Курсова робота

на тему:

Обчислення визначених інтегралів за формулами прямокутників, трапецій та Сімпсона.

Зміст

Вступ

1. Огляд літератури

2. Формулювання задачі

3. Алгоритм розв’язку задач

4. Опис програми на мові Turbo С

5. Інструкція користувачеві програми

6. Контрольні приклади та аналіз їх реалізацій

Висновок

Список використаної літератури

Додатки

Вступ

Обчислювальну техніку останніми разами широко застосовують у всіх сферах діяльності людини. Вона стала каталізатором науково–технічного процесу. Історія розвитку обчислювальної техніки починається з 1945 року, коли американський вчений Фал Нейман та інші визначили основні принципи побудови ПК (так звані основні принципи програми управління).

У 1946 р. в Пенсильванському університеті було побудовано першу машину – “Машину 1-го покоління”. Найхарактернішою ознакою цих машин було використання електричних ламп. Потім з’явилися зовнішні запам’ятовуючі пристрої – пристрої вводу інформації. Лампові машини мали великі габарити, у них була мала ємкість оперативної пам’яті, було слабке математичне забезпечення. Пізніше з’явилися напівпровідникові пристрої. Ці машини були більш надійними, мали менші габарити. На початку 60-тих років була розроблена технологія виробництва інтегральних схем. Це вирішило проблеми надійності і цінноссті машин ПК.

З 1968 р. Починається ІІІ покоління ПК. Використовується постійна пам’ять. Важливим кроком в цьому поколінні є використання дисплея, з’явилась уже клавіатура. З середини 80-тих років поряд з машинами ІІІ-го покоління з’являються машини ІV-го покоління. Характерною особливістю ІV-го покоління є використання інтегральних систем.

Обчислювальні машини можна використовувати ефективно лише за умови глибокого знання чисельних методів математики.

Бурхливий розвиток ПК сприяв широкому процесу математизації науки, техніки і господарства в цілому. Саме розробка і застосування математичних методів розв’язування прикладних задач на базі ПК є предметом сучасної прикладної математики.

Математика – одна з найдавніших наук – виникла з практичних потреб людини.

Застосування швидкодіючих ПК для розв’язування складних прикладних задач сформувало новий спосіб проведення теоретичних досліджень на базі математичних моделей – обчислювальний експеримент.

Виділяють 5 етапів технологічного циклу обчислювального експерименту, побудова математичної моделі задачі, розробка методу розв’язування математичної моделі, програмування, розрахунки на ПК, аналіз результатів розрахунків і застосування.

Завдяки обчислювальному експерименту вдалося розв’язати не тільки багато важливих прикладних задач, а й перевірити гіпотези класичної математики.

Відомо топологічною задачею є проблема 4-рьох фарб. Ця гіпотеза була підтверджена в 1976 р. американським математиком Аппелем і Хакеном за допомогою ПК. В даний час відбувається швидкий розвиток усіх галузей науки. Людство прагне зрозуміти себе та світ навколо себе. Тому виникають сотні нових задач, які зводяться до використання числових методів. В багатьох випадках для їхнього розв’язання використовується чисельне інтегрування, а саме методи прямокутників, трапецій, Сімпсона. Тому важливим і актуальним є розроблення програмного забезпечення, яке дозволяє виконувати обислення за допомогою цих формул.

Метою даної курсової роботи є практичне закріплення, поглиблення та розширення знань і вмінь, набутих мною при вивченні дисциплін “Основи програмування” та “Проблемно-орієнтовані мови програмування”.

Тема курсової роботи – “ Обчислення визначених інтегралів за формулами прямокутників, трапецій та Сімпсона”.

1. Огляд літератури

Підходячи до реалізації питань курсової роботи , не можливо обійтись без таких апаратних можливостей та структур як масиви, функції, структури, символьні рядки, та файли. Тому давайте ознайомимось із структурними можливостями мови програмування С/С++.

Масиви. Масив – це набір елементів одного і того ж типу. Елементи зберігаються в пам’яті у вигляді послідовності, а доступ до них відбувається з допомогою цілочислового індекса. В мові С перший елемент масиву має індекс 0, звідси випливає , що кінцевий елемент масиву , який зберігає n елементів. Має індекс n-1 . Контроль за правильним використанням індексів покладається на програміста, оскільки ні компілятор , ні виконуюча програма не відстежують за цим процесом. Працюючи з масивами ми стикаємось із великим об’ємом інформації. Досить часто масиви пропонують найкращі способи маніпуляції , зручні та ефективні. Ім’я масиву є адресою його першого елемента. Звертання до елементів масиву відбувається через індексну або вказівникову форми.

