Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра ІМІ

КОМПЛЕКС НАВЧАЛЬНО-МЕТОДИЧНОГО ЗАБЕЗПЕЧЕННЯ

навчальної дисципліни

«Програмування ч.1»

підготовки бакалавра
напряму 6.050903 «Телекомунікації»

Розробник О.П.Малінін, ст. викладач каф. ІМІ,

Схвалено на засіданні кафедри ІМІ
Протокол від 31 серпня 2017 р. № 1

Харків 2017 р.

ЗМІСТ

1. Робоча програма 3
2. Конспект лекцій 14
3. Методичні вказівки до самостійної роботи і практичних занять 106
4. Методичні вказівки до лабораторних робіт 121
5. Контрольні завдання 155
6. Екзаменаційні запитання 162

Форма № Н - 3.04

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
(повне найменування вищого навчального закладу)

«Кафедра інформаційно – мережної інженерії»

"**ЗАТВЕРДЖУЮ**"
Декан факультету ІК
А.В. Снігуров
"_____" _____2017 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Програмування -1
(назва дисципліни)

напрям підготовки:_____6.050903_«Телекомунікації» _____

спеціальність_____
(шифр і назва спеціальності)

Спеціалізація _____
(назва спеціалізації)

факультет _____Інфокомунікацій_____
(назва інституту, факультету, відділення)

2016-2017 навчальний рік

Робоча програма розроблена на підставі навчального плану підготовки фахівців освітньо-кваліфікаційного рівня бакалавр за напрямом 6.050903 "Телекомунікації".

Розробник: О.П. Малінін, ст. викладач . каф. ІМІ.

Робочу програму схвалено на засіданні кафедри «ІК »

Протокол від “__31__”____серпня_____2017 р. № _1__

Завідувач кафедри ІМІ _____ В.М. Безрук
(підпис)

Робочу программу схвалено методичною комісією факультету ІК

Протокол від “__31__”_____серпня_____2017 р. № _1__

Голова методичної комісії _____Костромицький А.І.
(підпис)

1 ОПИС НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

<table>
<tr><td rowspan="2">Найменування показників</td><td rowspan="2">Галузь знань, напрям підготовки, освітньо-кваліфікаційний рівень</td><td colspan="2">Характеристика навчальної дисципліни</td></tr>
<tr><td>денна форма навчання</td><td>заочна форма навчання</td></tr>
<tr><td rowspan="2">Кількість кредитів 3</td><td>Галузь знань
0509 «Радіотехніка, радіоелектронні апарати та зв’язок»</td><td colspan="2" rowspan="2">Варіабельна</td></tr>
<tr><td>Напрям підготовки
6.050903 «Телекомунікації»</td></tr>
<tr><td>Модулів 1</td><td rowspan="9">Спеціальність (професійне спрямування):
«Інформаційні мережі зв’язку</td><td colspan="2">Рік підготовки:</td></tr>
<tr><td>Змістових модулів 3</td><td>2</td><td>3</td></tr>
<tr><td>Індивідуальних завдань 1</td><td colspan="2">Семестр</td></tr>
<tr><td rowspan="3">Загальна кількість годин 90</td><td>3</td><td>4</td></tr>
<tr><td colspan="2">Кількість годин</td></tr>
<tr><td>90</td><td>90</td></tr>
<tr><td rowspan="10">Тижневих годин для денної форми навчання: аудиторних – самостійної роботи студента -</td><td colspan="2">Аудиторні: 1) лекції, год</td></tr>
<tr><td>20</td><td>4</td></tr>
<tr><td colspan="2">2) практичні, год</td></tr>
<tr><td>6</td><td>4</td></tr>
<tr><td colspan="2">3) лабораторні, год</td></tr>
<tr><td>16</td><td>12</td></tr>
<tr><td colspan="2">4) консультації, год</td></tr>
<tr><td>6</td><td>6</td></tr>
<tr><td colspan="2">Самостійна робота, год</td></tr>
<tr><td>42</td><td>64</td></tr>
<tr><td colspan="2">в тому числі: 1) інд. завд., год.</td></tr>
<tr><td>4</td><td>14</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Вид контролю: залік</td></tr>
</table>

Примітка.
Співвідношення кількості годин аудиторних занять до загальної кількості годин становить:
для денної форми навчання – 45%;
для заочної форми навчання – 20%.

2 МЕТА І ЗАВДАННЯ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Особливістю курсу є те, що мова програмування С ++ розглядається як алгоритмічна, а не як об'єктно-орієнтована. Даний курс є необхідною базою для подальшого вивчення мови С ++.

