Перелік
лабораторних робіт з курсу «Програмування»
на 1 семестр 2016/17 н.р.

Лабораторна робота № 1

Записати у зошиті дату власного народження (день, місяць, рік) у системах числення з основами 2, 16.

Лабораторна робота № 2

Напишіть програму, яка друкує охайне повідомлення (транслітерація допускається) такого змісту **`Цю програму створив студент … групи ….`** Замість … вставте особисті дані (прізвище, ім'я та по-батькові, номер групи).

Hints:

1. Див. програму «Hello World».
2. Якщо не подобається транслітерація, див. документацію на функцію setlocale.

Лабораторна робота № 3

Напишіть програму, яка запитує у користувача значення двох дійсних змінних та у відповідь друкує значення введених змінних, математичну формулу (як plain text), за якою обчислюється вираз, та значення виразу для цих значень змінних.

На початку роботи програми має виводитися інформація щодо виконавця та умова задачі. Вхідні дані вважати коректними.

1. $x^2 - \frac{3}{11}yx$
2. $\frac{2}{3} + \frac{y}{x+2}$
3. $\frac{4}{7}x + y$
4. $4y^2 - \frac{11}{3}x$
5. $\frac{1}{7} + \frac{2+y}{x}$
6. $x + \frac{14}{17}y - 6$
7. $\frac{4}{3}y(x+1) - 11$
8. $\frac{1}{7} + \frac{1+y}{xy}$
9. $\frac{x}{3(y+1)} + \frac{1}{3}$
10. $\frac{3x-1}{7y} + \frac{1}{3}$
11. $\frac{1}{xy} + \frac{3}{41}x$
12. $\frac{4}{13}x(2-y) - \frac{11}{xy}$
13. $\frac{1}{13} + \frac{2x+y}{y-x}$
14. $x + \frac{5}{17}y^2$
15. $\frac{1}{3}x - \frac{2}{7}y$
16. $\frac{1}{11} + \frac{1+y}{x}$

Вимоги

1. У середовищі програмування створено проект, в який додано одиницю трансляції з текстом програми. Програма успішно компілюється й будується виконуваний файл.
2. Програма відповідає такому алгоритму: повідомити призначення програми, ввести вхідні дані, провести обчислення, вивести результати. Відповідні частини тексту програми розташовуються послідовно й **не** перетинаються.
3. На початку роботи програми виводиться інформація щодо виконавця та умова задачі.
4. Значення змінних вводяться користувачем.
5. Наявні зрозумілі підказки на введення даних.
6. Програма правильно виконує обчислення виразу за введеними даними, що належать області визначення виразу.
7. Наприкінці у **зрозумілому** вигляді виводяться введені дані та результат обчислення на них.

Лабораторна робота № 4, версія 1

Напишіть програму, яка запитує у користувача значення дійсної змінної та у відповідь друкує значення введених змінних, математичну формулу, за якою обчислюється вираз, та значення виразу для цих значень змінних. На початку роботи програми має виводитися інформація щодо виконавця та умова задачі.

Вхідні дані вважати коректними.

1. $\log_3(x) + \pi$
2. $\arcsin(1/3) + 1/(1-x)$
3. $\mathrm{tg}(5)+e/(x^2)$
4. $\mathrm{arcctg}(6\sqrt{x}) + \pi$
5. $(\pi+e)/(\pi + ex)$
6. $\mathrm{tg}(1/8) + \sqrt{1-x}$
7. $\log_x(1000) + \pi/2$
8. $\sqrt{x\ \cos(1/8)}$
9. $\mathrm{ctg}(1/5) + e(\ ½\ \sqrt{x}\ )$
10. $\mathrm{arctg}(3) + 2e/\sqrt{1-x^2}$
11. $\arccos(1/4) + \log_2(x)$
12. $1/(\sin(1/3) - x)$
13. $1/(\sin(1/5) - \log_3(x))$
14. $½\cos x/(\sin x - \mathrm{tg}\,x)$
15. $(½ - 1/x)/(\ ½ + 1/x)$
16. $(1 + x)/(\log_3(x) + \pi)$
17. $\left|\frac{1-\sqrt{1-4x^2}}{2x}\right|$
18. $\log_{10}(\mathrm{tg}\ x)$

Вимоги

1. У середовищі програмування створено проект, в який додано одиницю трансляції з текстом програми. Програма успішно компілюється й будується виконуваний файл.
2. Програма відповідає такому алгоритму: повідомити призначення програми, ввести вхідні дані, провести обчислення, вивести результати. Відповідні частини тексту програми розташовуються послідовно й **не** перетинаються.
3. На початку роботи програми виводиться інформація щодо виконавця та умова задачі.
4. Значення змінних вводяться користувачем.
5. Наявні зрозумілі підказки на введення даних.
6. Програма правильно виконує обчислення виразу за введеними даними, що належать області визначення виразу.
7. Програма використовує значення математичних констант π та e (якщо вони наявні в заданій формулі) з стандартної математичної бібліотеки.
8. Наприкінці в **зрозумілому** вигляді виводяться введені дані та результат обчислення на них.
9. Використання власних глобальних змінних НЕ допускається.