Структура оголошення масиву :

<тип > <ім’я >[розмір];

Приклад:

int ar[10][6];

int temp[n][m];

void main (void)

{

int temp[n][m];

temp[0][0]=2;

ar[0][0]=2;

};

Ім’я масиву є адресою його першого елемента.

Функції. Функція – це самостійна одиниця програмного коду,яка розроблена для вирішення конкретної задачі, яка супроводжується обов’язково ( ). Уміло володіючи її можливостями у нас в руках опиняється потужний програмний інструмент.Користуючись аргументами для передачі значень функцій , а для повернення результату реалізації функція використовує ключове слово return . Якщо функція повертає якесь значення , тип якого не int , ми повинні описати тип функції в описі даної функції в розділі оголошень викликаючи функцій. Якщо ми хочемо , щоб функція реагувала на змінні що викликаються функцією , потрібно використовувати адреси і вказівники.

Стандарт ANSI С пропонує прототипні функції , що є потужним покращенням мови С, яке дозволяє компіляторам перевіряти чи правильно вказана кількість типів аргументів при виклику функції. Функція С здатна викликати саму себе , що називається рекурсією. Деякі задачі можна вирішувати мотодом рекурсії, але цей метод бути не ефективним з точки зору використання пам’яті і часу.

Використовуючи функції ми одержуємо деякі переваги такі як :

а).необхідність багаторазового повтору у програмах одного і того програмного коду. Якщо в програмі необхідно вирішувати одну і ту ж задачу декілька раз, нам достатньо написати відповідну функцію всього лиш раз. Програма використовуватиме дану функцію там, де це буде необхідно, а ми можемо використовувати цю функцію в декількох програмах. Навіть тоді, коли задача в програмі виконується лише раз , використання функції цілеспрямоване, поскільки при цьому повишається рівень модульності, через що програма стає більш зрозумілою при читанні, а ще в неї легко можна вноси зміни і доповнення.

Функції мають наступну структуру:

< тип результату > < ім’я функції >( <оголошення формальних параметрів>)

{ < опис внутрішніх змінних >;

<операції тіла функції>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

<тип результату>;

Функція може повертати в точку її виклику значення будь-якого типу в тому числі користувацького крім масиву (символьного рядка і функції) , але може повертати вказівник на довільний тип , включаючи масив і функцію. Внутрішні змінні функції описуються на початку і можуть використовуватися тільки в межах цієї функції. У разі , якщо функція повинна повертати значення якогось типу в операторі return вказують вираз значення якого є результатом роботи функції.

Приклад:

int Min3(int a, int b, int c)

{

int min;

min=a<b?a:b;

return min<c?min :c;

};

Символьні рядки. Символьні рядки – це один із найкорисніших і важливих типів даних в мові С. Бібліотека функцій в мові С пропонує широкий спектр функцій для читання і запису, копіювання, порівняння, комбінування, пошуку і виконання інших корисних операцій із рядками. Символьний рядок – це один із різновидів масиву типу char, який суттєво вирізняється тим, що в кінці кожного символьного рядка, як такого стоїть ескейп-символ «нуль-символ(\0)». Із символьних рядків можна утворювати складні синтаксичні структури. До елементів рядка можна звертатись двома способами : через індекси або через вказівники. Використовуючи вказівники ми оперуємо чудовим інструментом доступу до елементів, який набагато ефективніший за індексну форму. Використовуючи вказівники ми можемо звертатись до елементів не тільки як до символа, але й як адресу , на яку вказує вказівник.

Рядок може зберігатись в символьному масиві. Рядок може бути представлений у вигляді рядкової константи, в якій символи за виключенням нульового вставлені у подвійні лапки. Нуль-символ проставляє компілятор. Довжина рядка виміряють за допомогою функції strlrn( ), не враховуючи нуль-символ . Рядкові константи відомі ще як рядкові літерали і можуть використовуватись для ініціалізації символьних масивів. Розмір масиву повинен містити принаймі на один символ більше , щоб включити нуль-символ. Символьні константи використовують для ініціалізації вказівників на тип char.Для ідентифікації опрацьовуючого рядка

функції використовують вказівник на перший символ даного рядка . В загальному випадку відповідним фактичним параметрам являється ім’я масиву, змінна типу вказівник або рядок, поміщений у лапки.