знати:

- Етапи рішення задачі;
- Що таке мова програмування;
- Поняття алгоритму, види алгоритмів;
- Елементи програм на С ++;
- Синтаксис, семантику, та типи даних;
- Основні операції мови С++;
- Керуючі структури;
- Що таке масиви, структури, покажчики та функції.

вміти:

- Розробляти алгоритми для конкретних завдань;
- Розробляти програми за заданим алгоритмом;
- Володіти середовищем розробки додатків - Visual Studio

3 ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

(3-й семестр, денна форма; 4-й семестр, заочна)

Змістовий модуль 1. Введення і алгоритмізація.
Тема 1. Введення в програмування і рішення задач.
Тема 2. Алгоритмізація
Змістовий модуль 2. Основи С++
Тема 1. Синтаксис, семантика і розробка програм на С ++
Тема 2. Основні операції
Змістовий модуль 3. Керуючі структури
Тема 1 Умовні оператори
Тема 2. Цикли. Масиви
Тема 3. N мірні масиви
Змістовий модуль 4.
Тема 1. Строки. Функції.
Тема 2. Покажчики
Тема 3. Порозрядні оператори. Робота з файлами
Тема 4. Динамічна пам'ять
Тема 5. Структури даних

4 СТРУКТУРА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

(3-й семестр, денна форма; 4-й семестр, заочна)

Назви змістових модулів і тем	Кількість годин											
	денна форма						Заочна форма					
	Усього	у тому числі					Усього	у тому числі				
		л	п	лб	конс	с.р.		л	п	лб	конс	с.р.
1	2	3	4	5	6	7	8	9	10	11	12	13
Модуль 1												
Змістовий модуль 1. Введення і алгоритмізація.												
Тема 1. Введення в програмування і рішення задач	2	2						2				3
Тема 2. Алгоритмізація	6	2	2			2						3
Разом за зміст. мод. 1	8	4	2			2						
Змістовий модуль 2. Основи С++												
Тема 1. Синтаксис, семантика і розробка програм на С ++	2	2		4	2	6	11			4	2	5
Тема 2. Основні операції	2	2										
Разом за зміст. мод. 2		4		4	2	6	11			4	2	5
Змістовий модуль 3. Керуючі структури												
Тема 1. Умовні оператори	6	2	2	4		4	5					5
Тема 2. Цикли. Масиви	11	2	2			5	6			4		6
Тема 3 N мірні масиви	13	2		4	2	5	6			4	2	6
Разом за зміст. мод. 3	34	6	4	8	2	14	27			8	2	17
Змістовий модуль 4												
Тема 1. Строки. Функції.		2						**2**				
Тема 2. Покажчики		2		4								
Тема 5. Структури даних		2										
Разом за зміст. мод. 4		6		4								
Індивідуальні завдання												
РГЗ 1.	6					6						
Контрольні роботи 1							20					20
Усього годин за семестр	**90**	**20**	**6**	**16**	**6**	**42**	**90**	**4**	**4**	**12**	**6**	**64**

5 ТЕМИ ПРАКТИЧНИХ ЗАНЯТЬ

(3-й семестр, денна форма; 4-й семестр, заочна)

№	Назва теми	Кількість годин	
		денна	заочна
1	Алгоритмізація	2	
2	Цикли і масиви	2	2
3	Умови і цикли	2	
	Загальна кількість	**6**	**2**

6 ТЕМИ ЛАБОРАТОРНИХ ЗАНЯТЬ

(3-й семестр, денна форма; 4-й семестр, заочна)

№	Назва теми	Кількість годин	
		денна	заочна
1	Знайомство з середовищем розробки додатків - Visual Studio	4	4
2	Арифметичні висловлення. Організація інтерактивного введення і виведення	4	
3	Функції. Рядки і двовимірні масиви	4	4
4	Покажчики та оператори	4	4
	Загальна кількість	**16**	**12**

7 САМОСТІЙНА РОБОТА

(3-й семестр, денна форма; 4-й семестр, заочна)

№	Назва теми	Кількість годин	
		денна	заочна
1	Вивчення теоретичного матеріалу з використанням конспектів і навчальної літератури	20	28
2	Підготовка до лабораторних занять	8	6
3	"Безрозмірна" ініціалізація масивів	4	4
4	Розробка Алгоритму (РГЗ №1)	4	
5	Передача покажчиків і масивів в якості аргументів	3	3
6	Аргументи функції main (): argc і argv	3	3
11	Підготовка до виконання контрольних робіт № 1		20
	Загальна кількість	**42**	**64**

8 ІНДИВІДУАЛЬНІ ЗАВДАННЯ

8.1 Розрахунково-графічні завдання (РГЗ) та контрольні роботи (КР)

№	Назва теми	Кількість годин	
		денна	заочна
	Частина 1	3-й сем.	4-й сем.
1	РГЗ №1 Розробка Алгоритму	4	
3	Контрольні роботи № 1		20
	Загальна кількість	**4**	**20**

9. МЕТОДИ НАВЧАННЯ

Основні методи навчання – пояснювально-ілюстративний (лекція), практичний (проведення лабораторних робіт та практичних занять), перевірка знань та умінь (за результатами, контрольних робіт, контрольних завдань), робота з навчально-методичною літературою (самостійне опрацювання заданих розділів, виконання РГЗ тощо).

