МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНИЙ ПОСІБНИК

З ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ ТА ПРОГРАМУВАННЯ

Донецьк ДонНТУ 2008

МІНІСТЕРСТВООСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНИЙ ПОСІБНИК

З ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ ТА ПРОГРАМУВАННЯ

(для студентів спеціальностей ХТ і ТХР напрямку 7.091604)

Затверджено
на засіданні кафедри ВМ і П
протокол №9 від 4.04.08 р.

Затверджено на засіданні навчально-видавничої ради ДонНТУ
протокол № 4 від 19.05.08г.

Донецьк ДонНТУ 2008

УДК 681.3.06(071)

Методичний посібник з обчислювальної математики і програмування / Укладачі: І.В. Диннік, О.А.Тихонова - Донецьк: ДонНТУ, 2008 - 71 с.

Методичний посібник містить завдання і необхідний теоретичний матеріал до виконання лабораторних робіт з обчислювальної математики і програмування.

Розраховано на студентів спеціальностей ХТ і ТХР (напрямок 7.091604), що вивчають дисципліну «Обчислювальна математика і програмування».

Автори: И. В. Диннік, ст. викладач

О.А.Тихонова, асистент

Рецензент: І.Я.Зеленьова, доцент

ЗМІСТ

1. Основні етапи повної побудови алгоритму 5
2. Завдання до лабораторних робіт 12
2.1 Лабораторна робота №1
Тема : розгалужені обчислювальні процеси 12
2.2 Лабораторна робота №2
Тема : Циклічні обчислювальні процеси 23
2.3 Лабораторна робота №3
Тема : Робота з одномірними масивами 35
2.4 Лабораторна робота №4
Тема : Робота із двовимірними масивами 40
2.5 Лабораторна робота №5
Тема : Робота з файловими змінними 45
2.6 Лабораторна робота №6
Тема : Робота з підпрограмами 51
3. Методичні вказівки по роботі в середовищі програмування 61

Список використаної літератури 70

Додатка.
Д1.Зразок виконання титульного листа 71
Д2.Вимоги до оформлення лабораторної роботи 72

1.Основні етапи повної побудови алгоритму

Алгоритм - це одне з найважливіших понять обчислювальної математики. Можна визначити алгоритм як однозначно трактуєму процедуру рішення задачі. Процедура - це кінцева послідовність точно визначених кроків або операцій, для виконання кожної з яких потрібен кінцевий об'єм оперативної пам'яті й кінцевий час для обробки будь-яких вхідних даних. Основними етапами повної побудови алгоритму є:

1. Постановка задачі.
2. Побудова моделі.
3. Розробка алгоритму.
4. Перевірка правильності алгоритму.
5. Реалізація алгоритму.
6. Перевірка програми.
7. Складання документації.

Постановка задачі

Перш ніж вирішити задачу, її потрібно точно сформулювати. Це саме по собі не є достатнім для розуміння задачі, але воно абсолютно необхідно. Звичайно процес точного формулювання задачі зводиться до постановки правильних питань. Перелічимо деякі корисні питання для погано сформульованих задач:

- Чи зрозуміла термінологія, використовувана в попереднім формулюванні?
- Що дано?
- Що потрібно знайти?
- Як визначити рішення?

- Яких даних не вистачає й чи всі вони потрібні?
- Чи є якісь наявні дані марними?
- Які зроблені допущення?

Можливі й інші питання залежно від конкретної задачі. Часто після одержання повних або часткових відповідей на деякі з питання їх доводиться ставити повторно.

Побудова моделі

Задача чітко поставлена, тепер потрібно побудувати для неї математичну модель. Це дуже важливий крок у процесі рішення, і його треба добре обміркувати. Вибір моделі істотно впливає на інші етапи в процесі рішення.

Як можна догадатися, неможливо запропонувати набір правил, що автоматизують стадію моделювання. Більшість задач повинна розглядатися індивідуально. Проте існує кілька корисних керівних принципів. Вибір моделі - більшою мірою справа мистецтва. Вивчення вдалих моделей - це найкращий спосіб придбати досвід у моделюванні.

Приступаючи до розробки моделі, варто задати, принаймні, два основних питання:

1. Які математичні структури найбільше підходять для задачі?
2. Чи існують вирішені аналогічні задачі?

Друге питання, можливо, саме корисне у всій математиці. У контексті моделювання воно часто дає відповідь на перше питання. Дійсно, більшість розв'язуваних у математиці задач, як правило, є модифікаціями раніше вирішених. Більшість просто не має талант Ньютона, Гауса або Ейнштейна, і для просування вперед людині доводиться керуватися накопиченим досвідом.

При відповіді на перше питання потрібно описати математично, що відомо й що необхідно знайти. На вибір відповідної структури будуть впливати такі фактори, як:

1) обмеженість наших знань відносно невеликою кількістю структур,
2) зручність подання,
3) простота обчислень,
4) корисність різних операцій, пов'язаних з розглянутою структурою або структурами.

Зробивши пробний вибір математичної структури, задачу варто переформулювати в термінах відповідних математичних об'єктів. Цю модель можливо використати, якщо можна ствердно відповісти на наступні питання:

- ✓ Чи вся важлива інформація добре описана математичними об'єктами?
- ✓ Чи існує математична величина, асоційована із шуканим результатом?
- ✓ Чи виявлені які-небудь корисні відносини між об'єктами моделі?
- ✓ Чи можна працювати з моделлю? Чи зручно з нею працювати?

Розробка алгоритму

Як розробити гарний алгоритм? Із чого почати? В усіх є сумний досвід, коли дивишся на задачу й не знаєш, що робити. Є декілька загальних методів рішення задач, корисних для розробки алгоритмів.

Перший метод зв'язаний з переходом від рішення важкої задачі до рішення послідовності більше простих задач. Така процедура називається методом приватних цілей. Цей метод виглядає дуже розумно. Але, як і більшість загальних методів рішення задач або розробки алгоритмів, його не завжди легко перенести на конкретну задачу. Осмислений вибір більше простих задач - скоріше справа мистецтва або інтуїції, чим науки. Більше того, не існує загального набору правил для визначення класу задач, які можна вирішити за допомогою такого підходу. Міркування над будь-якою конкретною задачею починається з постановки питань. Приватні цілі можуть

бути встановлені, коли ми одержимо відповіді на наступні питання:

1. Чи можемо ми вирішити частину задачі? Чи можна, ігноруючи деякі умови, вирішити ту частину задачі, що залишилася?

2. Чи можемо ми вирішити задачу для окремих випадків? Чи можна розробити алгоритм, що дає рішення, що задовольняє всім умовам задачі, але вхідні дані якого обмежені деякою підмножиною всіх вхідних даних?

3. Чи є щось у задачі, що ми не досить добре зрозуміли? Якщо спробувати глибше вникнути в деякі особливості задачі, чи зможемо ми щось довідатися, що допоможе нам підійти до рішення?

4. Чи зустрічалися ми зі схожою задачею, рішення якої відомо? Чи можна видозмінити її рішення для рішення нашої задачі? Чи можливо, що ця задача еквівалентна відомій невирішеній задачі?

Другий метод розробки алгоритмів відомий як метод підйому. Алгоритм підйому починається із прийняття початкового припущення або обчислення початкового рішення задачі. Потім починається наскільки можливо швидкий рух «нагору» від початкового рішення в напрямку до кращих рішень. Коли алгоритм досягає такого виду, що більше неможливо рухатися наверх, алгоритм зупиняється. На жаль, ми не можемо завжди гарантувати, що остаточне рішення, отримане за допомогою алгоритму підйому, буде оптимальним. Цей «дефект» часто обмежує застосування методу підйому.

Третій метод відомий як відпрацювання назад, тобто починаємо з мети або рішення й рухаємося обернено в напрямку до початкової постановки задачі. Потім, якщо ці дії оборотні, рухаємося обернено від постановки задачі до рішення. Багато хто з вас робили це, вирішуючи головоломки .

Як тільки задача чітко поставлена й для неї побудована модель, можна приступитися до розробки алгоритму її рішення. Вибір методу розробки, що найчастіше сильно залежить від вибору моделі, може в значній мірі вплинути на ефективність алгоритму рішення. Два різних алгоритми можуть бути правильними, але дуже сильно відрізнятися по ефективності.

Існує тенденція: програмісти затрачають відносно невеликий час на стадію розробки алгоритму при створенні програми. Проявляється сильне бажання якнайшвидше почати писати саме програму. Цьому спонуканню не треба піддаватися. На стадії розробки потрібне ретельне обмірковування, варто приділити увагу всім етапам розробки. Як і варто очікувати, всі сім основних етапів не можна розглядати незалежно друг від друга.

Перевірка правильності алгоритму

Доказ правильності алгоритму - це один з найбільш важких, а іноді й особливо стомлюючих етапів створення алгоритму. Імовірно, найпоширеніша процедура доказу правильності програми - це прогін її на різних тестах. Якщо видані програмою відповіді можуть бути підтверджені відомими або обчисленими вручну даними, виникає спокуса зробити висновок, що програма «працює». У принципі це правильно, однак цей метод рідко виключає всі сумніви; може існувати випадок, де програма не буде працювати. Потрібно підкреслити той факт, що правильність алгоритму ще нічого не говорить про його ефективність. Вичерпні алгоритми рідко бувають гарними у всіх відносинах.

Реалізація алгоритму

Як тільки алгоритм виражений у вигляді послідовності кроків і ми переконалися в його правильності, настає черга реалізації алгоритму, тобто написання програми .

Цей істотний крок може бути досить важким. По-перше, труднощі полягають в тім, що дуже часто окремо взятий крок алгоритму може бути виражені у формі, що важко перевести безпосередньо в конструкції мови програмування. Наприклад, один із кроків алгоритму може бути записаний у вигляді, що вимагає цілої підпрограми для його реалізації. По-друге, реалізація

може виявитися важким процесом тому, що перед тим, як ми зможемо почати писати програму, ми повинні побудувати цілу систему структур даних для подання важливих аспектів використовуваної моделі. Щоб зробити це, необхідно відповісти, наприклад, на такі питання:

- Які основні змінні?
- Яких вони типів?
- Скільки потрібно масивів, і якої розмірності?
- Чи має зміст користуватися зв'язними списками?
- Які потрібні підпрограми (можливо, уже записані в пам'яті)?
- Якою мовою програмування користуватися?