Функції gets( ) i puts( ) ,відповідно, читають рядок вводу і показують рядок виводу. Вони являються частиною сімейства stdio.h.

Бібліотека С включає в себе декілька функцій опрацювання рядків. В умовах дії стандарту ANSI C ці функції оголошені в заголовному файлі string.h .Дана бібліотека містить декілька функції опрацювання символів; вони оголошені в заголовному файлі ctype.h . Існують функції перетворення символьного представлення чисел відповідно в тип int , long , double як atoi( ), atol( ), atof( ), і зворотні strtol( ), strtoul( ), strtod( ).

Структури. Однією із найважливіших дій при розробці програми являється вибір підходящого способу подання даних. В багатьох випадках простих змінних або навіть масивів недостатньо. Мова С дозволяє розширити можливості представлення даних за допомогою змінних типу структури. Структура в мові С являється досить гнучким засобом в своїй базовій формі, вона надає можливість представлення різнотипних даних, при чому дозволяє створювати нові форми. Структури починаються зарезервованим словом struct і мають таку структуру :

Struct <тег > {

<тип 1> <поле1>;

<тип 2 > <поле 2>

…………………….

< тип К > <поле К>

};

де < ТЕГ > --- унікальне поле структури , яке визначає дану структуру .

Поля структур --- це елементи з яких складається структура , кожне поле має свій тип . Тип поля --- довільний , простий або складений допустимий для С.

Розмір структури (обсяг оперативної пам’яті) , яку займає структура буде більший (рівний) за розмір усіх її полів. Тому реальний розмір структури визначають операцією size of(<назва структури>) . Шаблони структур фактично створюють новий користувацький тип, але для них не виділяється місце в оперативній пам’яті. Пам’ять виділяється тільки для структурних змінних, які можна оголошувати разом зі шаблоном , або окремо , посилаючись на шаблон.

Для ідентифікації шаблону і оголошення змінних даного типу можна використовувати дескриптори. Операція приналежності (.) дозволяє получити доступ до окремих елементів структури через використання міток шаблона структури.

Якщо у нас є вказівник на конкретну структуру, для доступу до окремого елемента структури ми можемо використати вказівник і складену операцію приналежності (->) замість імені і операції крапка. Здобути адресу структури можна з допомогою &. На відміну від масивів , ім’я структури не може бути адресою структури.

Традиційно функція , орієнтована на роботу зі структурими , використовують вказівники на структури в якості аргументів. Сучасна версія С допускає передачу структур в якості аргументів, використання структур в якості повертаючи значень і виконання операції присвоєння над структурами одного і того ж типу. Подібним синтаксисом володіють об’єднаня. Прикладом який демонструє структуру може бути :

struct book {

char avtor[60];

char name[120] ;

int year ;

} book1,book2;

Файли. Файли являються частиною теперішніх комп’ютерних систем. Вони використовуються для зберігання програм, даних, кореспонденції, форм, графічних даних та багато іншої інформації.Будучи програмістом , ми повинні знати , як писати програми, які створюють, записують та читають файли .

Часто у нас виникає необхідність в програмах , які можуть зчитувати інформацію із файлу або записувати результати виконання у файл. Однією із таких форм обміну даними між програмою і файлом є пере адресація файлу.

Мова С пропонує більш потужніший метод обміну даними з файлами. Він дозволяє відкривати файли із програми , а тоді з допомогою спеціальних функцій вводу-виводу виконає читання і запис в цей файл.

Файл являє собою іменований розділ пам’яті , зазвичай розміщений на диску. Однак для операційної системи файл являється більш складним об’єктом.

С розглядає файл як неперервну послідовність байтів, кожен з яких може бути прочитаний індивідуально. Це відповідає структурі файлу в операційній системі Unix, з якої мова С бере свій початок. Оскільки інші операційні системи не відповідають цій моделі , стандарт ANSI С пропонує два способи представлення файлів: текстове представлення і двійкове представлення.

Програма на мові С розглядає ввід як деякий потік даних. Джерелом цього потоку може бути файл, пристрій вводу(клавіатура) або навіть вивід в іншу програму . Програма на С трактує вивід як потік байтів , місцем призначення яких може бути файл, пристрій відображення і тому подібне . Як С інтерпретує вхідний або вихідний потік байтів, залежить від того , яку функцію вводу-виводу ми використовуємо. Програма може читати і зберігати

байти без змін , з другої сторони , вона може інтерпретувати байти як символи , які в свою очередь , можна розглядати як звичайний текст, або текстове представлення чисел. Аналогічно для двійкового значення без зміни або перетворення в текст або текстове представлення чисел.