10 МЕТОДИ КОНТРОЛЮ ТА РЕЙТИНГОВА ОЦІНКА ЗА ДИСЦИПЛІНОЮ

10.1 1 (3-й семестр, денна форма)

10.1.1 Розподіл балів, які отримують студенти (Кількісні критерії оцінювання)

Для оцінювання роботи студента протягом семестру підсумкова рейтингова оцінка $O_{\text{сем}}$ розраховується як сума оцінок за різні види занять та контрольні заходи.

Вид заняття / контрольний захід	Оцінка
Лб № 1, 2	(12…20)×2 = 18…40
Контрольна точка 1	**24…40**
Лб № 3, 4	(12…20)×2 = 24…40
РГЗ №1	12…20
Контрольна точка 2	**36…60**
Всього за 2-й семестр	60…100

10.1.2 Якісні критерії оцінювання

Необхідний обсяг знань для одержання позитивної оцінки.

1. Поняття алгоритму. види алгоритмів. Блок-схеми. Графічна реалізація алгоритмів
2. Синтаксис, семантика і розробка програм на С ++. Типи даних
3. Основні операції мови С++
4. Потік управління. Умови та логічні вирази. Умовний оператор. Вкладені умовні оператори
5 Керуючі структури – Потік управління. Умови та логічні вирази. Умовний оператор. Вкладені умовні оператори
6. Цикли та масиви
7. Рядки. Функції. Функції бібліотеки cstring
8. Покажчики та змінні. Покажчики та масиви. Покажчики та рядки
9. Динамічний розподіл пам'яті з використанням операторів
10. Загальний формат оголошення структур. Масиви структур. Створення функцій, які працюють зі структурою

Критерії оцінювання роботи студента протягом семестру.

Задовільно, D, E (60-74). Мати мінімум знань і умінь. Відпрацювати та захистити всі лабораторні роботи та ІДЗ. Знати основи побудови алгоритмів. Знати основні операції. Вміти написати нескладну програму

Добре, C (75-89). Знати основні теми дисципліни. Відпрацювати та захистити всі лабораторні роботи та ІДЗ. Розробити алгоритм для вирішення завдання яке має середню складність. Написати програму середньої складності

Відмінно, A, B (90-100). Знати всі теми дисципліни. Відпрацювати та захистити всі лабораторні роботи, ІДЗ і реферат. Вміти написати програму будь-якої складності

Шкала оцінювання: національна та ECTS

Сума балів за всі види навчальної діяльності	Оцінка ECTS	Оцінка за національною шкалою	
		для екзамену, курсового проекту (роботи), практики	для заліку
96–100	A	відмінно	зараховано
90–95	B		
75–89	C	добре	
66–74	D	задовільно	
60–65	E		
35–59	FX	незадовільно з можливістю повторного складання	не зараховано з можливістю повторного складання
0-34	F	незадовільно з обов’язковим повторним вивченням дисципліни	не зараховано з обов’язковим повторним вивченням дисципліни

11 МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ ТА РЕКОМЕНДОВАНА ЛІТЕРАТУРА

11.1 Базова література

1 Белоцерковская И.Е. Галина Н.В. Катаева Л.Ю.Алгоритмизация. Введение в язык программирования С++2-е издание, исправленное "ИНТУИТ" 2016

2 Клейнберг Дж., Тардос Е.Алгоритмы: разработка и применение. Классика Computers Science / Пер. с англ. Е. Матвеева. — СПб.: Питер, 2016. — 800 с.: ил. — (Серия «Классика computer science»).

3 Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил. — Парал. тит. англ.

4 Дейл Н., Уимз Ч., Хедингтон М. Программирование на С++: Пер. с англ. - М.: ДМК, 2000. - 672 с.: ил., (Серия «Учебник»).

5 Шилдт Г. Самоучитель С++: Пер. с англ. — 3-е изд. — СПб.: БХВ-Петербург, 2003. — 688 с.

11.2 Допоміжна література

6 Доусон М. Изучаем С++ через программирование игр. - СПб.: Питер, 2016. - 352 с.:

7 Эллайн.С++. От ламера до программера. - СПб.: Питер, 2015. — 480с: ил.

8 Мунтян Александр Юрьевич. «Основы программирования на С++, Часть І».Учебное пособие по курсу. © «Физтехшкола» 2005г.