Лабораторна робота № 4, версія 2

Модифікувати ЛР № 4.1 (ЛР № 4, версія 1) так, щоб обчислення виразу було виділено в окрему функцію, яка викликається з головної.

Додаткові вимоги до версії 2 (перші 9 такі самі, як й до ЛР № 4.1)

10. Код обчислення виразу виділено в окрему функцію f.
11. Функція f належно задокументована.
12. Функція f відповідає **виключно** за обчислення виразу. Спілкування з користувачем до її відповідальності **НЕ** входить.
13. Для обчислення виразу за введеними користувачем даними використовується функція f.

Лабораторна робота № 4, версія 3

Модифікувати ЛР № 4.2 так, щоб функція обчислення виразу була виділена в окрему «бібліотеку» (розташовувалася в іншій одиниці трансляції ніж main).

Додаткові вимоги до версії 3 (перші 13 такі самі, як й до ЛР № 4.2)

14. Означення функції f виділено в окрему одиницю трансляції.

15. Створено відповідний заголовний файл, що містить прототип функції f з відповідним коментарем-документацією.

16. В одиниці трансляції, де розташовано main, підключається створений заголовний файл. Підключення файлу з означенням функції НЕ допускається.

Лабораторна робота № 4, версія 4

Модифікувати ЛР № 4.3 так, щоб за вхідних даних, що не належать області визначення виразу, друкувалося відповідне повідомлення про неможливість обчислень.

Додаткові вимоги до версії 4 (перші 16 такі самі, як й до ЛР № 4.3)

17. За даних, що не належать області визначення виразу, обчислення за формулою не відбуваються, і в цьому випадку виводиться відповідне повідомлення щодо неможливості проведення обчислень за введеними вхідними даними.

18. У коді відсутнє дублювання.

19. Механізм винятків не використовувати!

!!! Зверніть увагу на дотримання вимоги 2.

Лабораторна робота № 4, версія 5

Модифікувати ЛР № 4.4 згідно з додатковими вимогами.

Додаткові вимоги до версії 5 (перші 19 такі самі, як й до ЛР № 4.4)

20. У коді наявна окрема функція copyright, що виводить інформацію щодо виконавця (і більше нічого!).

21. Виведення інформації про виконавця здійснюється за допомогою виклику цієї функції.

22. Функція copyright розташована в окремій одиниці трансляції.

23. Створено відповідний заголовний файл, що містить прототип функції copyright.

24. В одиниці трансляції, де розташовано main, підключається створений заголовний файл. Підключення файлу з означенням функції copyright НЕ допускається.

25. Одиниця трансляції з функцією copyright відокремлена в окремий проект (project) (з типом конфігурації static library), який розташований в одному solution з проектом лабораторної роботи. Для проекту лабораторної роботи правильно вказано залежності (Project dependencies) (оскільки він використовує створену статичну бібліотеку, то він від неї залежить).

Лабораторна робота № 4, версія 6

Модифікувати ЛР № 4.5 так, щоб обробка некоректних вхідних даних використовувала механізм винятків.

Вимоги до версії 6

Вимоги 1-18 та 20-25 такі самі, як й до ЛР № 4.5. Та ще:

26. Обробка помилок використовує механізм винятків.

Загальні вимоги до коду лабораторних робіт (2016-17)

Увага! Усі подальші лабораторні роботи повинні відповідати наступним вимогам:

1. **Студент вільно орієнтується в коді лабораторної роботи, яку він здає, розуміє використані синтаксичні елементи мови, зміст та призначення частин коду.**

2. У середовищі програмування створено проект/проекти, в який додано необхідні одиниці трансляції з текстом програми. На початку кожного оригінального файлу в коментарях вказано його автора. Усі одиниці трансляції успішно компілюються, успішно створюється виконуваний файл. Програма запускається.

3. На початку роботи програми виводиться інформація щодо виконавця (за допомогою функції з раніш розробленої статичної бібліотеки) та умова задачі.