Якщо є числові , які необхідно зберегти , а потім використати без втрати точності їх представлення , використовують двійковий режим і функції fread( ) i fwrite( ). Якщо зберігати текстову інформацію і хочемо створити файл , який легко переглядається з допомогою звичайних текстових редакторів, скористаємося текстовим режимом і функціями getc( ) i fprint( ).

Щоб отримати доступ до файлу , потрібно створити вказівник на файл (типу FILE \*) і пов’язати його з конкретним ім’ям файлу.

C підтримує операції введення/виведення даних через відповідні набори бібліотечних функцій , при чому вони можуть бути реалізовані на трьох рівнях :

А). високий --- це так звані потокоорієнтовані операції буферизованого вводу/виводу , їх прототипи містяться у заголовному файлі «stdio.h», які є дуже мобільні.

Б). низький --- (не підтримуваний стандартом) базується на операціях MS DOS , прототипи містяться у заголовном уфайлі <io.h>.

В). консольний --- базується на засобах BIOS , прототипи в заголовному файлі < conio.h>.

Важливо пам’ятати , що С сприймає поняття кінця файлу і виконує читання файлу вциклі до тих пір доки не досягне кінця файлу. Функції вводу у мові С не знаходять кінець файлу до того часу , доки не попробують читати символи , що йдуть за кінцем файлу.

Ввід і вивід файлів буферезований . Стандартний пакет вводу-виводу автоматично створює буфер для вводу і виводу, що прискорює передачу даних. Функція fopen( ) відкриває файл для стандартного вводу-виводу і створює структури даних , які пизначені для зберігання інформації про файл і буфер. Повертає вказівник на цю структуру даних , а сам вказівник використовується іншими функціями. Розрізняють декілька режимів виконання для функції fopen( ):

“r”—відкриття текстового файлу для читання;

“w”—відкриває текстовий файл для запису, відрізаючи довжину існуючого файлу до нуля, або створює файл якщо такого не існує;

“a”--- відкриває текстовий файл для запису, добавляючи дані в кінець існуючого файлу, або створює файл якщо такого не існує;

“r+”--- відкриває текстовий файл для обновлення;

“w+”--- відкриває текстовий файл для обновлення, виконавши спочатку обтинання файлу до нульової довжини , якщо він існує , або створює файл , якщо той ще не існує;

“a+”--- відкриває текстовий файл для обновлення, добавляючи дані в кінець існуючого файлу, або створює файл якщо такого не існує,але при цьому можна читати весь файл , однак записана інформація добавляється в кінецьфайла;

Функції feof( ) і ferror( ) повідомляють про причини невдалого завершення операцій вводу-виводу.

До задач числових методів відносяться прикладні задачі, розв’язок яких містить числову інформацію. Вони зводяться до математичрих обчислювальних методів. Виділяють наступні етапи розв’язування задач числових методів, які наведені [1,2]:

побудова математичних моделей (математичне формулювання задачі) - охоплює найважливіші для даної задачі сторони, явища;

вибір методу розв’язування – для найпростіших задач знаходять аналітичний розв’язок, складніші розв’язуються наближеними методами, зокрема числовими;

алгоритмізація процесу – складання алгоритму розв’язку задач (якщо задача розвязується на ПК, складається програма);

виконання обчислення на ПК чи вручну;

аналіз результатів.

Числові методи для розв’язання різноманітних задач почали широко використовуватися в 50-х рр. ХХ століття, коли починається розвиток комп’ютерної техніки. Громіздкі обчислення з появою електронно-обчислювальних машин перестали бути такими складними, адже їх виконувала машина.

Формула прямокутників.

Ідея формули прямокутників полягає в тому, що на малому відрізку [x;x+h] площа криволінійної трапеції наближено рівна площі прямокутника з основою (x;x+h) і висотою рівною ординаті будь-якої точки ,яка належить відрізку [x;x+h].



(1)



В залежності від вибору точки отримаємо різновидності формул прямокутників (1).



Розіб’ємо відрізок [a;b] на n рівних частин точками a=x; x; x, причому x=x+h; x=x+2h; x=x+nh.



(2)



На кожному відрізку [x;x+h] замінемо відповідну криволінійну трапецію на прямокутник висоту якого можна визначити по різному.



Формула „лівих” прямокутників.