11.3 Методичні вказівки до різних видів занять

Конспект лекцій з курсу «Програмування-1» для студентів усіх форм навчання напряму 6.050903 – Телекомунікації ” –Х.: ХНУРЕ, 2016 Електронний варіант.

Методичні вказівки до самостійної роботи та практичних занять з дисципліни «Програмування-1» для студентів усіх форм навчання напряму 6.050903 – Телекомунікації Х.: ХНУРЕ, 2016 Електронний варіант.

Методичні вказівки до лабораторних робіт з дисципліни «Бази даних» для студентів усіх форм навчання напряму 6.050903 – Телекомунікації Х.: ХНУРЕ, 2016 Електронний варіант.

12 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

1. Програмній пакет - «Visual Studio 2017»

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра ІМІ

КОМПЛЕКТ СЛАЙД-ЛЕКЦІЙ

з дисципліни «Програмування ч.1»

для студентів усіх форм навчання
напряму 6.050903 «Телекомунікації»

Електронний документ

Харків 2017 р.

Комплект слайд - лекцій з дисципліни «Програмування ч.1» для студентів усіх форм навчання напряму 6.050903 «Телекомунікації»
[Електронний документ] / Упоряд.: О.П. Малінін . Харків: ХНУРЕ, 2017. 91 с.

Упорядник О.П. Малінін

ЗМІСТ

Лекція 1. Вступ в програмування і рішення задач 17
Лекція 2 Алгоритмізація 20
Лекція 3 Синтаксис, семантика і розробка програм на С ++ 29
Лекція 4 Основні операції 37
Лекція 5 Керуючі структури 48
Лекція 6 Введення і оформлення тексту 55
Лекція 7 Рядки. Функції 62
Лекція 8 N-вимірні масиви 71
Лекція 9 Покажчики 79
Лекція 10 Порозрядні оператори, Робота з файлами 85
Лекція 11 Динамічна пам'ять, і ще про оператори 93
Лекція 12 Структури даних 101

«Програмування ч.1»

ЛЕКЦІЇ

О. П. Малінін

Факультет «Інфокомунікацій»

Кафедра «Інформаційно-мережної інженерії»

ХНУРЕ

2017

Лекция №1 «Введення в програмування і рішення задач»

План

1 Введення в програмування

2 Що таке мова програмування

Введення в програмування

Програмування - планування, розподіл у часі або виконання завдань або подій

Комп'ютерне програмування - планування послідовності кроків, які повинен виконати комп'ютер

Комп'ютерна програма - список інструкцій, які повинен виповнити комп'ютер

Як пишуться програми

Процес написання програми складається з двох етапів: рішення

завдання та реалізація.

Етап рішення задачі.

1. Аналіз і специфікація. Зрозуміти і визначити суть завдання, а також технічні та функціональні вимоги до вирішення.

2. Загальне рішення (алгоритм). Розробити логічну послідовність кроків, що приводить до рішення.

3. Перевірка. Переконатися в правильності рішення, наприклад, повторивши всі його

етапи.

Що таке мова програмування:

Машинна мова - мова, що складається з команд в двійковому поданні, які використовуються безпосередньо комп'ютером.

Компілятор - програма, яка переводить мову високого рівня в машинний код.

Інтерпретатор - простий інтерпретатор аналізує і тут же виконує (власне інтерпретація) програму покомандно (або через підрядник), у міру надходження її вихідного коду на вхід інтерпретатора.

Вихідна програма - програма на мові програмування..

Об'єктна програма - програма в машинному коді, отримана в результаті трансляції вихідної про

Лекція 2 «Алгоритмізація»

План

1. Поняття алгоритму. види алгоритмів
2 Блок-схеми. Графічна реалізація алгоритмів

Мета даної лекції - ознайомити студентів з поняттям алгоритму; показати, що така абстрактна річ як алгоритм оточує нас у повсякденному житті.

Ознайомити студентів з поняттям блок-схеми; показати основні конструкції реалізації різних видів алгоритмів; показати принципи перевірки блок-схем і одержання по ним відповіді

2.1. Поняття алгоритму. види алгоритмів

Алгоритм є базовим поняттям для тих, хто хоче почати програмувати на будь-якій мові програмування. Будь-яке завдання може бути формалізована алгоритмічно. Щоб зрозуміти, з чого почати, розглянемо основні види алгоритмів.

Існує кілька визначень поняття алгоритму. Уявімо два найпоширеніших.

Алгоритм - послідовність чітко визначених дій, виконання яких веде до вирішення завдання. Алгоритм, записаний на мові машини, є програма рішення задачі.

Алгоритм - це сукупність дій, що призводять до досягнення результату за кінцеве число кроків.