4. Інтерфейс має бути зручним та адекватним. Користувач програми повинен мати можливість розібратися в інтерфейсі без допомоги виконавця (зокрема, це стосується введення даних; також повинно бути зрозуміло, що виводить програма). Слід притримуватися принципу найменшого здивування.

5. Для некоректних вхідних даних програма повинна повідомляти про це користувача (якщо інше не передбачається умовою). Якщо умова задачі не вимагає точної діагностики помилок, то її можна не робити. Аварійне завершення програми є неприпустимим.

6. Код не повинен містити дублювання та невикористовувані фрагменти.

7. Система запису позначень та термінології повинна бути однорідною (зокрема, це стосується йменування змінних та функцій).

8. Код має бути структурованим за деякими виключеннями, переважно пов’язаними з обробкою помилкових або неочікуваних ситуацій.

Завжди слід виконувати правило: кожен блок (тут в розумінні «логічна частина коду», а не compound statement) повинен мати один вхід (інструкцію, з якої починається виконання блоку) та один вихід (інструкцію, якою завершується виконання блоку). «Відсікання» часткових випадків на початку виконання функції є допустимим (наприклад, «некоректних» значень параметрів або тривіальних випадків обчислення). Також, якщо явно не сказано інше, для обробки помилок та неочікуваних ситуацій допускається використання механізму винятків, але без зловживань.

9. Код має бути зрозумілим.

10. На виконання п. 9 імена змінних бажано робити змістовними та не зловживати коментарями.

11. На виконання п. 9 до прототипу функції варто додавати коментарі з перед- та післяумовами, та поведінки функції за некоректних даних (а також призначенням та описом параметрів, якщо їх не вдалося назвати так, щоб це стало є зрозумілим з їхніх назв).

12. Слід дотримуватися функціонального проектування.

Зокрема, кожен фрагмент коду/функція повинні мати рівно один обов’язок.

Класи не повинні мати різнопланові обов’язки.

13. Розташування коду по одиницях трансляції.

Код розташовано в кількох одиницях трансляції, існує логіка розподілу коду по одиницях трансляції (зокрема, за можливістю сильна зв’язність (cohesion) всередині одиниці трансляції та за можливістю слабкий зв’язок (coupling) між різними одиницями трансляції).

При розподілі функцій по одиницях трансляції функції/класи окремої одиниці трансляції повинні бути між собою логічно зв’язані. Наприклад, одиниця трансляції може визначати набір математичних функцій або функції, розташовані в одиниці трансляції разом вирішують спільну задачу (наприклад, реалізовують тестування певної математичної функції).

При проектуванні одиниць трансляції варто згадувати про функціональне проектування. Функції однієї одиниці трансляції не повинні спільно вирішувати більше однієї різнопланової задачі. Розташування в одній одиниці трансляції як самої функції обчислення, так й функцій для її тестування є помилковим: у цьому випадку одиниця трансляції відповідає вже не тільки за обчислення, але й за тестування запрограмованого обчислення.

Має сенс відокремлювати функції розв'язання поставленої задачі від функцій, суто допоміжних для їх демонстрації.

Функції, що разом вирішують спільну задачу й між якими є суттєві залежності, варто розміщувати не в окремих одиницях трансляції, а в одній. Наприклад, функція обчислення якоїсь математичної функції f та функція перевірки належності аргументів області визначення функції f мають розташовуватися в одній одиниці трансляції.

Підключення (#include) cpp-файлів не допускається (крім шаблонів). Тобто замість власних файлів з означеннями функцій та змінних мають підключатися відповідні файли з їхніми оголошеннями.

Клієнтський код класів має відокремлюватися від коду класів в іншу одиницю(і) трансляції.

Різні за призначенням та логічно мало зв'язані між собою класи мають розташовуватися в різних одиницях трансляції.

Назви відповідних h- та cpp-файлів мають бути однаковими (з точністю до розширення, звісно).

14. Наступних речей слід уникати:
    1. дублювання фрагментів коду
    2. довгі функції
    3. складні умовні вирази
    4. використання неіменованих «магічних» констант
    5. непристойна демонстрація (зокрема, код-клієнт має знати про код-сервер тільки те, що йому дійсно потрібно знати; зайвих для компіляції підключень бібліотек у заголовних файлах бути не повинно; глобальних змінних також слід уникати)
    6. функції, які вирішують кілька задач одночасно
15. Код має захищати свої дані від несанкціонованих змін.