Якщо за висоту прямокутника на кожному відрізку [x;x+h] вибрати ординату в лівому кінці, тобто y, то криволінійна трапеція заміниться на ступінчасту фігуру, площу якої можна приймати за площу криволінійної трапеції.



Формула „лівих” прямокутників:

(3)



Формула „правих” прямокутників.

Якщо за висоту прямокутника вибрати ординату правого кінця, тобто y(мал.3), то отримаємо формулу „правих” прямокутників (4).



Формула „правих” прямокутників:

(4)



Формула трапецій.

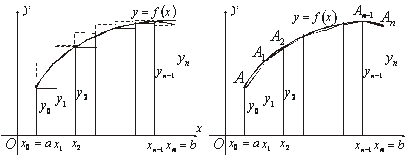
Вона полягає в тому, що на відрізку [x;x+h] до кривої y=f(x) замінюють хордою, яка стягує кінці цієї дуги, тобто використовують лінійну інтерпуляцію функції y=f(x). При цьому площа криволінійної трапеції замінюється площею криволінійної трапеції з основами і висотою h.



(5)



Розіб'ємо відрізок [a;b] на n рівних частин і замінимо дугу кривої ламаною (мал.1).



Мал.1.Розбиття площі фігури для формули трапецій

Формула трапецій:

(6)



Формула Сімпсона (парабол)

Метод Сімпсона найпоширеніший і широко застосовний для програмування. Його суть полягає в наближенні підінтегральної функції відрізками парабол

Отже, розглянемо спочатку інтеграл

, де



- парабола; - деякі параметри (або числа).

Розбиваємо відрізок [a;b] на парне число частин n=2m. На кожному з відрізків замінюємо дугу кривої y=f(x) параболою, яка проходить через 3 точки M(x;y), M (x;y), M(x;y).



Отримані фігури називаються параболічними трапеціями.

y=Ax+Bx+C (7)



S= (8)



Формула Сімпсона:

(9)



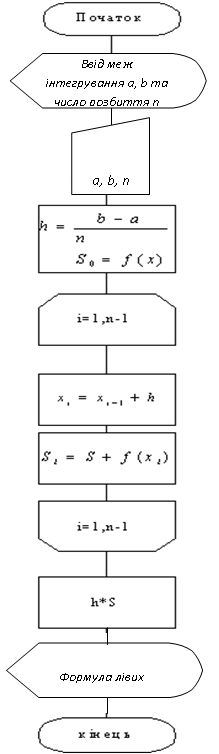
2. Формулювання задачі

Завдання курсової роботи відноситься до класу задач з числових методів. Метою виконання роботи є отримання результатів обчислення визначених інтегралів за формулами прямокутників, трапецій та Сімпсона для подальшого їх використання у різних типах задач.

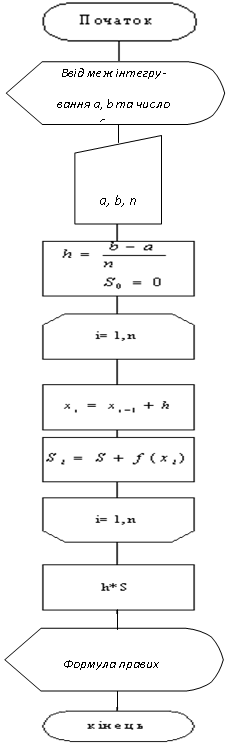
Вхідними даними для програми є межі інтегрування, а також число розбиття проміжку. Вихідні дані представлені у вигляді значень інтегралів обчисленими чотирма методами.

При обчисленні інтегралу логарифмічної функції розв’язок існує тільки тоді, коли межі інтегрування є додатні. Контроль вводу меж для данаї функції передбачений в програмній реалізації даної задачі.

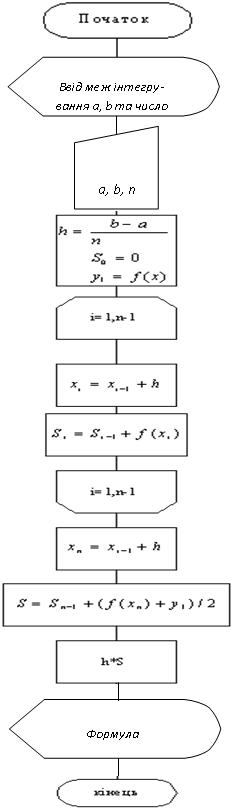
3. Алгоритми розв’язку задач



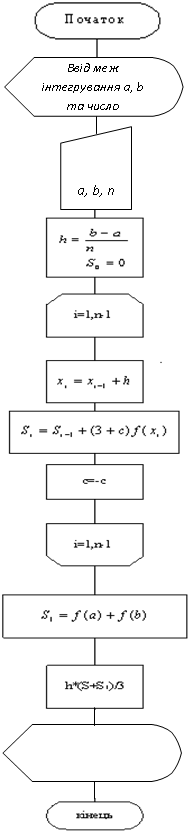
Формула „лівих” прямокутників



Формула „правих” прямокутників



Формула трапецій



Формула Сімпсона

4. Опис програми мові Turbo С

Загальна характеристика програми:

Ім’я програми – Kursak.exe

Назва файлу – Kursak.cpp

Мова програмування – Turbo С

Об’єм програми – текстових рядків або байт – 389 текстових рядків або 8497 байт.

Призначення програми:

Ця програма призначена для виконання операцій обчислення визначених інтегралів.

Вхідна інформація:

Вхідні дані вводяться з клавіатури: межі інтегрування(аа,bb) та

крок інтегрування(xx).

Вихідна інформація:

Виводиться у вигляді стрічки з повідомленням про результат

роботи даної програми.

Ідентифікатори програми

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ідентифікатор змінної | Зміст у програмі | Тип змінної | Розмір (байт) | Спосіб формування |
| aa,bb | межі інтегрування | int | 2 | вхідна |
| vubir1 | номер функції | int | 2 | вхідна |
| hh | крок інтегрування | double | 8 | проміжна |
| xx | нижня межа інтегрування | double | 8 | проміжна |

5. Інструкція користувачеві

До програми входить файл:

Kursak.exe – програма для DOS.

Для початку роботи потрібно запустити exe-файл. Далі необхідно вибрати необхідну дію, ввівши букву, що вказана в головному меню, яке показано на (мал. 1).



Мал.1

При натисненні клавіші F зявляється меню вибору функції (мал.2).



Мал.2

Після вибору функції відбувається ввід даних. Формат вводу представлений на (мал.3).



Мал.3

При роботі в даній програмі можливе натиснення невірної клавіші. В такому виподку зявляється повідомлення (мал.4).



Мал.4

6. Контрольні приклади та аналіз їх реалізації

Приклад 1: Інтегрування функції exp(x).



Приклад 2: Інтегрування функції sin(2\*x)+x.



Приклад 3: Інтегрування функції log(x\*x).



Дані методи інтегрування пов’язані з використанням наближених формул.

Тому результати, які ми отримуємо не є точними. Виходячи з геометричного

змісту цих методів найменша похибка буде в методі Сімпсона. Параболічні

трапеції, сумування площ яких використовується в методі парабол, є

найбільш наближеними до фігур, які утворює задана функція.

Висновки

У даній курсовій роботі я закріпив знання з дисципліни “Проблемно-орієнтовані мови програмування” і виконав поставлене в курсовій роботі завдання, а також реалізував програмне забезпечення на мові програмування Turbo C.

Даний програмний продукт при потребі може бути вдосконалений та розширений.

Список використаної літератури

1.Бахвалов Н.Б., Жидков Н.П., Коббельков Г.М. Числові методи.-М.:Наука,1987

2.Даниліна Н.І,, Дубровська Н.С., Кваша О.П. Числові методи.-М.:Вища шк.,1976

3.Рублев А.Н. Линейная алгебра. М.: Высшая школа, 1968.

4.Керниган Б., Ритчи Д. Язык программирования Си. - М.: Финансы и статистика, 1992.

5. Прата С. Язык программирования С. — М., СПб., К.: ДиаСофт, 2002.

6. Брєдлі Л., Ейткен П. Освой самостоятельно С за 21 день.-М.:Изд.дом “Вильямс”,2003.

7.Шпак З. Я. “Програмування мовою С”, 2006.

Додатки

Текст програми на мові Turbo C

#include<stdio.h>

#include<conio.h>

#include<math.h>

//Функція задання кольору фону

void Fon(void)

{

textcolor(15); textbackground(BLUE);

return;

}

//Функція задання кольору букв

void Bykvu(void)

{

textcolor(1);

textbackground(15);

return;

}

//Заголовок

void Poc(void)

{

Fon();

clrscr();

printf("\n\n\n\n\n\n\t\t\t╔══════════════════════════╗");

printf("\n\t\t\t║ ║");

printf("\n\t\t\t║ Курсова робота ║");

printf("\n\t\t\t║ на тему: ║");

printf("\n\t\t\t║ Обчислення інтегралу за ║");

printf("\n\t\t\t║ за допомогою формул прямокутників, ║");

printf("\n\t\t\t║ трапецій, Сімпсона. ║");

printf("\n\t\t\t║ ║");

printf("\n\t\t\t╚════════════════════════════════╝");

getch();

clrscr();

return;