Властивості алгоритмів:

1 Дискретність (від лат. Discretus - розділений, переривчастий) - це розбиття алгоритму на ряд окремих закінчених дій (кроків).

2 детермінованість (від лат. Determinate - визначеність, точність) - будь-яка дія алгоритму має бути строго і недвозначно визначено у кожному випадку.

Наприклад, алгоритм проїзду до одного якщо до зупинки підходять автобуси різних маршрутів, то в алгоритмі має бути вказаний конкретний номер маршруту 5. Крім того, необхідно вказати точну кількість зупинок, яке треба проїхати, скажімо, три.

3 Кінцівка - кожна дія окремо і алгоритм в цілому повинні мати можливість завершення.

4 Масовість - один і той же алгоритм можна використовувати з різними вихідними даними.

5. Результативність - алгоритм повинен призводити до достовірного вирішення

Приклади алгоритму:

1. Будь-який прилад, куплений в магазині, забезпечується інструкцією щодо його використання. Дана інструкція і є алгоритмом для правильної експлуатації приладу.

2. Кожен шофер повинен знати правила дорожнього руху. Правила дорожнього руху днозначно регламентують поведінку кожного учасника руху. Знаючи, ці правила шофер повинен діяти за певним алгоритмом.

3 Масовий випуск автомобілів став можливий тільки тоді, коли був придуманий порядок складання машини на конвеєрі. Певний порядок складання автомобілів - це набір дій, в результаті яких виходить автомобіль.

Існує кілька способів запису алгоритмів. На практиці найбільш поширеним є такі форми подання алгоритмів:

1 словесна (запис на природній мові);
2 псевдокоду (напівформалізоване опису алгоритмів
3 графічна (зображення з графічних символів - блок- схема);
4 програмна (тексти на мовах програмування - код програми).

Розрізняють три основних види алгоритмів:

- лінійний алгоритм,
- розгалужується алгоритм.
- циклічний алгоритм.

Лінійний алгоритм - це алгоритм, в якому дії виконуються одноразово і строго послідовно.

Розгалужується алгоритм - це алгоритм, в якому в залежності від умови виконується або одна, або інша послідовність дій.

Циклічний алгоритм - це алгоритм, команди якого повторюються кілька разів поспіль.

2.2 Блок-схеми. Графічна реалізація алгоритмів

Блок-схеми являють собою наочну реалізацію алгоритму. Поняття блок-схеми. Основні види блоків

Блок-схема - его графічна реалізація алгоритму.

Блок-схема являє собою зручний і наочний спосіб запису алгоритму.

Блок-схема складається з функціональних блоків різної форми, пов'язаних між собою стрілками. У кожному блоці описується одне або кілька дій.

Короткі підсумки

Будь-яке завдання може бути розбита на елементарні дії. Для будь-якої математичної задачі або ситуації з життя можна скласти алгоритм рішення. Алгоритм може бути описаний словесно, псевдокодом, графічно або програмно. Завдання завжди вирішується за допомогою базових типів алгоритму - лінійного, разветвляющегося або циклічного.

Лекція 3 «Синтаксис, семантика і розробка програм на С ++»

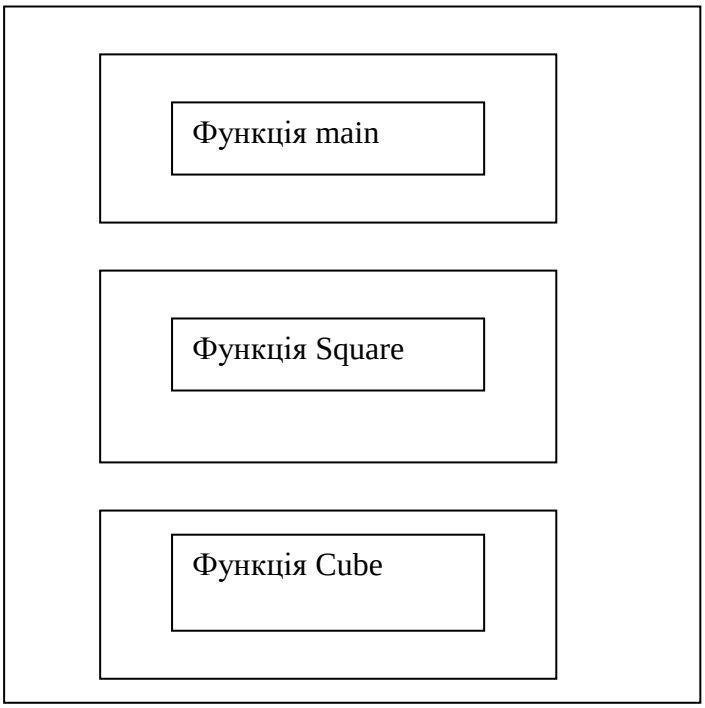
План

1 Елементи програм на С ++
- Структура програми
2 Синтаксис і семантика
3 Дані і їх типи

Елементи програм на С ++

У цій лекції розглядаються правила і символи, з яких складається мова програмування С ++, а також описуються кроки, необхідні для створення і виконання програми.