Перевірка працездатності програми

Програма написана, настає час експлуатувати її. Експлуатації програми передує налагодження. Після того як виправлена множина синтаксичних і логічних помилок , програму, нарешті, можна прогнати на простому прикладі . Але процес перевірки програми повинен містити в собі значно більше, ніж було зазначено вище. Було б перебільшенням сказати, що перевірка програми в обчислювальній математиці аналогічна експериментуванню в природничих науках. Перевірка програми може бути охарактеризована як експериментальне підтвердження того факту, що програма робить саме те, що повинна робити. Перевірка програми є також експериментальною спробою встановити границі використання алгоритму (програми).

Як вибрати вхідні дані для тестування? На це питання неможливо дати загальної відповіді. Звичайна множина всіх уведень величезна, і повна перевірка практично неможлива. Ми повинні вибрати множину уведень, які перевіряють кожну ділянку програми. Треба обов'язково досить повно перевірити випадки, які з великою ймовірністю зустрінуться в практиці.

Складання супровідної документації

Насправді етап документації не є останнім кроком у процесі повної побудови алгоритму. Зокрема, він не полягає в тім, щоб додати коментарі, коли ви закінчили все інше. Процес документації повинен переплітатися з усім процесом побудови алгоритму, і особливо з етапами розробки й реалізації.

Важко читати чужу програму в коді. Найбільш очевидний мотив для документації - дати можливість людям зрозуміти програми, які написані іншими. Звичайно, кращий спосіб - це скласти програму настільки зрозуміло, щоб вона сама себе пояснювала. Але це неможливо здійснити ні для яких програм, крім найпростіших; і програма в коді повинна бути доповнена іншими формами пояснень. Звичайно для цього використаються коментарі.

Але в дійсності це тільки надводна частина айсберга. Документація містить у собі всю інформацію й допомагає пояснити, що робиться в програмі, тобто, зокрема, блок-схеми, описи щаблів у вашій побудові алгоритму зверху - униз, доказу правильності(якщо вони є), результати тестування, детальні описи формату й вимог до уведення/виводу.

2.Завдання до лабораторних робіт

2.1 Лабораторна робота №1

Тема : Розгалужені обчислювальні процеси

Ціль роботи : Навчитися працювати з операторами розгалуження й створювати дружній інтерфейс до своїх програм.

Завдання : Розробити алгоритм і скласти програму рішення зазначеної у варіанті задачі. Тести для перевірки працездатності програми підібрати таким чином, щоб перевірялася робота всіх галузей алгоритму. Розробити документацію до програми , у якій необхідно вказати призначення програми, вимоги до вхідних даних , формат виведеної інформації й дати опис алгоритму

рішення задачі . Документація повинна відбивати всі рішення , прийняті в процесі розробки й реалізації програми.

Завдання до лабораторної роботи №1 :

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

3.

14.

15.

21

16.

17.

18.

19.

20.

21.

22.

23.

24.

25.

26.

27.

28.

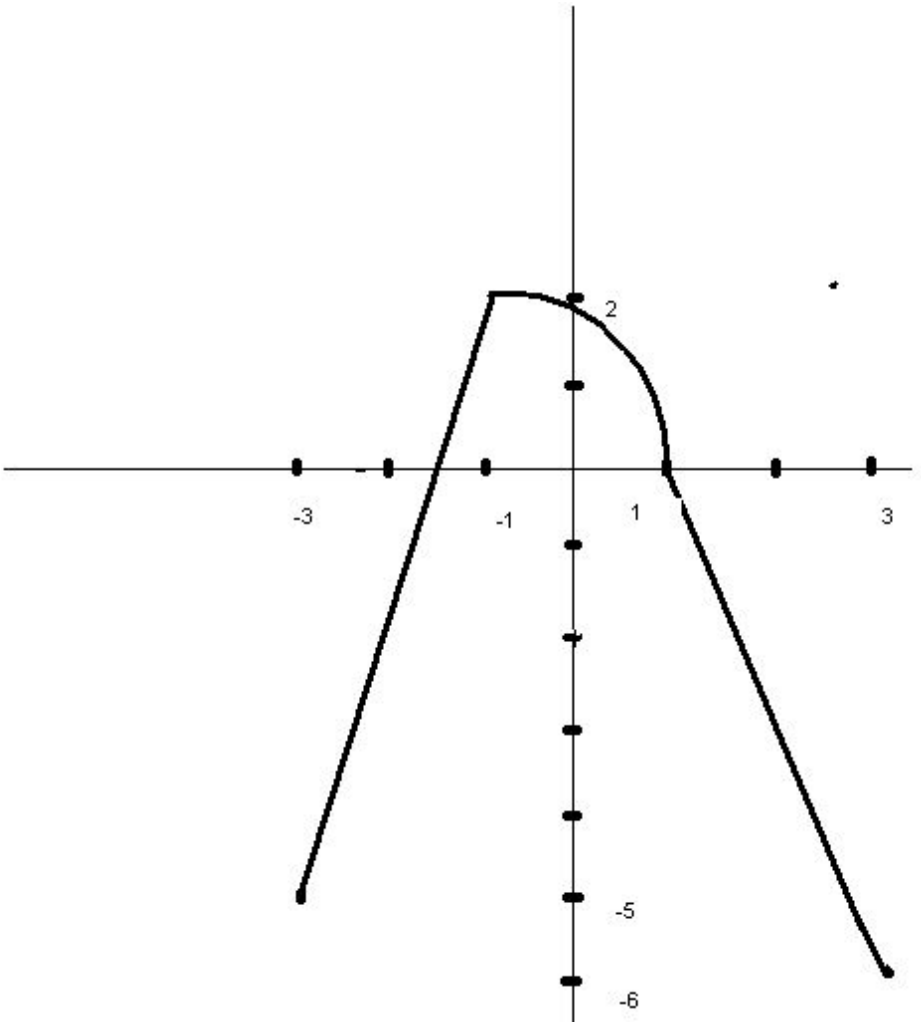
29.

30.

Приклад побудови математичної моделі .

Завдання : Скласти програму, що за уведеним значенням аргументу обчислює значення функції, заданої графічно в такий спосіб :

Пряма, що проходить через дві крапки A1(x1, y1) і A2(x2, y2) представляється рівнянням $\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$



Для першого інтервалу x1= -3 , y1= -5 , x2= -1 , y2=2 , отже

$$\frac{x+3}{-1+3}=\frac{y+5}{2+5}$$, звідси одержуємо рівняння прямої для першого інтервалу y= 3.5*x +5.5

Для третього інтервалу x1= 1 , y1= 0 , x2= 3 , y2= -6 , отже

, звідси одержуємо рівняння прямої для третього інтервалу y= -3*x +3

Окружність радіуса R із центром у крапці C(A, B) представляється рівнянням (x - A)2 + (y - B)2 = R2

Для другого інтервалу A= -1 , B=0 , R=2 , звідси одержуємо рівняння

окружності для другого інтервалу : (x +1)2 + (y – 0)2 = 22 , y2= 3 - 2*x – x2 ,отже $y = \sqrt{3 - 2 * x - x^2}$.

У такий спосіб функція задана на трьох інтервалах у такий спосіб

$$y = \begin{cases} 3.5 * x + 5.5 & ,если \quad x \le -1 \\ \sqrt{3 - 2 * x - x^2} & ,если \quad -1 < x < 1 \\ -3 * x + 3 & ,если \quad x \ge 1 \end{cases}$$

2.2 Лабораторна робота №2

Тема : Циклічні обчислювальні процеси

Ціль роботи : Навчитися працювати з операторами циклу й створювати алгоритми для циклічних процесів різних типів.

Завдання : Розробити алгоритм і скласти програму рішення зазначеної у варіанті задачі. Розробити документацію до програми , у якій необхідно вказати призначення програми, вимоги до даних , формат виведеної інформації й дати опис алгоритму рішення задачі . Документація повинна відбивати всі рішення , прийняті в процесі розробки й реалізації програми.

Теоретичний матеріал, необхідний для виконання роботи:

Цикл - це фрагмент програми, повторюваний багаторазово. Оператори, виконання яких повторюється, називаються тілом циклу. Один прохід циклу називається ітерацією. Для обслуговування циклу необхідні наступні операції:

1. початкові установки;
2. перевірка необхідності повторення циклу;
3. модифікація параметра циклу.

True

False *True*

Цикл із передумовою Цикл із постумовою

Параметром циклу називається змінна, значення якої міняється при кожному повторі циклу на певну постійну величину, називану кроком зміни параметра циклу. Початкові установки містять у собі :

1. Завдання інтервалу зміни параметра циклу;
2. Завдання кроку зміни параметра циклу;
3. Початкові значення змінних, які будуть містити суму, добуток або кількість.

Завдання до лабораторної роботи №2 :

Варіант №1

$$F = \begin{cases} z, \text{если } z > 0 \\ 0, \text{если } -1 < z < 0 \\ z^2, \text{если } z < -1 \end{cases}$$

Обчислити: y=F+0.38, $z=x^3+5x$,
Визначити кількість і суму y > F

Вихідні дані: $-1 \le x \le 5$ =0.6

Варіант №2

$$N=\begin{cases}10\ ,если\ x<0\\5\ ,если\ x=0\\20\ ,если\ x>0\end{cases}\qquad y=\begin{cases}e^{\cos(x)},\ если\quad x<b^3\\(x^2-b),\ если\quad x=b^3\\\cos(x)-b^3,\ если\ a^2x>b^3\end{cases}$$

Обчислити:

Вихідні дані: b=0,8; $-2\le x\le 5$ Δx =0.9

Варіант №3

Обчислити: $y=\begin{cases}2x^2+1, & \text{если } x\ge 0\\ x^2-1, & \text{если } x<0\end{cases}$ $F = \frac{1}{N}\sum_{i=1}^{N} y$,

де N - кількість обчислень y.

Вихідні дані: $-3\le x\le 3$ Δx =0,5

Варіант №4

Обчислити : $z=\begin{cases}x-y\ ,\ если\quad x^2-y^2<1\\ \frac{x+y}{x-y},\ если\quad x^2-y^2=1\\ \frac{2x-2y}{2x+2y},\ если\quad x^2-y^2>1\end{cases}$

Суму всіх ненегативних і добуток позитивних значень z.