Про відмову від глобальних змінних було сказано раніше. Також не використовувати без потреби посилання та вказівники в якості параметрів. Якщо параметр технічно передається функції адресою (й така передача є доцільною та/або єдиною синтаксично можливою) і якщо призначення функції не передбачає змін цього параметру, то треба блокувати зміни такого параметра на рівні компіляції (користуємось const). Класи мають відповідати принципу інкапсуляції.

16. При роботі з динамічними даними усі створені динамічні змінні мають коректно знищуватися під час виконання програми.
17. Робота з пам'яттю ведеться коректно.
18. Стиль програмування відповідає принципам ООП.
19. Методи розроблених класів реалізовано за межами означень класів.

Лабораторна робота № 5

Протабулювати функції $f(x)$, $S(x)$ та $S(x)-f(x)$ на відрізку $[a; b]$ з кроком h, де $a_0<a$, $b<b_0$ та

1) $a_0 = -0.9,\ b_0 = 0.9, f(x) = \operatorname{arctg} x,\ S(x) = \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n+1}}{2n+1}$;

2) $a_0 = -0.9,\ b_0 = 0.9, f(x) = \operatorname{ch} x,\ S(x) = \sum_{n=0}^{+\infty} \frac{x^{2n}}{(2n)!}$;

3) $a_0 = -0.9,\ b_0 = 0.9, f(x) = \operatorname{sh} x,\ S(x) = \sum_{n=0}^{+\infty} \frac{x^{2n+1}}{(2n+1)!}$;

4) $a_0 = -0.9,\ b_0 = 0.9, f(x) = \frac{m!}{(1-x)^{m+1}},\ S(x) = \sum_{n=0}^{+\infty} \frac{(n+m)!}{n!} x^n$;

5) $a_0 = -0.9,\ b_0 = 0.9, f(x) = \frac{2x}{(1-x)^3},\ S(x) = \sum_{n=1}^{+\infty} n(n+1)x^n$;

6) $a_0 = -\pi/5,\ b_0 = \pi/5, f(x) = \cos x,\ S(x) = \sum_{n=0}^{+\infty} \frac{(-1)^n x^{2n}}{(2n)!}$;

7) $a_0 = -1,\ b_0 = 1, f(x) = 2(\arcsin(\frac{x}{2}))^2,\ S(x) = \sum_{n=0}^{+\infty} \frac{(n!)^2}{(2n+2)!} x^{2n+2}$;

8) $a_0 = -1/3,\ b_0 = 1/3, f(x) = \frac{1}{2}(\arcsin(2x)),\ S(x) = \sum_{n=0}^{+\infty} \frac{(2n)!}{(n!)^2(2n+1)} x^{2n+1}$;

9) $a_0 = 1/10,\ b_0 = 1, f(x) = \ln x,\ S(x) = \sum_{n=1}^{+\infty} \frac{(-1)^{n-1}(x-1)^n}{n}$;

10) $a_0 = -1/5,\ b_0 = 1/5, f(x) = (1-4x)^{-\frac{3}{2}},\ S(x) = \sum_{n=0}^{+\infty} \frac{(2n+1)!}{(n!)^2} x^n$;

11) $a_0 = -1/5,\ b_0 = 1/5, f(x) = -\sqrt{1-4x},\ S(x) = \sum_{n=0}^{+\infty} \frac{(2n)!}{(n!)^2(2n-1)} x^n$;

12) $a_0 = -1/5,\ b_0 = 1/5, f(x) = \frac{1}{\sqrt{1-4x}},\ S(x) = \sum_{n=0}^{+\infty} \frac{(2n)!}{(n!)^2} x^n$;

13) $a_0 = -1/5,\ b_0 = 1/5, f(x) = \frac{1-\sqrt{1-4x}}{2x},\ S(x) = \sum_{n=0}^{+\infty} \frac{C_{2n}^n}{(n+1)} x^n$.

При обчисленні за значення $S(x)$ прийняти першу часткову суму S_n, за якої $|S_n - S_{n-1}| < \varepsilon$. Значення a, b, h та ε задає користувач.

Вимоги

0. Див. Загальні вимоги.

1. У середовищі програмування створено проект, в який додано необхідні одиниці трансляції з текстом програми. Програма успішно компілюється й будується виконуваний файл.

2. На початку роботи програми виводиться інформація щодо виконавця (за допомогою функції з раніш розробленої статичної бібліотеки) та умова задачі.

3. Хід виконання програми відповідає такому алгоритму: повідомити інформацію щодо виконавця та призначення програми, ввести вхідні дані, вивести задані вхідні дані та табличку результатів або повідомити про некоректність вхідних даних. Відповідні частини тексту програми розташовуються послідовно.