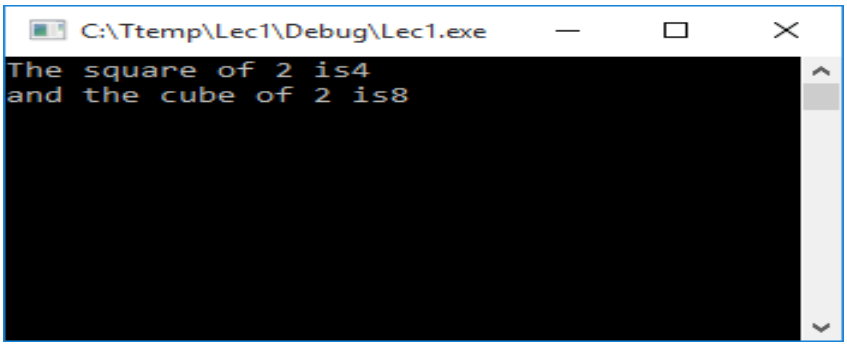
Структура програми



У мові С ++ підпрограми мають називу функції (functions), а програма на С ++ являє собою набір з однієї або більш функцій. Кожна функція виконує певну дію, а всі разом вони вирішують завдання в цілому.

```
#include "stdafx.h"
#include "conio.h"
#include "math.h"
#include "iostream"

using std::cout;
using std::endl;

int Square (int);
int Cube (int);


int main()
{
	cout << "The square of 2 is" << Square(2) << endl;
	cout << "and the cube of 2 is" << Cube(2) << endl;
	_getch();
	return 0;
}

int Square(int n)
{
	return n*n;
}

int Cube(int n)
{
	return n*n*n;
}
/*
Складний коментар
/*
```

Таким чином, повністю висновок цієї програми виглядає наступним чином:



Синтаксис і семантика

Синтаксичні правила - це набір правил, відповідно до яких з команд будується програма. Ці правила дозволяють вибрати елементи мови програмування (основні будівельні блоки мови) і зібрати з них конструкції, тобто синтаксично правильні мовні структури. Якщо текст програми порушує якесь правило мови, скажімо, через описки в ключовому слові або через пропуску коми, кажуть, що вона містить синтаксичні помилки. У цьому випадку програма не може бути правильно відкомпільована, поки помилки не виправлені.

Семантика (semantics), в свою чергу, визначає значення команд мови програмування.

Синтаксичні шаблони

Синтаксичні правила мови C ++ можна зобразити за допомогою синтаксичних шаблонів (syntax template). Синтаксичний шаблон - це узагальнений приклад визначається конструкції C ++. Необов'язкові частини конструкції і ті частини, які можуть повторюватися, виділяються графічно відповідно до визначених угодами. Слово або символ, виділений жирним шрифтом, є буквальне (зарезервоване) слово або символ мови C ++.

Ідентифікатори

Елементи програм це ідентифікатори

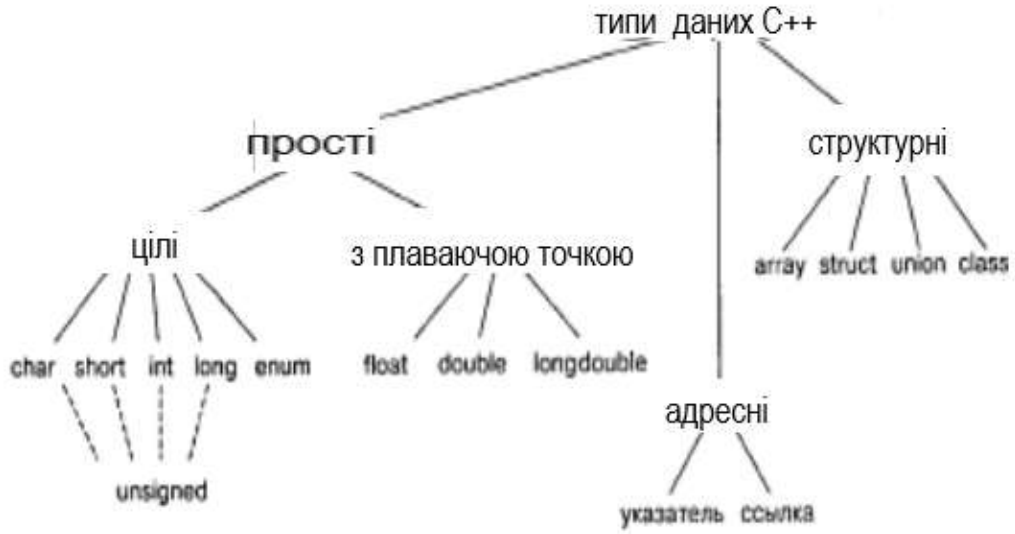
Ідентифікатори (identificators) використовуються в C ++ для іменування різних об'єктів. Ідентифікатори складаються з літер (A -Z, a -z), цифр (0 -9) і символів підкреслення (_), але при цьому повинні починатися тільки з букви або з підкреслення.