Вихідні дані: $-10\le x\le 3$ =1,5 , y=7.5

Варіант №5

Обчислити: $$y = \begin{cases} e^{\cos(x)}, \text{ если } & x < b^3 \\ (x^2 - b), \text{ если } & x = b^3 \\ \cos(x) - b^3, \text{ если } & a^2 x > b^3 \end{cases} \qquad P = \prod_{y<0} y$$

Вихідні дані: 0,5≤ x ≤ 2 Δx =0,3 ; b=1,1.

Варіант №6

Обчислити: $$z = \begin{cases} 2x^3 + 3, & \text{если } x \geq 5 \\ 7x + 6, & \text{если } 1 \leq x < 5 \\ -2x^3, & \text{если } x < 1 \end{cases}$$

Визначити кількість значень Z, що лежать в інтервалі [-1; 1].

Вихідні дані: 0 ≤ x ≤10 Δx =1

Варіант №7

Обчислити :

Визначити суму й кількість позитивних значень F.

Вихідні дані: $-3 \le x \le 4$ $\Delta x = 0{,}5$

Варіант №8

Обчислити:

$$y = \begin{cases} x^2 + 1, & \text{если } x < 0 \\ 5, & \text{если } x = 0 \\ \sin(x), & \text{если } x > 0 \end{cases}$$

Визначити кількість y > 0, y < 0, y = 0 і більше з них.

Вихідні дані: $-12 \le x \le 12$ $\Delta x = 2$.

Варіант №9

Обчислити:

$$F = \begin{cases} \ln x, & \text{если } |x| \le 1 \\ x^3, & \text{если } |x| > 1 \end{cases}$$

Вихідні дані: $-1 \le x \le 6$ $\Delta x = 1$.

Визначити суму й кількість непозитивних значень F.

Варіант №10

Обчислити: , якщо $S \le 0{,}51$;

, якщо $S > 0{,}51$

$$R = \Sigma \frac{F + s}{2^{s}}$$

Вихідні дані: -1≤ S≤ 4 ΔS =1,2.

Варіант №11

Обчислити:

$$Y = \begin{cases} 1 + e^{-x}, & если \quad x > 1 \\ \sin(x), & если \quad x = 1 \\ \cos(x), & если \quad x < 1 \end{cases}$$

$$S = \Sigma \cos(z) \qquad z = y + \frac{x^{2}}{0.5 + x}$$

Вихідні дані: $2 \le x \le 16$ Δx =1.

Варіант №12

Обчислити:

$$y = \begin{cases} \frac{1.5b + x^{2}}{2a}, & \text{если} \quad x < 0 \\ 6.5x - b, & \text{если} \quad x \ge 0 \end{cases} \qquad z = y e^{x} + 0.52 x$$

$$P = \prod \sin(k * Z)$$

Вихідні дані: $-4 \le x \le 7$ Δx =1; a=5, b=6.8

Варіант №13

Обчислити:

$$y = \begin{cases} ax + b, \text{ если } x < 0.5 \\ (a - b)x, \text{ если } x = 0.5 \\ abx, \text{ если } x > 0.5 \end{cases} \qquad f = \frac{(N - K) \cdot b}{(N + K) \cdot a}$$

А також: N - кількість y > 0;
K – кількість y ≤ 0

Вихідні дані: 1 ≤ x ≤ 3 Δx =0.2

a=7, b=5

Варіант №14

Обчислити:

$$y = \begin{cases} e^{\sin(x)}, \text{ если } a^2 x < b^3 \\ (x^2 - a)\sin(x), \text{ если } a^2 x = b^3 \\ 4.5x + \cos(x), \text{ если } a^2 x > b^3 \end{cases} \qquad S = \sum_{y \geq 0} y$$

Вихідні дані: 0,5π ≤ x ≤ 2π Δx =0,1π ; a=0,9; b=1,1.

Варіант №15

$$y = \begin{cases} x^3, \text{ если } a \leq x \leq b \\ x^{1/3}, \text{ если } c \leq x \leq d \\ 0 \quad , \text{в остальных случаях} \end{cases}$$

Обчислити:

де N – кількість обчислень f.

Вихідні дані: 0,1≤x ≤1,3 =0,1; a=5.6; b=7.8;c=1.5; d=4.9

Варіант №16

Обчислити: $z = \begin{cases} \cos^2(\pi\, x)\,, если\, |y| > x^2 \\ 1 + x, если\, |y| \le x^2 \end{cases}$ $y=\sin^2(x)+\cos(x^2)$

$R = \sqrt{\sum_{i=1}^{N} (y - z)^2}$, де N – кількість обчислень Z.

Вихідні дані: $0 \le x \le 1{,}3$ $\Delta x = 0.1$

Варіант №17

Обчислити: $z = \begin{cases} \ln(y - i)\,, если\ y > 1.25 \\ y + i, если\ y \le 1.25 \end{cases}$ $y = \frac{\cos(i)}{2}$

$P = \prod_{i=1}^{12} (y + z)^2$

Вихідні дані: $1 \le i \le 13$ $\Delta i=1$

Варіант №18

Обчислити:

Вихідні дані: 1(i (13 ?i=1

Варіант №19

Обчислити:
$$y = \begin{cases} x \sin^3 x + |1 - x| \text{ , если } -4 \le x \le 2 \\ e^x + x^3 \text{ , если } 5 \le x \le 8 \\ 0 \text{ , в остальных случаях} \end{cases}$$

А також N, M, K - відповідно кількість позитивних, негативних, нульових значень y

Вихідні дані: -6 ≤ x ≤ 6 Δx =1

Варіант №20

Обчислити:
$$p = \begin{cases} 0.5x \cos(x) \text{ , если } q > 0.5 \\ 2x \sin(x), \text{ если } q \le 0.5 \end{cases} \qquad q = \frac{\sin(\pi x)}{2}$$

$S = \sum p$ $M = \sum q$ D=S+M

Вихідні дані: 0 ≤ x ≤ 1,2 Δx =0,1

Варіант №21

Обчислити:

$$x=\begin{cases} n^3+\cos(\dfrac{t-N}{t+N}),\ если\ \ln(t)<1.5 \\ t^2-t\ ,\ если\ \ln(t)\geq 1.5 \end{cases} \qquad z=\begin{cases} \ln(x)+1\ ,\ если\ x>0.5 \\ x\ ,\ если\ x\leq 0.5 \end{cases}$$

Кількість і добуток позитивних значень z.
Вихідні дані: 0,1 ≤ t≤ 5,2 , t=0,5 , N=7

Варіант №22

Обчислити:
$$h=\begin{cases} (1+\cos(\dfrac{a+8}{5-a}),\ если\ \ e^{-a}<0.1 \\ (\ln\ a+\sin(\ 0.1+a\ ),\ если\ \ e^{-a}<0.1 \end{cases}$$

Середнє арифметичне значень h і кількість нульових значень.

Вихідні дані: 0 ≤ a≤ 7 Δa=0.5

Варіант №23

Обчислити: кількість позитивних значень f, $f=t^3\ln(z+1)$

$$z=\begin{cases} \dfrac{1.5\ t^2}{2\ a}+b\ ,\ если\ \ 0\leq t\leq 3 \\ 6.5\ t+b\ ,\ если\ \ t>3 \\ \dfrac{\cos(\ t\ )}{a+b}\ ,\ если\ \ t\succ 0 \end{cases}$$

Вихідні дані: 0 ≤ t ≤ 7 Δt=0.5 ; a=3; b=2

Варіант №24

Обчислити: $P = \sum_{i=1}^{6} (z^2 - y^2)$, $y = \sqrt{x} + 0.2x - \frac{x^2}{x+1}$

$$z = \begin{cases} y - 0.3\frac{y^2}{y+1}, \text{если} & y > 1 \\ \cos y, \text{если} & |y| < 1 \\ 0, \text{если} & y < -1 \end{cases}$$

Вихідні дані: $0.5 \le x \le 7.5$ $\Delta x = 0.5$

Варіант №25

Обчислити: $S = \sum \ln \left|\frac{y}{x}\right|$ $y = \begin{cases} x \sin \frac{x}{4}, \text{если} & \cos x > 0.3 \\ 2 - \frac{1}{x}, \text{если} & \cos x \le 0.3 \end{cases}$

Вихідні дані:$-0.5 \le x \le 2.5$ $\Delta x = 0.5$

Варіант №26

$$y = \frac{3\sin(wt + x)}{2 + \cos(x - wt)} \qquad P = \prod_{0.1 \le y \le 0.5} y$$

Обчислити: $w = \begin{cases} \frac{\pi}{2} - 2x, \text{если} & |x| < \frac{\pi}{4} \\ \pi - 2x, \text{если} & |x| \ge \frac{\pi}{4} \end{cases}$

Вихідні дані: $-\pi/3 \le X \le \pi/3$,$\Delta X=\pi/20$, t=75.

Варіант №27

Обчислити: $y=te^{-x}+5$, визначити кількість і добуток y>t

$$t = \begin{cases} 0.7 - x \text{ , } если \;\; x \ge 0 \\ x + 0.3 \text{ , } если \;\; x < 0 \end{cases}$$

Вихідні дані: $-6 \le x \le 6$,$\Delta x=1,5$

Варіант №28

Обчислити:

$$r = \begin{cases} \cos(x) \text{ , } если \;\; q > 0.5 \\ 2x - \sin(x), \; если \;\; q \le 0.5 \end{cases} \qquad y = \frac{\cos(\pi \;\; x)}{2 - \sin \; x}$$

$$S = \prod_{r>1} r \qquad D = \sum_{y<0.4} y$$

Вихідні дані: $0 \le x \le 1,2$ Δx =0,1

Варіант №29

Обчислити: кількість позитивних значень h, $h=t^3 \ln(z+1)$

Вихідні дані: 01≤ t ≤ 7 Δt=0.85 ; a=3; b=2

Варіант №30

Обчислити: $f=y^2+x$, кількість $f<0$

$$y = \begin{cases} 1.7x^2 + b \text{ , если } x < 3 \\ 8.5x - b^3 \text{ , если } x \geq 3 \end{cases}$$

Вихідні дані: 0 ≤ x ≤ 6, Δx=0,5 ; b=7.8.

2.3 Лабораторна робота №3

Тема : Робота з одномірними масивами

Ціль роботи : Навчитися працювати з одномірними масивами й створювати проектну документацію до своїх програм.

Завдання : Розробити алгоритм і скласти програму , що обробляє одномірний масив відповідно до зазначеного у варіанті завданням. Розробити документацію до програми , у якій необхідно вказати призначення програми, вимоги до даних , формат виведеної інформації й дати опис алгоритму рішення задачі . Документація повинна відбивати всі рішення , прийняті в процесі розробки й реалізації програми.