}

//Головне меню

void Menu(void)

{

Fon();

clrscr();

Bykvu(); printf("\a\n\n\n\n\t\t╔════════════════════════════════╗");

printf("\n\t\t║ ║");

printf("\n\t\t║\t\t ");

cprintf(" МЕНЮ ");

printf("\t\t ║");

printf("\n\t\t║ ║");

printf("\n\t\t║ ");

cprintf("\*1\*");

printf(" ");

cprintf("Інтегрування функції");

printf(" --> ");

cprintf("натисніть");

printf(" ");

cprintf("F");

printf(" ║");

printf("\n\t\t║ ║");

printf("\n\t\t║ ");

cprintf("\*2\*");

printf(" ");

cprintf("Вихід");

printf(" -----------------> ");

cprintf("натисніть");

printf(" ");

cprintf("V");

printf(" ║");

printf("\n\t\t║ ║");

printf("\n\t\t╚═════════════════════════════════╝");

return;

}

//Меню вибору функції

void Vubir\_Fun(void)

{

Fon();

clrscr();

Bykvu(); printf("\a\n\n\n\n\t\t╔════════════════════════════════╗");

printf("\n\t\t║ ║");

printf("\n\t\t║ ");

cprintf("Виберіть функцію:");

printf(" ║");

printf("\n\t\t║ ║");

printf("\n\t\t║ ");

cprintf("\*");

printf(" ");

cprintf("exp(x)");

printf(" --------------> ");

cprintf("натисніть");

printf(" ");

cprintf("1");

printf(" ");

printf("║");

printf("\n\t\t║ ║");

printf("\n\t\t║ ");

cprintf("\*");

printf(" ");

cprintf("sin(2\*x)+x");

printf(" ----------> ");

cprintf("натисніть");

printf(" ");

cprintf("2");

printf(" ");

printf("║");

printf("\n\t\t║ ║");

printf("\n\t\t║ ");

cprintf("\*");

printf(" ");

cprintf("log(x\*x)");

printf(" ------------> ");

cprintf("натисніть");

printf(" ");

cprintf("3");

printf(" ");

printf("║");

printf("\n\t\t║ ║");

printf("\n\t\t║ ");

cprintf("\*");

printf(" ");

cprintf("повернення в меню");

printf(" ---> ");

cprintf("натисніть");

printf(" ");

cprintf("4");

printf(" ");

printf("║");

printf("\n\t\t║ ║");

printf("\n\t\t╚═════════════════════════════════╝");

return;

}

//ввід меж інтегрування

void Megi(void)

{

Fon();

clrscr();

Bykvu();

printf("\n\t\t══════════════════════════\n");

printf("\t\t");

cprintf("Введіть межі інтегрування:");

printf("\n\t\t══════════════════════════\n");

return;

}

//ввід кроків

void Rozb(void)

{

printf("\n\n\t\t═════════════════════════\n");

printf("\t\t");

cprintf("Введіть кількість кроків:");

printf("\n\t\t═════════════════════════\n");

return;

}

//функція exp(w)

double fun1(double w)

{

return exp(w);

}

double fun2(double w)// функція sin(2\*w)+w

{

return sin(2\*w)+w;

}

// функція log(w\*w)

double fun3(double w)

{

return log(w\*w);

}

//обчислення за формулою „лівих” прямокутників

double Liv\_pram(double (\*fun)(double),double x,double h,int n)

{

double s;

int i;

s=fun(x);

for(i=0;i<n-1;i++){

x+=fabs(h);

s+=fun(x);

}

return printf("\n\tФормула лівих прямокутників: %g",(fabs(h)\*s));

}

//обчислення за формулою „правих” прямокутників

double Prav\_pram(double (\*fun)(double),double x,double h,int n)

{

int i;

double s=0;

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

x+=fabs(h);

s+=fun(x);

}

return printf("\n\tФормула правих прямокутників: %g",(fabs(h)\*s));

}

//обчислення за формулою трапецій

double Trap(double (\*fun)(double),double x,double h,int n)

{

int i;

double s=0,y1;

y1=fun(x);

for(i=0;i<n-1;i++){

x+=fabs(h);

s+=fun(x);