4. Значення змінних вводяться користувачем. Зокрема, користувач задає

– значення a початку відрізку, на якому табулюється функція;

– значення b кінця відрізку, на якому табулюється функція;

– крок h з яким виконується табуляція функції;

– значення ε, яке обумовлює точність обчислення;

– у 4-му варіанті користувач також задає додатнє натуральне значення m.

5. Наявні зрозумілі підказки на введення даних.

6. Значення a_0 і b_0 «прошиті» в коді (наприклад, const double a0=1000.0;).

7. Наявні функції обчислення значень $f(x)$ і $S(x)$, які за допустимих значень обчислюють ці значення правильно. Якщо в математичній бібліотеці наявна функція для обчислення $f(x)$, то власну функцію писати НЕ треба. ***Увага***: функція для обчислення $S(x)$ приймає не тільки x, але й ε (а в 4-му варіанті ще й m).

8. Обчислення значення $S(x)$ використовує рекурентні співвідношення та є якомога економним за кількістю виконуваних обчислень.

9. За потреби (якщо сума збігається дуже-дуже повільно) обчислення $S(x)$ дозволяється додатково обмежити розглядом не більше, ніж N перших часткових сум S_n. Відповідне значення варто оформити незмінюваною цілою локальною змінною функції обчислення $S(x)$ (hint: у тілі функції: const int N=1000;). Використання неіменованих числових констант для позначення граничної кількості обчислених часткових сум слід уникати.

10. Програма **НЕ** використовує масиви та інші структури даних для зберігання послідовностей значень.

11. Наявна функція, що за коректних значень a, b, h, ε (та m у 4-му варіанті) виконує табуляцію функцій $f(x)$, $S(x)$ та $S(x)–f(x)$ на відрізку $[a; b]$ з кроком h. Табуляція полягає в тому, що друкується табличка значень, а саме для кожної точки $x=a$, $a+h$, $a+2h$, …, $a+[(b-a)/h]\cdot h$, а також для точки b (якщо вона відсутня в наведеному переліку), у окремому рядку виводяться значення x, $f(x)$, $S(x)$ та $S(x)–f(x)$.

Hint: зверніть увагу, щоб значення в останній точці проміжку не друкувалося двічі.

Символи псевдографіки для оформлення таблички можна не виводити, але стовпці таблиці мають чітко проглядатися (*hint*: задавайте ширину поля виведення та притискайте праворуч). Дійсні числа виводити у форматі з фіксованою крапкою з поміркованою кількістю знаків після крапки; для значення $S(x)–f(x)$ виводити не менше 6 знаків після крапки.

12. Табуляція є економною за кількістю виконуваних викликів функцій, що обчислюють f та S (ось тут $f(x)$, $S(x)$ та $S(x)–f(x)$ чотири виклики, але достатньо двох).

13. Наявна функція, що друкує рядок із значеннями x, $f(x)$, $S(x)$ та $S(x)–f(x)$.

14. У випадку некоректних вхідних даних табулювання не виконується.

15. У випадку некоректних вхідних даних виводиться відповідне повідомлення.

16. Перевірка помилок консольного введення може бути відсутня.

17. Використання власних глобальних змінних (крім глобальних констант для a_0 і b_0) НЕ допускається.

18. Програма складається зі статичної бібліотеки (виведення інформації про виконавця) та ще якнайменш двох одиниць трансляції.

19. Зокрема, функції обчислення $f(x)$ і $S(x)$ виділено в окрему одиницю трансляції, до якої створено відповідний заголовний файл. У цьому ж заголовному файлі мають бути значення a_0 і b_0.

20. За потреби можуть бути наявні й інші власні функції, крім зазначених вище.

21. Власні функції належним чином задокументовані.

22. У заголовних файлах записано ті й тільки ті оголошення, про які повинен знати користувач відповідної бібліотеки.

23. Підключення (#include) cpp-файлів не допускається.

24. У коді відсутнє дублювання.

25. У коді відсутні «магічні константи» (неіменовані константи, що, наприклад, задають межі a_0 і b_0).

26. Кожна функція (можливо, за винятком main()) має рівно один обов’язок.