Теоретичний матеріал, необхідний для виконання роботи:

Масиви являють собою обмежену впорядковану сукупність однотипних величин. Кожна окрема величина називається компонентом масиву. Тип компонентів може бути всяким, прийнятим у мові Паскаль, крім файлового типу. Тип компонентів називається базовим типом.

Опис типу масиву задається в такий спосіб:

<ім'я_типу>=ARRAY [<сп.інд.типів>] OF<тип>

Тут <ім'я типу > – правильний ідентифікатор; ARRAY, OF – зарезервовані слова (масив, з); <сп. інд. типів> – список з одного або декількох індексних типів, розділених комами; квадратні дужки, що обрамляють список, це вимога синтаксису; <тип> - будь-який тип Турбо Паскаля. Як індексні типи в Турбо Паскалі можна використати будь-які порядкові типи, крім LONGINT і типів-діапазонів з базовим типом LONGINT.

Наприклад:

Var Massiv: array[1..100] of Real; {одномірний масив}

Вся сукупність компонентів визначається одним ім'ям. Для позначення окремих компонентів використається конструкція, називана змінної з індексом або з індексами A[5] S[k+1] .

Для уведення або виводу масиву в список уведення або виводу міститься змінна з індексом, а оператори уведення або виводу виконуються в циклі.

Завдання до лабораторної роботи №3 :

Варіант 1

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму негативних елементів масиву, розташованих перед першим нульовим елементом;

Варіант 2

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму позитивних елементів масиву, розташованих перед першим негативним елементом;

Варіант 3

В одномірному масиві, що складається з *n* цілих елементів, обчислити добуток елементів масиву з парними номерами й записати його замість першого нульового елемента;

Варіант 4

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму елементів масиву з непарними номерами й записати його замість останнього нульового елемента;

Варіант 5

В одномірному масиві, що складається з *n* дійсних елементів, обчислити максимальний елемент масиву серед розташованих перед першим нульовим елементом;

Варіант 6

В одномірному масиві, що складається з *n* дійсних елементів, знайти мінімальний елемент масиву серед розташованих перед останнім нульовим елементом;

Варіант 7

В одномірному масиві, що складається з *n* цілих елементів, обчислити номер максимального елемента масиву серед негативних елементів;

Варіант 8

В одномірному масиві, що складається з *n* дійсних елементів, обчислити номер мінімального елемента масиву серед позитивних елементів;

Варіант 9

В одномірному масиві, що складається з *n* дійсних елементів, знайти максимальний по модулю елемент масиву й записати його замість першого нульового елемента;

Варіант 10

В одномірному масиві, що складається з *n* цілих елементів, знайти мінімальний по модулю елемент масиву й записати його замість останнього нульового елемента;

Варіант 11

В одномірному масиві, що складається з *n* дійсних елементів, обчислити номер мінімального по модулю елемента масиву серед парних елементів;

Варіант 12

В одномірному масиві, що складається з *n* дійсних елементів, обчислити номер максимального по модулю елемента масиву серед негативних елементів;

Варіант 13

В одномірному масиві, що складається з *n* дійсних елементів, обчислити кількість і суму непарних елементів масиву, що лежать у діапазоні від *A* до *B;*

Варіант 14

В одномірному масиві, що складається з *n* дійсних елементів, обчислити кількість елементів масиву, рівних нулю, і видалити їх;

Варіант 15

В одномірному масиві, що складається з *n* дійсних елементів, обчислити кількість елементів масиву, більших С и знайти номер першого такого елемента;

Варіант 16

В одномірному масиві, що складається з *n* дійсних елементів, обчислити кількість негативних елементів масиву й видалити їх;

Варіант 17

В одномірному масиві, що складається з *n* цілих елементів, обчислити кількість позитивних елементів масиву розташованих після першого нульового елемента;

Варіант 18

В одномірному масиві, що складається з *n* дійсних елементів, обчислити кількість елементів масиву, менших *C* и знайти номер останнього такого елемента*;*

Варіант 19

В одномірному масиві, що складається з *n* дійсних елементів, обчислити добуток негативних елементів і записати його замість першого нульового елемента ;

Варіант 20

В одномірному масиві, що складається з *n* дійсних елементів, обчислити добуток позитивних елементів масиву й записати його замість останнього нульового елемента;

Варіант 21

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму елементів масиву, розташованих перед першим нульовим елементом.

Варіант 22

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму елементів масиву, розташованих після останнього нульового елемента.

Варіант 23

В одномірному масиві, що складається з *n* дійсних елементів, обчислити добуток ненульових елементів масиву, розташованих після першого нульового елемента.

Варіант 24

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму модулів елементів масиву, розташованих після першого елемента, рівного нулю.

Варіант 25

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму модулів елементів масиву, розташованих після першого елемента, рівного нулю.

Варіант 26

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму позитивних елементів масиву й записати її замість першого нульового елемента.

Варіант 27

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму позитивних елементів масиву, розташованих перед першим негативним елементом.

Варіант 28

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму елементів масиву, розташованих після останнього негативного елемента.

Варіант 29

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму елементів масиву, розташованих після першого негативного елемента.

Варіант 30

В одномірному масиві, що складається з *n* дійсних елементів, обчислити суму модулів елементів масиву, розташованих після першого позитивного елемента.

2.4 Лабораторна робота №4

Тема : Робота із двовимірними масивами

Ціль роботи : Навчитися розробляти алгоритми обробки багатомірних масивів і створювати проектну документацію до своїх програм.

Завдання : Розробити алгоритм і скласти програму , що обробляє двовимірний масив відповідно до зазначеного у варіанті завдання. Розробити документацію до програми , у якій необхідно вказати призначення програми, вимоги до даних , формат виведеної інформації й дати опис алгоритму рішення задачі . Документація повинна відбивати всі рішення , прийняті в процесі розробки й реалізації програми.

Теоретичний матеріал, необхідний для виконання роботи:

Опис двовимірного масиву задається в такий спосіб:

Matrix: array[1..10,1..7] of Real; {двовимірний масив}

Двовимірні масиви зберігаються в пам'яті ЕОМ по рядках. Тут описаний двовимірний масив з ім'ям Matrix , що складається з 10 рядків й 7 стовпців. Вся сукупність компонентів визначається одним ім'ям. Для позначення окремих компонентів використається конструкція, називана змінною з індексами (НАПРИКЛАД B[3,5]). При роботі із двовимірним масивом перший індекс компоненти визначає номер рядка, другий - номер стовпця.

Завдання до лабораторної роботи №4 :

Варіант 1

Дана прямокутна матриця, що містить цілі числа. Визначити кількість рядків, що не містять жодного нульового елемента.

Варіант 2

Дана прямокутна матриця, що містить цілі числа. Визначити кількість стовпців, що не містять жодного нульового елемента .

Варіант 3

Дана прямокутна матриця, що містить цілі числа. Визначити кількість стовпців, що містять хоча б один нульовий елемент .

Варіант 4

Дана квадратна матриця, що містить цілі числа. Визначити загальний добуток елементів у тих рядках, які не містять негативних елементів .

Варіант 5

Дана квадратна матриця, що містить цілі числа. Визначити загальну суму елементів у тих стовпцях, які не містять негативних елементів .

Варіант 6

Дана прямокутна матриця, що містить цілі числа. Визначити загальну суму елементів у тих рядках, які містять хоча б один негативний елемент .

Варіант 7

Дана прямокутна матриця, що містить цілі числа. Знайти загальну суму елементів у тих рядках, які містять хоча б один негативний елемент.

Варіант 8

Дана прямокутна матриця, що містить цілі числа.. Знайти загальну суму елементів у тих стовпцях, які містять хоча б один негативний елемент .

Варіант 9

Дана квадратна матриця, що містить цілі числа. Знайти суму модулів елементів, розташованих нижче головної діагоналі й замінити нею всі діагональні елементи.

Варіант 10

Дана квадратна матриця, що містить цілі числа. Знайти суму модулів елементів, розташованих вище головної діагоналі й замінити нею всі діагональні елементи.

Варіант 11

Дана прямокутна матриця, що містить цілі числа. Знайти кількість рядків, середня арифметичне елементів яких менше заданої величини.

Варіант 12

Дана прямокутна матриця, що містить цілі числа. Знайти номер першої з рядків, що містять хоча б один позитивний елемент.

Варіант 13

Дана прямокутна матриця, що містить цілі числа. Знайти кількість рядків, середня геометричне елементів яких більше заданої величини.

Варіант 14

Дана прямокутна матриця, що містить цілі числа. Визначити добуток елементів у тих стовпцях, які містять хоча б один негативний елемент .

Варіант 15

Дана прямокутна матриця, що містить цілі числа. Визначити номер першого зі стовпців, що містять хоча б один нульовий елемент .

Варіант 16

Дана прямокутна матриця, що містить цілі числа. Знайти номер першого зі стовпців, що не містять жодного негативного елемента .

Варіант 17

Дана прямокутна матриця, що містить цілі числа. Знайти номер першого з рядків, що не містять жодного позитивного елемента .

Варіант 18

Дана прямокутна матриця, що містить цілі числа. Визначити кількість рядків, що містять хоча б один нульовий елемент .

Варіант 19

Дана квадратна матриця, що містить цілі числа. Визначити суму елементів у тих рядках, які не містять позитивних елементів .

Варіант 20

Дана прямокутна матриця, що містить цілі числа. Визначити кількість негативних елементів у тих рядках, які містять хоча б один нульовий елемент .

Варіант 21

Дана прямокутна матриця, що містить цілі числа. Визначити кількість рядків, що містять два нульових елементи.

Варіант 22

Дана прямокутна матриця, що містить цілі числа. Визначити кількість стовпців, що містять два позитивних елементи.

Варіант 23

Дана прямокутна матриця, що містить цілі числа. Визначити кількість стовпців, що містять тільки нульові елементи .

Варіант 24

Дана прямокутна матриця, що містить цілі числа. Знайти суму модулів елементів, розташованих нижче головної діагоналі й замінити нею всі діагональні елементи.

Варіант 25

Дана квадратна матриця, що містить цілі числа. Визначити суму елементів у тих стовпцях, які містять три негативних елементи .

Варіант 26

Дана прямокутна матриця, що містить цілі числа. Визначити суму елементів у тих рядках, які складаються тільки з негативних елементів .