В якості імен ідентифікаторів можна використовувати зарезервовані імена

C ++ є мовою, що враховує регістр символів. Іншими словами, великі літери відрізняються в C ++ від малих літер.

Дані та їх типи

Дані, необхідні для роботи програми, зберігаються в пам'яті комп'ютера. Кожна область пам'яті має однозначно певну адресу, на який посилаються, коли необхідно зберегти або прочитати дані. Адреса розташування даних в пам'яті - це двійкове число в машинному коді. У мову е С ++ ідентифікатори використовуються дл я того, щоб назвати ту чи іншу область пам'яті, а компілятор транслює імена у відповідні адреси. У цьому полягає одна з переваг мови програмування високого рівня, що звільняє програміста від необхідності відстежувати дійсне розташування даних і команд в пам'яті комп'ютера. У С ++ кожен елемент даних повинен належати до будь - якого певного типу даних (data type). Тип визначає, в якому вигляді дані представлені в комп'ютері, а також які перетворення комп'ютер може до них застосовувати



Типи даних мови С ++

Лекція 4 «Основні операції»

План

1 Описи змінних
2 Основні арифметичні операції
3 Побудова програми
4 Препроцесор
5 Арифметичні вирази

Описи змінних

INT

Розмір знакового або беззнакового елемента int відповідає стандартному розміру цілочисельного значення на конкретному комп'ютері. Наприклад, в 16-розрядних операційних системах тип int зазвичай має розмір 16 біт, або 2 байта. У 32-розрядних операційних системах тип int зазвичай має розмір 32 біта, або 4 байта. Таким чином, в залежності від цільової середовища тип int еквівалентний типу short int або long int, а тип unsigned int - типу unsigned short або unsigned long. Всі типи int представляють знакові значення, якщо не вказано інше.

FLOAT

Числа з плаваючою комою використовують формат IEEE (Інституту інженерів з електротехніки та електроніки). Значення з одиночної точністю і типом float мають 4 байта, складаються з 1 біта знака, 8-розрядної двійковій експоненти і 23-бітної мантиси. Мантиса представляє число від 1,0 до 2,0. Оскільки біт вищого порядку мантиси завжди дорівнює 1, він не зберігається в числі. Це уявлення забезпечує для типу float діапазон приблизно від 3,4E-38 до 3,4E + 38.

```
int  num;  (4 байта)
int  alpha;
float   rate;   (4 байта) (1 +3(-1бит)
char  ch;     (1 байт)
bool log;    (1 байт)
```

Приклади

```
alpha  = 2856;
rate  = 0.36;
ch  = 'B';
num  = alpha;
```

Основні арифметичні операції

Бінарні операції:

- + – бінарний плюс.
- - – бінарний минус.
- + – додавання.
- - – віднімання
- * – множення.
- / – ділення (при розподілі двох цілих чисел виходить ціла частина від приватного).
- % – отримання залишку від ділення.

Операції присвоювання:

- = – привласнити операнду з лівої частини значення виразу з правої частини.
- += – привласнити операнду з лівої частини суму операндів лівої і правої частин.
- -= – привласнити операнду з лівої частини різницю операндів лівої і правої частин..
- /= – привласнення частки від розподілу.
- %= – привласнення залишку від ділення.

Операції порівняння:

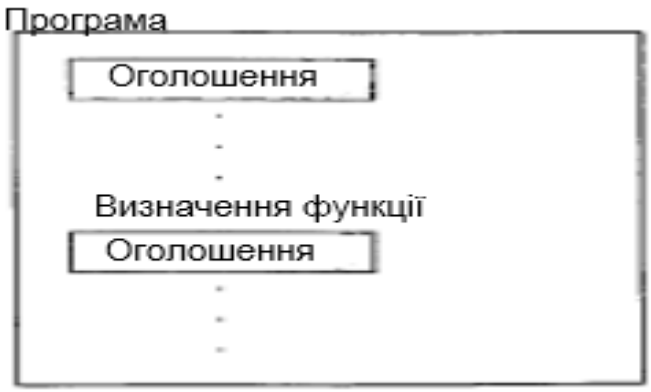
- < – менше.
- > – більше.
- <= – менше або дорівнює.
- >= – більше або дорівнює.
- == – дорівнює
- != – не дорівнює.

