a, int n)

{

int i,s=0;

for (i=0;i<n;i++)

s+=a[i];

return s;

}

/\*функция удаления строк в зависимости от суммы их эл-тов, четных или нечетных\*/

int delete\_lines(int\*\*a, int n, int m, int p)

{

int i,s,c=0;

/\*проход циклом по всем строкам\*/

for(i=0; i<n; i++)

{

/\*вызываем ф-ию нахождения суммы строки и записываем в s\*/

s=sum\_line(a[i],m);

/\*определяем четность числа\*/

if (s%2==p)

{

/\*удаляем строку\*/

delete\_line(a,i,n);

/\*подсчитываем удаленных строки\*/

c++;

/\*уменьшаем i, чтобы не пропустить потенциальную строку к удалению\*/

i--;

/\*фиксируем, что строк стало меньше\*/

n--;

};

};

/\*возвращаем кол-во удаленных строк\*/

return c;

}

Определить функцию *Integral()* для приближенного вычисления определенного интеграла вида методом трапеций.Использовать эту функцию для вычисления значений двух интегралов, передавая подынтегральную функцию в функцию *Integral()* в качестве параметра., , N=40

Формула трапеций

*Исходные данные:*

нижний и верхний пределы интегрирования задаются константами

подынтегральные функции описываются функциями в программе.

*Результирующие данные:*

значение интеграла – вещественное число

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
|  | https://www4d.wolframalpha.com/Calculate/MSP/MSP231614841cha0522492a00004505gecgehei6ie2?MSPStoreType=image/gif&s=6  [Wolframalpha](https://www.wolframalpha.com/input/?i=integrate+2x%2Fe%5E%282x%29+dx+from+x%3D-1+to+3) |  |
|  | https://www4d.wolframalpha.com/Calculate/MSP/MSP46881dhcg306bhg1682g0000455b5632144fe416?MSPStoreType=image/gif&s=15[Wolframalpha](https://www.wolframalpha.com/input/?i=integrate+sqrt%28x%5E2%2B1%29+dx+from+x%3D-3+to+2) |

Результат работы программы не совпадает с ожидаемым из-за малого числа равных отрезков или же частей, на которые разбивается интервал для вычисления площади под кривой.

Схема программы

начало

окончание

integral(40, -3, 2, f2)

integral(40, -1, 3, f1)

вывод значений, вернувших ф-ми

Вспомогательные алгоритмы

Алгоритм вычисления интеграла

*Входные данные:* кол-во частей на которые поделен отрезок, начало отрезка, конец отрезка, имя функции.

*Результирующие данные*: нет

Схема алгоритма:

integral(n, a, b, f)

s\*h

Цикл 1

a=a+h, b-2h ш. h

Цикл 1

s = s+f(a)

h =

s =

Этот алгоритм описывается в программе функцией

*double integral(int, double, double, double(\*)(double))*

*Параметры:*

первый параметр – кол-во частей

второй параметр – начало отрезка

второй параметр – конец отрезка

второй параметр – адрес на функции

*Возвращаемое значение –* значение интеграла

*Вспомогательные переменные:*

*h* – длина части, тип *double*

*s –* сумма площадей трапеций, тип *double*

Текст программы

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

/\*объявление ф-ий\*/

double f1(double);

double f2(double);

double integral(int, double, double, double(\*)(double));

int main()

{

/\*печать значения интеграла\*/

printf("%lf\n%lf",integral(40, -1, 3, f1),integral(40, -3, 2, f2));

return 0;

}

double f1(double x)

{

return 2\*x/exp(2\*x);

}

double f2(double x)

{

return sqrt(x\*x+1);

}

/\*ф-ия вычисления интеграла методом трапеций\*/

double integral(int n, double a, double b, double (\*f)(double))

{

double h =(b-a)/n, s=((f(a)+f(b))/2);

for(a+=h; a<b-h; a+=h)

s+=f(a);

return h\*s;

}