Варіант 27

Дана прямокутна матриця, що містить цілі числа. Знайти загальний добуток елементів у тих рядках, які складаються тільки з позитивних елементів.

Варіант 28

Дана прямокутна матриця, що містить цілі числа.. Знайти загальну суму елементів у тих стовпцях, у яких перший елемент негативний .

Варіант 29

Дана прямокутна матриця, що містить цілі числа. Знайти добуток елементів у тих рядках, у яких перший елемент позитивний .

Варіант 30

Дана квадратна матриця, що містить цілі числа. Визначити добуток елементів у тих рядках, які не містять нульових елементів .

2.5 Лабораторна робота №5

Тема : Робота з файловими змінними

Ціль роботи : Навчитися працювати з файловими змінними й створювати проектну документацію до своїх програм.

Завдання : Розробити алгоритм і скласти програму , що обробляє вихідний файл відповідно до зазначеного у варіанті завдання. Розробити документацію до програми , у якій необхідно вказати призначення програми, вимоги до даних , формат виведеної інформації й дати опис алгоритму рішення задачі . Документація повинна відбивати всі рішення , прийняті в процесі розробки й реалізації програми й докладно описувати формат вихідного файлу .

Теоретичний матеріал, необхідний для виконання роботи:

Під файлом розуміється іменована область зовнішньої пам'яті ПК. Любий файл має три характерні риси. По-перше, у нього є ім'я, що дає можливість програмі працювати одночасно з декількома файлами. По-друге, він містить компоненти одного типу. Типом компонентів може бути будь-який тип Турбо Паскаля, крім файлів. По-третє, довжина знову створюваного файлу ніяк не оговорюється при його оголошенні й обмежується тільки ємністю пристроїв зовнішньої пам'яті.

Файли стають доступні програмі тільки після виконання особливої процедури відкриття файлу . Ця процедура полягає у зв'язуванні раніше оголошеної файлової змінної з ім'ям існуючого або знову створюваного файлу, а також у вказівці напрямку обміну інформацією: читання з файлу або запис у нього.

Файлова змінна зв'язується з ім'ям файлу в результаті звертання до стандартної процедури ASSIGN (<ф.п.>, <ім'я файлу>); .

Тут <ф.п. > - файлова змінна (правильний ідентифікатор, оголошений у

програмі як змінна файлового типу);

<ім'я файлу> - текстовий вираз, що містить ім'я файлу.

Для читання файл ініціюється за допомогою стандартної процедури RESET (<ф.п.>); Тут <ф.п.> - файлова змінна, зв'язана раніше процедурою ASSIGN із уже існуючим файлом. При виконанні цієї процедури спеціальна змінна-покажчик, пов'язана із цим файлом, буде вказувати на початок файлу, тобто на компонент із порядковим номером 0.

Стандартна процедура REWRITE (<ф.п.>) ініціює запис інформації у файл. Процедурою REWRITE не можна ініціювати запис інформації в раніше існуючий дисковий файл: при виконанні цієї процедури старий файл знищується і ніяких повідомлень про це в програму не передається. Новий файл підготовляється до прийому інформації і його покажчик приймає значення 0.

Стандартна процедура APPEND (<ф.п. >) ініціює запис у раніше існуючий текстовий файл для його розширення, при цьому покажчик файлу встановлюється після останнього компонента. Процедура APPEND застосовна тільки до текстових файлів, тобто їх файлова змінна повинна мати тип TEXT.

Завдання до лабораторної роботи №5 :

Варіант 1

Дано файл f, компоненти якого є дійсними числами. Знайти суму компонентів файлу f; добуток компонентів файлу f; суму квадратів компонент файлу f і записати їх в інший файл.

Варіант 2

Дано файл f, компоненти якого є дійсними числами. Знайти модуль суми й квадрат добутку компонент файлу f, останній компонент файлу й записати їх в інший файл.

Варіант 3

Дано файл f, компоненти якого є дійсними числами. Знайти найбільше зі значень компонентів; найменше зі значень компонентів з парними номерами й записати їх в інший файл.

Варіант 4

Дано файл f, компоненти якого є дійсними числами. Знайти найбільше зі значень модулів компонент із непарними номерами; суму найбільшого й найменшого зі значень компонентів і записати їх в інший файл.

Варіант 5

Дано файл f, компоненти якого є дійсними числами. Знайти різницю першого й останнього компонентів файлу; кількість парних чисел серед компонентів і записати їх в інший файл.

Варіант 6

Дано файл f, компоненти якого є дійсними числами. Знайти кількість подвоєних непарних чисел серед компонентів; кількість таких компонентів і записати їх в інший файл.

Варіант 7

Дано файл f, компоненти якого є цілими числами. Одержати у файлі g усе компоненти файлу f , що є точними квадратами.

Варіант 8

Дано символьні файли f й q. Переписати зі збереженням порядку проходження компоненти файлу f і компоненти файлу q у файл q.

Варіант 9

Дано файли f1, f2, компоненти яких є дійсними числами. Записати у файл q парні компоненти цих файлів.

Варіант 10

Дано файл f, компоненти якого є дійсними числами. Знайти найбільше й найменше зі значень компонентів з непарними номерами і їх середнє

геометричне й записати їх в інший файл.

Варіант 11

Дано файл f, компоненти якого є цілими числами. Одержати у файлі g усе компоненти файлу f , що є парними числами.

Варіант 12

Дано файл f, компоненти якого є цілими числами. Переписати компоненти файлу f у файл g так, щоб у файлі g не було двох сусідніх чисел з одним знаком.

Варіант 13

Дано файл f, компоненти якого є дійсними числами. Знайти найбільше зі значень компонентів; найменше зі значень компонентів з непарними номерами й записати їх в інший файл.

Варіант 14

Дано файл f, компоненти якого є дійсними числами. Знайти найбільше й найменше зі значень компонентів з парними номерами і їх середнє арифметичне й записати їх в інший файл.

Варіант 15

Дано файл f, компоненти якого є цілими числами. Переписати компоненти файлу f у файл g так, щоб у файлі g не було двох сусідніх нулів (викинувши розташовані поруч нульові елементи).

Варіант 16

Дано файл f, компоненти якого є дійсними числами. Знайти суму й квадрат добутку позитивних компонентів файлу f; останній негативний компонент файлу й записати їх в інший файл.

Варіант 17

Дано файл f, компоненти якого є дійсними числами. Знайти суму й добуток компонентів файлу f і їх середнє арифметичне й записати їх в інший файл.

Варіант 18

Дано файл f, компоненти якого є цілими числами. Одержати файл g, утворений з файлу f виключенням кожного третього запису.

Варіант 19

Дано файл f, компоненти якого є цілими числами. Одержати у файлі g усе компоненти файлу f , що діляться на 3 і не діляться на 7.

Варіант 20

Дано файл f, компоненти якого є цілими числами. Переписати компоненти файлу f у файл g так, щоб у файлі g спочатку йшли позитивні, потім негативні числа, нульові компоненти не переписувати.

Варіант 21

Дано файл f, компоненти якого є цілими числами. Переписати компоненти файлу f у файл g так, щоб у файлі g числа розташовувалися в наступному порядку : два позитивних, два негативних, два позитивних, два негативних і т.д. , нульові компоненти не переписувати.

Варіант 22

Дано файл f, компоненти якого є цілими числами. Переписати компоненти файлу f у файл g так, щоб у файлі g не було двох сусідніх чисел , що відрізняються друг від друга на 7.

Варіант 23

Дано файл f, компоненти якого є дійсними числами. Знайти суму негативних компонентів файлу f; добуток позитивних компонентів файлу f; суму квадратів компонент файлу f і записати їх в інший файл.

Варіант 24

Дано файл f, компоненти якого є дійсними числами. Знайти середнє геометричне першої й останньої компонент файлу; кількість ненегативних чисел серед компонентів і записати їх в інший файл.

Варіант 25

Дано файл f, компоненти якого є цілими числами. Знайти різницю мінімального й максимального компонентів файлу; кількість позитивних

чисел серед компонентів і записати їх в інший файл.

Варіант 26

Дано файл f, компоненти якого є цілими числами. Записати у файл g всі парні числа файлу f у зворотному порядку.

Варіант 27

Дано файл f, компоненти якого є дійсними числами. Знайти квадрат добутку негативних компонентів файлу f і записати його в інший файл .

Варіант 28

Дано файл f, компоненти якого є дійсними числами. Знайти найбільше зі значень компонентів і найменше зі значень компонентів з непарними номерами й записати їх в інший файл.

Варіант 29

Дано файл f, компоненти якого є дійсними числами. Знайти найбільше зі значень модулів компонент із парними номерами; добуток найбільшого й найменшого зі значень компонентів і записати їх в інший файл.

Варіант 30

Дано файл f, компоненти якого є дійсними числами. Знайти середнє арифметичне першої й останньої компонент файлу; кількість непарних чисел серед компонентів і записати їх в інший файл.

2.6 Лабораторна робота №6

Тема : Робота з підпрограмами

Ціль роботи : Навчитися структурувати програму й створювати

проектну документацію до всіх структурних частин програми.

Завдання : Розробити алгоритми заданої у варіанті підпрограми й основної програми , що використає цю підпрограму, і скласти програму. Розробити документацію до всіх структурних частин програми , у якій необхідно вказати призначення , вимоги до даних , формат виведеної інформації й дати опис використовуваних параметрів . Документація повинна відбивати всі рішення , прийняті в процесі розробки й реалізації програми .

Теоретичний матеріал, необхідний для виконання роботи:

Процедури й функції являють собою важливий інструмент Турбо Паскаля, що дозволяє писати добре структуровані програми. У структурованих програмах звичайно легко простежується основний алгоритм, їх неважко зрозуміти , вони простіше в налагодженні й менш чутливі до помилок програмування. Всі ці властивості є наслідком важливої особливості підпрограм, кожна з яких являє собою багато в чому самостійний фрагмент програми, пов'язаний з основною програмою лише за допомогою декількох параметрів. Самостійність підпрограм дозволяє локалізувати в них всі деталі програмної реалізації тієї або іншої алгоритмічної дії й тому зміна цих деталей, наприклад, у процесі налагодження, звичайно не приводить до змін основної програми.

Писати складні програми як щось єдине ціле, без розділення на відносно самостійні фрагменти, тобто без структурування, просто неможливо. Практично у всіх мовах програмування є засоби структурування. Мови, у яких передбачені такі механізми, називаються процедурно- орієнтованими. До їхнього числа належить і Турбо Паскаль.