Лабораторна робота № 6

Написати програму, що за числовим масивом, що вводиться користувачем, знаходить такі значення:

1) середнє арифметичне індексів усіх мінімальних елементів та значення мінімального елементу;

2) середнє арифметичне індексів усіх максимальних елементів та значення максимального елементу;

3) мінімальний невід'ємний елемент масиву;

4) максимальний від'ємний елемент масиву;

5) суму мінімального та максимального елементів масиву;

6) суму всіх елементів масиву;

7) кількість від'ємних елементів;

8) середнє арифметичне усіх додатних елементів масиву;

9) середнє арифметичне усіх від'ємних елементів масиву;

10) кількість додатних елементів;

11) середнє арифметичне індексів мінімального та максимального елементів масиву;

12) середнє арифметичне індексів усіх від'ємних елементів масиву;

13) середнє арифметичне індексів усіх додатних елементів масиву;

14) суму усіх парних елементів масиву.

Вимоги

0. Див. Загальні вимоги.

1. У середовищі програмування створено проект, в який додано необхідні одиниці трансляції з текстом програми. Програма успішно компілюється й будується виконуваний файл.

2. На початку роботи програми виводиться інформація щодо виконавця (за допомогою функції з раніш розробленої статичної бібліотеки) та умова задачі.

3. Хід виконання програми відповідає такому алгоритму: повідомити інформацію щодо виконавця та призначення програми, ввести вхідні дані, вивести задані вхідні дані та табличку результатів або повідомити про некоректність вхідних даних. Відповідні частини тексту програми розташовуються послідовно.

4. Програма складається зі статичної бібліотеки (виведення інформації про виконавця) та ще якнайменш двох одиниць трансляції.

5. Функції роботи з масивом (ввести, вивести, обробити) зібрані в окрему одиницю трансляцію, до якої створено заголовний файл.

6. Введення масиву реалізовано у вигляді функції, яка повертає з виклику як сам масив, так й кількість його елементів.

7. Числовий масив вводиться користувачем. Користувач не знає скільки елементів збирається ввести.

8. При введенні масиву робота з пам'яттю ведеться коректно (тобто контролюється кількість введених елементів і чужа пам'ять не використовується).

9. Перевірка помилок консольного введення може бути відсутня.

10. Основна функціональність програми (обробка вже наявного масиву) реалізована у вигляді окремої функції.

11. Наявна функція, що виводить масив. Виведення ведеться через кому, у кінці послідовності значень, якщо вона непорожня ставиться крапка й виконується перехід на новий рядок. За порожнього масиву друкується відповідне повідомлення й також виконується перехід на новий рядок.

12. За потреби можуть бути наявні й інші власні функції, крім зазначених вище.

13. Підключення (#include) cpp-файлів не допускається.

14. Програма має передбачати обробку масивів довжини до 1000 елементів включно.

Лабораторна робота № 7

Варіанти 1-6

Написати програму, що за числовою послідовністю знаходить такі значення:

1) кількість «серій» у послідовності (серією послідовності $(a_n)_{n\in N}$ будемо вважати її максимальну неспадну/незростаючу підпослідовність);

2) кількість «піків» у послідовності (піком послідовності $(a_n)_{n\in N}$ будемо вважати максимальну підпослідовність $a_i, \ldots, a_j, \ldots, a_k$, таку, що $a_i \le \ldots \le a_j \ge a_{j+1} \ge \ldots \ge a_k$);

3) довжину найдовшої «серії» послідовності;

4) довжину найдовшого «пику» послідовності;

5) значення верхівки останнього найдовшого «пику» послідовності;

6) довжину найдовшої «серії» послідовності, що не містить нульових значень;

Послідовність вводиться користувачем. У кінці програма має повідомляти як результати обчислень, так й уведені значення.

Вимоги до варіантів 1-6

0. Див. «Вимоги до коду лабораторних робіт».

1. У середовищі програмування створено проект, в який додано необхідні одиниці трансляції з текстом програми. Програма успішно компілюється й будується виконуваний файл.

2. На початку роботи програми виводиться інформація щодо виконавця (за допомогою функції з раніш розробленої статичної бібліотеки) та умова задачі.

3. Хід виконання програми відповідає такому алгоритму: повідомити інформацію щодо виконавця та призначення програми, ввести вхідні дані, вивести задані вхідні дані та табличку результатів або повідомити про некоректність вхідних даних. Відповідні частини тексту програми розташовуються послідовно.

4. Програма складається зі статичної бібліотеки (виведення інформації про виконавця) та ще якнайменш двох одиниць трансляції.

5. Послідовність вводиться користувачем та зберігається в масиві. Користувач не знає скільки елементів збирається ввести.

6. Функції роботи з масивом (ввести, вивести, обробити) зібрані в окрему одиницю трансляцію, до якої створено заголовний файл.

7. Введення масиву реалізовано у вигляді функції, яка повертає з виклику як сам масив, так й кількість його елементів.