}

x+=fabs(h);

s+=(fun(x)+y1)/2;

return printf("\n\tФормула трапецій:\t\t %g",(fabs(h)\*s));

}

//обчислення за формулою Сімпсона

double Simp(double (\*fun)(double),double x,double h,int a,int b,int n)

{

double s=0,s1;

int i,c=1;

for(i=0;i<n-1;i++){

x+=fabs(h);

s+=(3+c)\*fun(x);

c=-c;

}

s1=fun(a)+fun(b);

return printf("\n\tФормула Сімпсона:\t\t %g",fabs(h)\*(s+s1)/3)

}

//повідомлення про помилку

void Pomulka(void)

{

Fon();

clrscr();

Bykvu();

printf("\a\a\a\n\n\n\n\t\t\t╔════════════════════════╗ \n");

printf("\t\t\t║\t\t\t\t ║\n");

printf("\t\t\t║\t\t\t\t ║\n");

printf("\t\t\t║");

printf(" ");

cprintf("Ви допустили помилку !!!");

printf(" ║\n");

printf("\t\t\t║\t\t\t\t ║\n");

printf("\t\t\t║ ");

cprintf(" Натиснули невірну клавішу !!!");

printf(" ║\n");

printf("\t\t\t║\t\t\t\t ║\n");

printf("\t\t\t║\t\t");

cprintf(" ENTER ");

printf(" ║\n");

printf("\t\t\t║\t\t\t\t ║\n");

printf("\t\t\t╚════════════════════════════════╝\n");

getch();

return;

}

//меню закінчення

void Zakin(void)

{

Fon();

clrscr();

Bykvu();

printf("\n\n\n\n\t\t\t╔═══════════════════════════╗ \n");

printf("\t\t\t║\t\t\t\t ║\n");

printf("\t\t\t║");

printf(" ");

cprintf(" Хочите закінчити роботу? ");

printf(" ");

printf("║\n");

printf("\t\t\t║\t\t\t\t ║\n");

printf("\t\t\t║ ");

cprintf("\*ТАК\*");

printf(" ");

printf(" --------> ");

cprintf("натисніть");

printf(" ");

cprintf("Y");

printf(" ║");

printf("\n\t\t\t║\t\t\t\t ║\n");

printf("\t\t\t║ ");

cprintf("\*НІ\*");

printf(" ");

printf(" ---------> ");

cprintf("натисніть");

printf(" ");

cprintf("N");

printf(" ║");

printf("\n\t\t\t║\t\t\t\t ║\n");

printf("\t\t\t╚════════════════════════════════╝\n");

return;

}

//головна функція

int main(void)

{

int aa,bb,nn;

int vubir1;

double hh,xx;

clrscr();

Poc();

l2: Menu();

switch (getch())

{

case 'F': case 'f': {

{

l1: Vubir\_Fun();

vubir1=getch();

fflush(stdin);

if (vubir1>52) {

Pomulka();

goto l1;

}

if(vubir1==52) goto l2;

l4: Megi();

printf("\t\ta=");

scanf("%d",&aa);

printf("\t\tb=");

scanf("%d",&bb);

if(vubir1==51)

if((aa<0)||(bb<0)){

printf("\t\tМежі мають бути додатні.Введіть нові.");

getch();

goto l4;

}

Rozb();

printf("\t\tn=");

scanf("%d",&nn);

hh=(double(aa-bb))/double(nn);

if(aa>bb) xx=bb;else xx=aa;

switch (vubir1)

{

case 49: {

Liv\_pram(fun1,xx,hh,nn);

Prav\_pram(fun1,xx,hh,nn);

Trap(fun1,xx,hh,nn);

Simp(fun1,xx,hh,aa,bb,nn);

getch();

break;

}

case 50: {

Liv\_pram(fun2,xx,hh,nn);

Prav\_pram(fun2,xx,hh,nn);

Trap(fun2,xx,hh,nn);

Simp(fun2,xx,hh,aa,bb,nn);

getch();

break;

}

case 51: {

Liv\_pram(fun3,xx,hh,nn);

Prav\_pram(fun3,xx,hh,nn);

Trap(fun3,xx,hh,nn);

Simp(fun3,xx,hh,aa,bb,nn);

getch();

break;

}

}

}

l3: Zakin();

switch (getch())

{

case 'N': case 'n': goto l2;

case 'Y': case 'y':return 0;

default:

Pomulka();

goto l3;

}

}

case 'V': case 'v': return 0;

default:

Pomulka();

goto l2;

}

}