Процедурою в Турбо Паскалі називається особливим чином оформлений фрагмент програми, що має власне ім'я. Згадування цього імені в тексті програми приводить до активізації процедури й називається її викликом. Відразу після активізації процедури починають виконуватися вхідні в неї оператори, після виконання останнього з них керування повертається в основну

програму й виконується оператор, що розташований безпосередньо за оператором виклику процедури.

Для обміну інформацією між основною програмою й процедурою використається один або кілька параметрів виклику. Процедури можуть мати й інший механізм обміну даними із основною програмою, так що параметри виклику можуть і не використатися. Якщо параметри є, то вони перелічуються в круглих дужках за ім'ям процедури й разом з ним утворюють оператор виклику процедури.

Функція відрізняється від процедури тим, що результат її роботи повертається у вигляді значення цієї функції, і, отже, виклик функції може використатися поряд з іншими операндами у виразах.

У Турбо Паскалі є багато стандартних процедур і функцій. Наявність багатої бібліотеки таких програмних заготівель істотно полегшує розробку прикладних програм. Однак іноді програмістові доводиться розробляти свої, нестандартні процедури й функції.

Нестандартні процедури й функції необхідно описати, щоб компілятор міг встановити зв'язок між оператором виклику й тих дій, які передбачені в підпрограмі. Опис підпрограми міститься в розділі описів і зовні виглядає як програма, але замість заголовка програми фігурує заголовок процедури (функції).

Програмуючи з використанням підпрограм користувач працює з універсальним методом конструювання складних програм, що отримав назву спадне програмування. Відповідно до цього методу створення програми починається «зверху», тобто з розробки самого головного, генерального алгоритму. На верхньому рівні звичайно ще не ясні деталі реалізації тієї або іншої частини програми, тому ці частини варто замінити тимчасовими заглушками. Бажано, щоб тимчасовий варіант програми був синтаксично правильним, тоді можна його відкомпілювати й переконатися у відсутності в ньому синтаксичних помилок. Такий прогін дасть певну впевненість перед

розробкою й реалізацією алгоритмів нижнього рівня, тобто перед заміною заглушок реально працюючими процедурами. Якщо реалізований у заглушці алгоритм досить складний, його знову структурують, виділяючи головний алгоритм і застосовуючи нові заглушки, і т.д. Процес триває «униз» доти, поки не буде створений повністю працездатний варіант програми.

При оформленні програм необхідно прагнути використовувати гарний стиль написання програм, тобто таку їхню форму, що дає найбільш повне подання про структуру програми в цілому і її окремих частин. Не існує якого-небудь стандарту, що визначає гарний стиль програми. Звичайно це інтуїтивне поняття включає спосіб розташування операторів й описів по рядках (не рекомендується розміщати більше одного оператора на кожному рядку), а також виділення відступами тіла складених й умовних операторів.

Опис підпрограми складається із заголовка й тіла підпрограми.

Заголовок процедури має вигляд: PROCEDURE <ім'я> [(<сп. ф.п. >)] ;

Заголовок функції: FUNCTION <ім'я> [(<сп.ф.п.>)] : <тип>;

Тут <ім'я> - ім'я підпрограми (правильний ідентифікатор);

<сп.ф. п. > - список формальних параметрів;

<тип> - тип результату, що повертається функцією.

Список формальних параметрів необов'язковий і може бути відсутнім. Якщо ж він є, то в ньому повинні бути перелічені імена формальних параметрів й їхні типи. Оператори тіла підпрограми розглядають список формальних параметрів як своєрідне розширення розділу описів: всі змінні із цього списку можуть використатися в будь-яких виразах усередині підпрограми. Механізм заміни формальних параметрів на фактичні дозволяє потрібним образом настроїти алгоритм, реалізований у підпрограмі. Турбо Паскаль стежить за тим, щоб кількість і тип формальних параметрів строго відповідали кількості й типам фактичних параметрів у момент звертання до підпрограми. Зміст використовуваних фактичних параметрів залежить від того, у якому порядку вони перераховані при виклику підпрограми. Користувач повинен сам стежити

за правильним порядком перерахування фактичних параметрів при звертанні до підпрограми. Любий з формальних параметрів підпрограми може бути або параметром-значенням, або параметром-змінною, або параметром-константою. Визначення формального параметра у той або інший спосіб істотно тільки для основної програми: якщо формальний параметр оголошений як параметр-змінна, то при виклику підпрограми йому повинен відповідати фактичний параметр у вигляді змінної потрібного типу; якщо формальний параметр оголошений як параметр-значення або параметр-константа, то при виклику йому може відповідати довільний вираз. Контроль за дотриманням цього правила здійснюється компілятором Турбо Паскаля.

Якщо параметр визначений як параметр-значення, то перед викликом підпрограми це значення обчислюється, отриманий результат копіюється в тимчасову пам'ять і передається підпрограмі. Важливо врахувати, що навіть якщо фактичний параметр представлений найпростішим виразом у вигляді змінної або константи, однаково підпрограмі буде передана лише копія змінної (константи). Будь-які можливі зміни в підпрограмі параметра-значення ніяк не сприймаються основною програмою, тому що в цьому випадку змінюється копія фактичного параметра.

Якщо параметр визначений як параметр-змінна, то при виклику підпрограми передається сама змінна, а не її копія (фактично в цьому випадку підпрограмі передається адреса змінної). Зміна параметра-змінної приводить до зміни самого фактичного параметра в основній програмі.

У випадку параметра-константи в підпрограму також передається адреса області пам'яті, у якій розташовується змінна або обчислене значення. Однак компілятор блокує будь-які присвоювання параметру-константі нового значення в тілі підпрограми.

Отже, параметри-змінні використаються як засіб зв'язку алгоритму, реалізованого в підпрограмі, із зовнішнім миром: за допомогою цих параметрів підпрограма може передавати результати своєї роботи основній програмі. У

розпорядженні програміста завжди є й інший спосіб передачі результатів - через глобальні змінні. Однак зловживання глобальними зв'язками робить програму , як правило, заплутаною, важкою в розумінні й складною в налагодженні. Відповідно до вимог гарного стилю програмування рекомендується там, де це можливо, використати передачу результатів через фактичні параметри.

Опис всіх формальних параметрів як параметрів-змінних небажано по двох причинах. По-перше, це виключає можливість виклику підпрограми з фактичними параметрами у вигляді виразів, що робить програму менш компактною. По-друге, і головне, у підпрограмі можливо випадкове використання формального параметра, наприклад, для тимчасового зберігання проміжного результату, тобто завжди існує небезпека ненавмисно зіпсувати фактичну змінну. От чому перемінними- змінними варто оголошувати тільки ті, через які підпрограма в дійсності передає результати основній програмі. Чим менше параметрів оголошено параметрами-змінними й чим менше в підпрограмі використається глобальних змінних, тим менше небезпека одержання непередбачуваних програмістом побічних ефектів.

Існує ще одна обставина, яку варто враховувати при виборі виду формальних параметрів. При оголошенні параметра-значення здійснюється копіювання фактичного параметра в тимчасову пам'ять. Якщо цим параметром буде масив великої розмірності, то істотні витрати часу й пам'яті на копіювання при багаторазових звертаннях до підпрограми можна мінімізувати, оголосивши цей параметр параметром-константою. Параметр-константа не копіюється в тимчасову область пам'яті, що скорочує витрати часу на виклик підпрограми, однак будь-які його зміни в тілі підпрограми неможливі - за цим строго стежить компілятор.

Оголошення змінних у списку формальних параметрів підпрограми відрізняється від оголошення їх у розділі опису змінних : типом будь-якого параметра в списку формальних параметрів може бути тільки стандартний або

раніше оголошений тип. Якщо в підпрограму передається весь масив, то варто спочатку описати його тип.

Завдання до лабораторної роботи №6 :

Варіант 1

Дано натуральне число n, масив $a_1,...,a_n.$ Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які є повними квадратами й знайти суму таких елементів.

Варіант 2

Два трикутники задані координатами своїх вершин . Скласти процедуру, що дозволяє знайти площі трикутників, і визначити, яка з них більше.

Варіант 3

Скласти процедуру, що заміняє у вихідному масиві всі парні елементи їхніми квадратами. Заміна повинна виконуватися, починаючи із заданої позиції .

Варіант 4

Дано натуральне число n, масив $a_1,..., a_n.$ Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які є ступенями 3 і знайти кількість таких елементів.

Варіант 5

Два трикутники задані координатами своїх вершин . Скласти процедуру, що дозволяє визначити, чи лежать вони в заданій координатній чверті й знайти площі трикутників.

Варіант 6

Два трикутники задані координатами своїх вершин . Скласти процедуру, що дозволяє визначити, чи лежать вони в різних координатних чвертях й знайти площі трикутників.

Варіант 7

Дано натуральне число n, масив $a_1,...,a_n$. Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які є ступенями 4 і знайти добуток таких елементів.

Варіант 8

Дві окружності задані координатами своїх центрів і радіусами . Скласти процедуру, що дозволяє визначити, чи лежать вони в заданій координатній чверті й знайти їх площі.

Варіант 9

Скласти процедуру, що заміняє у вихідному масиві всі елементи, рівні мінімальному, нулями. Заміна повинна виконуватися, починаючи із заданої позиції .

Варіант 10

Дано натуральне число n, масив $a_1,...,a_n$. Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які є парними числами й знайти суму таких елементів.

Варіант 11

Два квадрати задані координатами своїх вершин . Скласти процедуру, що дозволяє визначити, чи лежать вони в зазначеній координатній чверті й знайти площі трикутників.

Варіант 12

Дано натуральне число n, масив Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які є непарними числами й знайти кількість таких елементів.

Варіант 13

Скласти процедуру, що заміняє у вихідному масиві всі парні числа числом π. Заміна повинна виконуватися, починаючи із заданої позиції .

Варіант 14

Дано натуральне число n, масив $a_1,...,a_n$. Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які є позитивними числами й знайти добуток таких елементів.

Варіант 15

Два прямокутники задані координатами своїх вершин . Скласти процедуру, що дозволяє знайти площі прямокутників, і визначити, яка з них більше.

Варіант 16

Скласти процедуру, що заміняє у вихідному масиві всі елементи, рівні максимальному, одиницями. Заміна повинна виконуватися, починаючи із заданої позиції .

Варіант 17

Дано натуральне число n, масив $a_1,...,a_n$. Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які є негативними числами й знайти суму таких елементів.