8. При введенні масиву робота з пам'яттю ведеться коректно (тобто контролюється кількість введених елементів і чужа пам'ять не використовується).

9. Перевірка помилок консольного введення може бути відсутня.

10. Основна функціональність програми (обробка вже наявного масиву) реалізована у вигляді окремої функції.

11. Наявна функція, що виводить масив. Виведення ведеться через кому, у кінці послідовності значень, якщо вона непорожня ставиться крапка й виконується перехід на новий рядок. За порожнього масиву друкується відповідне повідомлення й також виконується перехід на новий рядок.

12. За потреби можуть бути наявні й інші власні функції, крім зазначених вище.

13. Підключення (#include) cpp-файлів не допускається.

14. Програма має передбачати обробку послідовностей довжини до 1000 елементів включно.

(*Hint*: можна використати функції введення/виведення з ЛР №6.)

Варіанти 7-14

Рядок містить слова, розділені білими символами (з кодами від 0 до 31; тільки у цій лабораторній! також білим вважати символ підкреслення) та знаками пунктуації (, . ; :).

7) Перетворити рядок, видаливши всі знаки пунктуації, білі символи на початку та кінці рядка та замінивши послідовності білих символів на проміжок.

8) За рядком знайти довжину його найдовшого слова парної довжини та кількість таких слів.

9) Перетворити рядок, замінивши в ньому усі слова, довші 5 символів, на їх перші 5 символів, за яким йде знак **?**

10) Перетворити рядок, видаливши з нього всі слова довжини не більше 3.

11) Перетворити рядок, перегорнувши в ньому кожне слово непарної довжини.

12) Перетворити рядок, перегорнувши в ньому кожне друге слово, що починається з літери **Y**.

13) Перетворити рядок, переставивши в ньому пари сусідніх слів (наприклад, з «123 345 678» має отриматися «345 123 678»).

14) Перетворити рядок, перегорнувши в ньому кожне слово.

Рядок вводиться користувачем. У кінці програма має повідомляти як результати обчислень, так й уведені значення.

Вимоги до варіантів 7-14

0. Див. «Вимоги до коду лабораторних робіт».

1. У середовищі програмування створено проект, в який додано необхідні одиниці трансляції з текстом програми. Програма успішно компілюється й будується виконуваний файл.

2. На початку роботи програми виводиться інформація щодо виконавця (за допомогою функції з раніш розробленої статичної бібліотеки) та умова задачі.

3. Хід виконання програми відповідає такому алгоритму: повідомити інформацію щодо виконавця та призначення програми, ввести рядок, обробити та вивести результати обробки.

4. Програма використовує статичну бібліотеку (виведення інформації про виконавця).

5. Рядок вводиться користувачем. (**Увага!** У цій роботі дозволяється заборонити користувачу використовувати у рядку білі символи!) Для зберігання рядка використовується масив символів. Вважати, що користувач не може задати рядок, довший за 2048 символів.

6. Основна функціональність програми (обробка рядка) реалізована у вигляді окремої функції.

7. При обробці рядків робота з пам’яттю ведеться коректно.

8. Використання додаткових масивів не допускається. (Єдиний виняток – у варіанті 8 дозволяється (але не вимагається) використати допоміжний рядок для найдовшого слова.)

9. За потреби можуть бути наявні й інші власні функції, крім зазначених вище.

10. Підключення (#include) cpp-файлів не допускається.

Hint. Щоб прочитати з консолі рядок, усередині якого є білі символи (крім ′\n′) можна зробити таке

Підключити стандартні бібліотеки

```
#include <string>
#include <iostream>
```

Далі послідовність інструкцій задає зчитування з консолі рядка (кінцем введення буде натискання клавіші <Enter>) та збереження в пам’яті.

```
string s; getline(cin, s); const char *ps=s.c_str();
```

Після виконання фрагменту коду вказівник ps вказуватиме на початок зчитаного С-рядка. Треба звернути увагу на те, що відповідний рядок буде незмінюваним. За потреби виконувати над ним перетворення його слід скопіювати (див. функцію strcpy).

У випадку використання такого методу введення рядка, зчитування рядка має бути оформлено окремою функцією.

Лабораторна робота № 8

– За квадратною матрицею знайти суму її елементів, що розташовані

1) у верхньому чверть-трикутнику;

2) у нижньому чверть-трикутнику;

3) у правому чверть-трикутнику;

4) у лівому чверть-трикутнику.

– За квадратною матрицею надрукувати послідовність її елементів обходом трикутника над головною діагоналлю матриці "змійкою"

5) від лівого верхнього кута за горизонталями;

6) від правого верхнього кута за горизонталями;

7) від лівого верхнього кута діагоналями;

8) від правого верхнього кута діагоналями.