Варіант 18

Два прямокутники задані координатами своїх вершин . Скласти процедуру, що дозволяє визначити, у якій координатній чверті вони лежать і знайти площі прямокутників.

Варіант 19

Дано натуральне число n, масив Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву , індекси яких є непарними числами й знайти кількість таких елементів.

Варіант 20

Три прямокутники задані координатами своїх вершин . Скласти

процедуру, що дозволяє визначити площі прямокутників, і знайти більшу з них.

Варіант 21

Дано натуральне число n, масив $a_1,...,a_n.$ Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву , індекси яких є парними числами й знайти добуток таких елементів.

Варіант 22

Дано натуральне число n, масив $a_1,...,a_n.$ Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які належать інтервалу [-10 ; 5] і знайти середнє значення таких елементів.

Варіант 23

Скласти процедуру, що заміняє у вихідному масиві всі непарні елементи їхніми квадратами. Заміна повинна виконуватися, починаючи із заданої позиції .

Варіант 24

Дано натуральне число n, масив $a_1,...,a_n.$ Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які менше їх середнього геометричного і знайти кількість таких елементів.

Варіант 25

Скласти процедуру, що заміняє у вихідному масиві всі елементи, рівні середньому арифметичному, нулями. Заміна повинна виконуватися, починаючи із заданої позиції .

Варіант 26

Скласти процедуру, що заміняє у вихідному масиві всі одиниці нулями й всі нулі одиницями. Заміна повинна виконуватися після першого нульового елемента.

Варіант 27

Дано натуральне число n, масив $a_1,...,a_n$. Скласти процедуру, що дозволяє формувати масив X з тих елементів вихідного масиву, які належать інтервалу [0,5] і знайти суму й кількість таких елементів.

Варіант 28

Скласти процедуру, що заміняє у вихідному масиві всі позитивні елементи одиницями. Заміна повинна виконуватися після останнього нульового елемента.

Варіант 29

Три трикутники задані координатами своїх вершин . Скласти процедуру, що дозволяє визначити їхньої площі, і знайти найменшу з них.

Варіант 30

Скласти процедуру, що заміняє у вихідному масиві всі негативні елементи нулями. Заміна повинна виконуватися після мінімального елемента масиву.

3. Методичні вказівки по роботі в середовищі Турбо Паскаля

Функціональні клавіші

Функціональні клавіші використаються для керування середовищем Турбо Паскаля. Вони позначаються F1, F2, ..., F12 і розташовуються в самому верхньому ряді клавіатури. З кожної із цих клавіш зв'язується деяка команда меню. Дію майже всіх функціональних клавіш можна модифікувати трьома особливими клавішами: Alt (від ALTernative - додатковий), Ctrl (ConTRo -керуючий) і Shift (SHIFT- зрушуючий). Ці клавіші використаються подібно клавіші тимчасової зміни регістра на друкарській машинці: потрібно нажати на одну з них і потім, не відпускаючи її, нажати функціональну клавішу. Надалі таке спільне натискання двох клавіш будемо позначати рисою. Наприклад, Alt-F3 означає, що разом із клавішею Alt необхідно натиснути клавішу F3,

Ctrl-F9 - разом з Ctrl натискається F9 і т.д.

Нижче приводяться команди, які передаються середовищу Турбо Паскаля функціональними клавішами й деякими їхніми комбінаціями із клавішами Ctrl й Alt:

Fl - звернутися за довідкою до убудованої довідкової служби (Help-допомога);

F2 - записати у дисковий файл текст, що редагувався;

F3 - прочитати текст із дискового файлу у вікно редактора;

F4 - використається в режимі налагодження : почати або продовжити виконання програми й зупинитися перед виконанням того її рядка, на якій вказує курсор;

F5 - розгорнути активне вікно на весь екран;

F6 - зробити активним наступне вікно;

F7 - використається в режимі налагодження: виконати наступний рядок програми; якщо в рядку є звертання до процедури (функції), увійти в цю процедуру й зупинитися перед виконанням першого її оператора;

F8- використається в режимі налагодження: виконати наступний рядок програми; якщо в рядку є звертання до процедури (функції), виконати її й не простежувати її роботу;

F9 - компілювати програму, але не виконувати її;

F10 - перейти до діалогового вибору режиму роботи за допомогою головного меню;

Ctrl-F9 - виконати прогін програми: компілювати програму, що перебуває в редакторі, завантажити її в оперативну пам'ять і виконати, після чого повернутися в середовище Турбо Паскаля.

Alt-F5 - перемінити вікно редактора на вікно демонстрації результатів роботи (прогону) програми.

Текстовий редактор

Текстовий редактор середовища Турбо Паскаля надає користувачеві

зручні засоби створення й редагування текстів програм. Ознакою того, що середовище перебуває в стані редагування, є наявність у вікні редактора курсору -невеликого миготливого прямокутника. Режим редагування автоматично встановлюється відразу після завантаження Турбо Паскаля. З режиму редагування можна перейти до будь-якого іншого режиму роботи Турбо Паскаля за допомогою функціональних клавіш або вибору потрібного режиму з головного меню. Якщо середовище перебуває в стані вибору з меню, курсор зникає, а в рядку меню з'являється кольоровий вказівник-прямокутник, що виділяє одне з кодових слів (опцій меню). Для переходу від стану вибору режиму з головного меню в стан редагування потрібно натиснути клавішу Esc (ESCape - вислизати, тікати), а для переходу до вибору з головного меню - F10.

Для створення тексту програми потрібно ввести цей текст за допомогою клавіатури . Після заповнення чергового рядка варто натиснути на клавішу Enter, щоб перевести курсор на наступний рядок (курсор завжди показує те місце на екрані, куди будуть поміщені чергові символи програми, що вводяться).

Вікно редактора імітує довгий і досить широкий аркуш паперу, фрагмент якого видний у вікні. Якщо курсор досяг нижнього краю, здійснюється прокручування вікна редактора : його вміст зміщається нагору на один рядок і знизу з'являється новий рядок аркуша. Якщо курсор досяг правої границі екрана, вікно починає в міру уведення символів зміщатися вправо, показуючи правий край аркуша. Розміри аркуша по горизонталі й вертикалі обмежуються тільки загальним числом символів у файлі, яких не повинне бути більше 64535, однак компілятор Турбо Паскаля сприймає рядки програми довжиною не більше 126 символів.

Вікно можна зміщати щодо аркуша за допомогою наступних клавіш:

Page Up -на сторінку нагору;

Page Down - на сторінку вниз;

Home - у початок поточного рядка;

End - у кінець поточного рядка;

Ctrl-Page Up -у початок тексту;

Ctrl-Page Down - у кінець тексту.

Клавішами переводу курсору (ці клавши позначені стрілками й розташовуються в правій частині клавіатури) його можна зміщати по екрану. При досягненні границь вікна воно зміщається на рядок або на символ.

Якщо Ви помилилися при уведенні чергового символу, його можна стерти за допомогою клавіші зі стрілкою (або написом Backspace), розташованої над клавішею Enter. Клавіша Delete стирає символ, на який у цей момент указує курсор, а команда Ctrl-Y- весь рядок, на якій розташований курсор.

Варто пам'ятати, що редактор Турбо Паскаля вставляє наприкінці кожного рядка невидимий символ-роздільник. Цей символ уставляється клавішею Enter, а стирається клавішами Backspace або Delete. За допомогою вставки/стирання роздільника можна «розрізати»/«склеїти» рядки. Щоб розрізати рядок, варто підвести курсор до потрібного місця й натиснути Enter ; щоб склеїти сусідні рядки, потрібно встановити курсор у кінець першого рядка (для цього зручно використати клавішу End) і натиснути Delete або встановить курсор у початок другого рядка (клавішею Ноті) і натиснути Backspace.

Нормальний режим роботи редактору - режим вставки, у якому кожен символ, що вводить знову, як би «розсовує» текст на екрані, зміщаючи вправо остачу рядка. Варто враховувати, що розрізування тексту й наступна вставка пропущених рядків можливі тільки в цьому режимі. Редактор може також працювати в режимі накладення нових символів на існуючий старий текст: у цьому режимі новий символ заміняє собою той символ, на який указує курсор, а остача рядка не зміщається вправо. Для переходу до режиму накладення потрібно натиснути клавішу Insert, а якщо натиснути цю клавішу ще раз, знову встановлюється режим вставки. Ознакою того, у якому режимі працює редактор, є форма курсору: у режимі вставки він схожий на миготливий символ

підкреслення, а в режимі накладення він являє собою великий миготливий прямокутник, що заслоняє символ .

І ще про одну можливість редактора. Звичайно редактор працює в режимі автовідступу. У цьому режимі кожен новий рядок починається в тій же позиції на екрані, що й попередній. Режим автовідступу підтримує гарний стиль оформлення тексту програми: відступи від лівого краю виділяють тіло умовного або складеного оператора й роблять програму більше наочною. Відмовитися від автовідступу можна командою Ctrl-O I (при натиснутої Ctrl натискається спочатку клавіша з латинською буквою O, а потім O відпускається й натискається I), повторна команда Ctrl-O I відновить режим автовідступу.

Нижче перераховані найбільше часто використовувані команди редактор Турбо Паскаля.

Змішання курсору

Page Up -на сторінку нагору;

Page Down - на сторінку вниз;

Home - у початок поточного рядка;

End - у кінець поточного рядка;

Ctrl-Page Up - у початок тексту;

Ctrl-Page Down - у кінець тексту.

Команди редагування

Backspace - стирає символ ліворуч від курсору;

Delete - стирає символ, на який показує курсор;

Ctrl-Y- стирає рядок з курсором;

Enter - вставляє новий рядок, розріже старий;

Ctrl-Q L - відновлює змінений рядок (діє, якщо курсор не залишав рядок після її зміни).

Робота із блоком

Ctrl-K B - починає виділення блоку;

Ctrl-K K- закінчує виділення блоку;

Ctrl-K Y - знищує виділений блок;

Ctrl-K C - копіює блок;

Ctrl-K V - переміщає блок на нове місце;

Ctrl-K W - записує блок у файл;

Ctrl-K R - читає блок з файлу;

Ctrl-K P - друкує блок.