– За квадратною матрицею надрукувати послідовність її елементів обходом трикутника під головною діагоналлю матриці "змійкою"

9) від лівого нижнього кута за горизонталями;

10) від правого нижнього кута за горизонталями;

11) від лівого нижнього кута діагоналями;

12) від правого нижнього кута діагоналями.

– За квадратною матрицею надрукувати послідовність її елементів обходом центрального ромба

13) за спіраллю за годинниковою стрілкою;

14) за спіраллю проти годинникової стрілки;

15) "змійкою" за горизонталями від верхнього кута;

16) "змійкою" за діагоналями від верхнього кута.

Користувач задає розмірність матриці. Матриця заповнюється автоматично послідовними числами, наприклад, $\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{pmatrix}$.

У кінці програма має повідомляти матрицю, що оброблялася, та результат обробки/отриману послідовність.

Вимоги

0. Див. «Вимоги до коду лабораторних робіт».

1. У середовищі програмування створено проект, в який додано необхідні одиниці трансляції з текстом програми. Програма успішно компілюється й будується виконуваний файл.

2. На початку роботи програми виводиться інформація щодо виконавця (за допомогою функції з раніш розробленої статичної бібліотеки) та умова задачі.

3. Хід виконання програми відповідає такому алгоритму: повідомити інформацію щодо виконавця та призначення програми, ввести вхідні дані, вивести задані вхідні дані та результат роботи або повідомити про некоректність вхідних даних. Відповідні частини тексту програми розташовуються послідовно.

4. Програма складається зі статичної бібліотеки (виведення інформації про виконавця) та ще якнайменш двох одиниць трансляції.

5. Функції роботи з матрицею (заповнити, вивести стандартно, обробити) зібрані в окрему одиницю трансляцію, до якої створено заголовний файл.

6. Розмір матриці вводиться користувачем.

7. При заповненні матриці робота з пам’яттю ведеться коректно (тобто контролюється кількість введених елементів і чужа пам’ять не використовується).

8. Перевірка помилок консольного введення може бути відсутня.

9. Основна функціональність програми (обробка вже наявного масиву) реалізована у вигляді окремої функції.

10. Наявна функція, що виводить матрицю стандартним чином (по рядках, кожен рядок з нового рядка). За порожньої матриці друкується відповідне повідомлення. У варіантах 5-16 виведення послідовності елементів матриці згідно обходу ведеться через кому, якщо матриця порожня, то виводиться відповідне повідомлення.

11. За потреби можуть бути наявні й інші власні функції, крім зазначених вище.

12. Підключення (#include) cpp-файлів не допускається.

13. Програма повинна передбачати обробку матриць розмірності до 20×20 включно.

Лабораторна робота № 9

Написати програму, яка **відповідно до вмісту командного рядка** виконує задачу з лабораторних робіт 5, 6, 7 або 8.

Мають підтримуватися такі режими роботи програми.

help – виводиться довідка по аргументах командного рядка

lab5 – запускається задача 5

…

lab8 – запускається задача 8

За незрозумілого вмісту командного рядка виводиться довідка по аргументах командного рядка.

Вимоги

0. Див. «Вимоги до коду лабораторних робіт».

1. У середовищі програмування створено проект, в який додано необхідні одиниці трансляції з текстом програми. Програма успішно компілюється й будується виконуваний файл.

2. На початку роботи програми виводиться інформація щодо виконавця (за допомогою функції з раніш розробленої статичної бібліотеки).

3. Наявна функція help, що виводить довідку по аргументах командного рядка програми.

4. Робота з пам’яттю ведеться коректно.

5. Код кожної з лабораторних робіт 5, 6, 7, 8 модифікується так, щоб відповідний проект у головній функції викликав функцію laba5,…, laba8 (або з аналогічними назвами). Замість копіювання коду до ЛР №9 слід підключити відповідні проекти (вони повинні бути типу «статична бібліотека»).

6. Відповідно до вмісту командного рядка або викликається функція відповідної лабораторної роботи (наприклад, laba5 за аргументу lab5), або help.

7. У випадку незрозумілого вмісту командного рядка виводиться відповідне повідомлення і теж викликається функція help.

8. За відсутності робочої версії якоїсь з попередніх лабораторних робіт (наприклад, 8-ої) дозволяється у відповідному проекті зробити функцію laba8, яка друкує повідомлення з умовою лабораторної роботи про те, що ця лабораторна робота поки що не реалізована.