Робота з файлами

Після запуску Турбо Паскаля середовище переходить у режим редагування тексту, у якому можна підготувати нову програму або виправити існуючу. Основною формою зберігання текстів програм поза середовищем є файли. Після завершення роботи з Турбо Паскалем можна зберегти текст нової програми в дисковому файлі для того, щоб використати його наступного разу. Для обміну даними між дисковими файлами й редактором середовища призначені клавіші F2 (запис у файл) і F3 (читання з файлу). Якщо Ви створюєте нову програму, то середовище ще не знає ім'я того файлу, у який Ви захочете помістити текст цієї програми, і тому вона привласнює їй стандартне ім'я NONAME00.PAS (NO NAME - немає імені). Для збереження тексту програми у файлі потрібно натиснути F2. У цей момент середовище перевірить ім'я програми й, якщо це стандартне ім'я NONAME, запитає, потрібно чи його змінювати: на екрані з'явиться невелике вікно запиту з написом

Save File as

(Зберегти у файлі з ім'ям...)

Нижче напису розташовується поле для уведення імені файлу, у якому можна написати потрібне ім'я й натиснути Enter - текст буде збережений у файлі. Якщо в імені опущене розширення, середовище привласнить файлу стандартне розширення PAS. Якщо Ви захочете завершити роботу з Турбо Паскалем, а в редакторі залишився не збережений у файлі текст, на екрані з'явиться вікно із запитом

NONAMEOO.PAS has been modified.Save?

(Файл NONAME00.PAS був змінений. Зберегти?)

У відповідь варто натиснути Y(Yes - так), якщо необхідно зберегти текст у файлі, або N(No - ні), якщо зберігати текст не потрібно.

Прогін і налагодження програми

Після підготовки тексту програми можна спробувати виконати її, тобто відкомпілювати програму, зв'язати її (якщо це необхідно) з бібліотекою стандартних процедур і функцій, завантажити в оперативну пам'ять і передати їй керування. Вся ця послідовність дій називається прогоном програми й реалізується командою Ctrl-F9.

Якщо в програмі немає синтаксичних помилок, то всі дії виконуються послідовно одне за іншим, при цьому в невеликому вікні повідомляється про кількість відкомпільованих рядків й об'єм доступної оперативної пам'яті. Перед передачею керування завантаженій програмі середовище очищає екран (точніше, виводить на екран вікно прогону програми), а після завершення роботи програми середовище знову бере керування комп'ютером на себе й відновлює на екрані вікно редактора.

Якщо на якому-небудь етапі середовище виявить помилку, воно припиняє подальші дії, відновлює вікно редактора й поміщає курсор на той рядок програми, при компіляції або виконанні якої виявлена помилка. При цьому у верхньому рядку редактора з'являється діагностичне повідомлення про причину помилки. Все це дозволяє дуже швидко налагодити програму, тобто усунути в ній синтаксичні помилки й переконатися в правильності її роботи. Якщо помилка виникла на етапі прогону програми, простої вказівки того місця, де вона виявлена, буде недостатньо, тому що помилка може бути наслідком неправильної підготовки даних у попередніх операторах програми. Наприклад, якщо помилка виникла в результаті добування квадратного кореня з негативного числа, буде зазначений оператор, у якому здійснюється добування

кореня, хоча ясно, що першопричину помилки треба шукати десь раніше, там, де відповідної змінної привласнюється негативне значення. У таких ситуаціях звичайно прибігають до покрокового виконання програми за допомогою команд, пов'язаних із клавішами F4, F7 й F8. Поки ще не накопичений достатній досвід налагодження, можна скористатися однією клавішею F7, після натискання на яку середовище здійснить компіляцію, компонування (зв'язок з бібліотекою стандартних процедур і функцій) і завантаження програми, а потім зупинить прогін перед виконанням першого оператора. Рядок програми, що містить цей оператор, буде виділена на екрані покажчиком (кольорами). Тепер кожне нове натискання F7 буде викликати виконання всіх операцій, запрограмованих у поточному рядку, і зсув покажчика до наступного рядка програми. У підозрілому місці програми можна переглянути поточне значення змінної або вираження. Для цього потрібно встановити курсор у те місце рядки, де перебуває ім'я змінної, що Вас цікавить, і натиснути Ctrl-F4. На екрані з'явиться діалогове вікно, що складається із трьох полів (у верхнім полі буде стояти ім'я змінної, два інших поля будуть порожніми). Натисніть Enter, щоб у середнім полі одержати поточне значення змінної. Якщо перед натисканням Ctrl-F4 курсор стояв на порожній ділянці рядка або вказував на ім'я інший змінної, верхнє поле діалогового вікна також буде порожнім або містити ім'я цієї інший змінної. У цьому випадку варто ввести за допомогою клавіатури ім'я потрібної змінної й натиснути Enter. До речі, у такий спосіб можна вводити не тільки імена змінних , що простежуються, але й вирази - середовище обчислить і покаже значення уведеного виразу.

Довідкова служба Турбо Паскаля

Невід'ємною складовою частиною середовища Турбо Паскаля є убудована довідкова служба. Якщо Ви досить добре володієте англійською мовою, у Вас не буде проблем при роботі з Турбо Паскалем: у скрутній ситуації досить натиснути F1 і на екрані з'явиться необхідна довідка. Ця

довідка залежить від поточного стану середовища, тому довідкову службу називають контекстно-чутливою. Наприклад, якщо натиснути F1 у момент, коли середовище виявило помилку в програмі, у довідці будуть повідомлені додаткові відомості про причини помилки й дані рекомендації з її усунення.

Існують чотири способи звернення до довідкової служби безпосередньо з вікна редактора:

F1 - одержання контекстно-контекстно-залежної довідки;

Shift-F1 - вибір довідки зі списку доступних довідкових повідомлень;

Ctrl-F1 - одержання довідки про потрібну стандартну процедуру, функцію, про стандартну константу або змінну;

Alt-F1 - одержання попередньої довідки.

По команді Shift-F1 на екрані з'явиться вікно, що містить упорядкований за алфавітом список стандартних процедур, функцій, типів, констант і змінних, для яких можна одержати довідкову інформацію.

Цю же довідку можна одержати й по-іншому. Надрукуйте на екрані ім'я процедури (функції, типу й т.д.) або підведіть курсор до наявному в тексті стандартному імені й натисніть Ctrl-F1. Середовище проаналізує найближче оточення курсору, виділить ім'я й надасть потрібну довідку.

У багатьох випадках довідка містить невеликий приклад, що ілюструє відповідні можливості Турбо Паскаля. Не квапитеся запам'ятовувати його або записувати на аркуші паперу: його можна «вирізати» з довідки й перенести у вікно редактора. Для цього після виклику довідки натисніть Alt-E, виберіть у додатковому меню, що з'явилося, Copy examples (копіювати приклади) і натисніть Enter -текст приклада буде скопійований у внутрішній буфер редактора. Для добування прикладу з буфера, натисніть Esc, щоб вийти з довідкової служби, підвести курсор до вільного рядка у вікні редактора, натисніть Shift-Insert (копіювання вмісту буфера в текст програми) і Ctrl-K H, щоб не було виділення скопійованого тексту кольорами.

Список використаної літератури

1. С.Гудман, С.Хидетниєми. Введення в розробку й аналіз алгоритмів - Москва ,Мір ,1981р.-364 с.
2. А.Ахо, Дж.Хопкрофт, Дж.Ульман. Побудова й аналіз обчислювальних алгоритмів- Москва , Мир , 1979 р.-535с.
3. В. Турский. Методологія програмування-москва ,Мір ,1981р.-263 с.

4. Інформатика: Базовий курс / С.В.Симонович й ін. - Пітер, 2002.- 640с.
5. Методичний посібник для виконання курсової роботи з дисципліни «Основи інформаційних технологій і програмування»/Сост.: В.Г.Суслова, И.В.Диннік - Донецьк, ДонНТУ, 2005-68с.
6. Архангельський А.Я. Мова Pascal й основи програмування в Delphi. Навчальний посібник - М.:ТОВ “Біном^-Пресс” ,2004.- 496 с.
7. Методична допомога з побудови алгоритмів і проектуванню програм / Сост.: И.Н. Кузык-Артамонова, И.В. Диннік - Донецьк: ДонНТУ, 2006 - 57 с.

Додаток 1.Зразок виконання титульного аркуша

ДЕРЖАВНИЙ ВИЩІЙ НАВЧАЛЬНИЙ ЗАКЛАД
МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра “Обчислювальної математики і програмування”

ЗВІТ

з лабораторної роботи № 1

за темою:

"xxxxxxxxxxxxxxxxxxx"

з дисципліни “xxxxxxxxxxxxxxxxxxxxxx”

Виконав студент гр. ХХ-08 І.І.Іванов

Викладач П. П. Петров

Донецьк, 2008 р.

Додаток 2. Вимоги до оформлення лабораторної роботи

Стандарт ДонНТУ встановлює загальні вимоги по побудові, викладу й оформленню студентами денної й заочної форм навчання Донецького Національного Технічного Університету інформації дипломних і курсових проектів (робіт), звітів по лабораторних роботах, розрахунково-графічним роботам і всім видам практики. Звіт по лабораторній роботі оформляють на

аркушах формату А4. Текст звіту варто друкувати, дотримуючи наступних розмірів полів: верхнє, ліве й нижнє - не менш 20 мм, праве - не менш 10 мм. У звіті повинні бути чіткі, не розпливчасті лінії, букви, цифри й інші знаки. Помилки, описки й графічні неточності допускається виправляти підчищенням або зафарбуванням білою фарбою й нанесенням виправлень на тім же місці або між рядків. Сторінки звіту варто нумерувати арабськими цифрами, дотримуючи наскрізної нумерації по всьому тексту. Номер сторінки проставляють у правому верхньому куті сторінки без крапки наприкінці.

Звіт по лабораторній роботі повинен містити:

1) ціль роботи;
2) завдання; завдання на роботу розміщається відразу ж після титульного аркуша й визначає об'єм і порядок виконання роботи в конкретному виконанні;
3) математичну модель і її опис;
4) вихідні дані до роботи й попередні розрахунки (якщо вони необхідні);
5) результати роботи;
6) висновки; висновки являють собою оцінку основних результатів, отриманих студентом у підсумку виконання лабораторної роботи.

МЕТОДИЧНИЙ ПОСІБНИК

ПО ОБЧИСЛЮВАЛЬНІЙ МАТЕМАТИЦІ Й ПРОГРАМУВАННЮ

(учбово-методичний посібник для студентів спеціальностей ХТ і ТХВ)

Укладачі: И. В. Диннік, ст. викладач

О.А.Тихонова, асистент