Логичні бінарні операції:

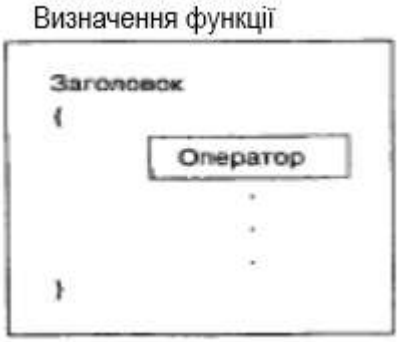
- && – логічне І.
- || – логічне ІЛІ

Побудова програми

Програма на С ++ складається з функції одна з яких повинна називатися main. Програма також може містити оголошення, які не включені ні в одну з функції. Синтаксичний шаблон програми



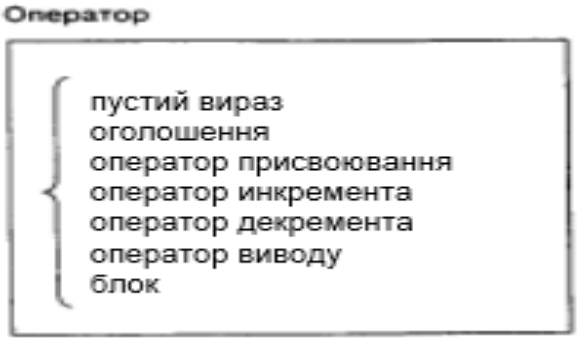
Визначення функції складається із заголовка і тіла функції, яке обмежене правою і лівою фігурними дужками :



Блоки або складові виразу

Тіло функції є прикладом блоку або складеного оператора

Наведемо синтаксичний шаблон оператора, що включає тільки оператори С ++:



Препроцесор

Поняття препроцесора є одним з ключових в мову е С ++. Препроцесор - це програма, що діє як фільтр на етапі компіляції. Перед тим, як потрапити на вхід компілятора, вихідна програма проходить через препроцесор Рядок, що починається символом решітки (#), не є виразом мови С ++ (і тому не закінчується крапкою з комою). Вони називаються директивою препроцесора (preprocessor directive)

Лістінг 4.1

```
#include "stdafx.h"

#include "conio.h"
#include "math.h"
#include "iostream"
```

Препроцесор розширює директиву #include, вставляючи вміст зазначеного файлу безпосередньо у вихідну програму.

Файли, які з'являються в директиві #include, зазвичай зазвичай мають розширення. h, що означає файл заголовків (head erfile). Файли заголовка також містять оголошення констант, змінних і функцій, необхідних для роботи програми.

Відзначимо, що в директиві використовують- <> Вони вказують препроцесору, що цей файл шукається в стандартному каталозі підключаються файлів (includ e directory), тобто в місці, де комп'ютерна система зберігає всі доступні для використання файли заголовків.

Арифметичні вирази

Розглянемо більш складні вирази, що складаються з декількох операцій які містять різні типи даних.

П'ять арифметичних операцій, а саме: додавання (+), віднімання (-), множення (*), ділення (/) і обчислення залишку (%), поряд з круглими дужками, впорядковані так само, як і відповідні математичні дії.

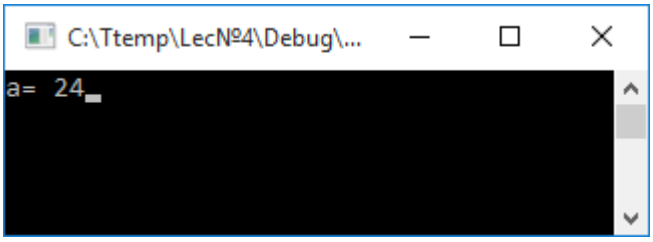
Типи даних можна змішувати в арифметичних виразах

Коженого разу, коли ціле значення і значення з плаваючою точкою є операндами будь-якої бінарної арифметичної операції, відбувається неявне перетворення типів за наступною схемою.

1. Ціле число тимчасово перетвориться в число з плаваючою крапкою.
2. Виконується арифметична операція.
3. Результатом операції стає число з плаваючою крапкою.

Лістінг 4.2

```
int main()
{
	int	a = 20;
	float	b = 4.0;
	int	c = 6;
	a = a + b;
	cout << "a= "<<a;
	_getch();
	return 0;
	}
```



Лекція 5 «Керуючі структури»

План

1 Потік управління

2 Умови та логічні вирази

3 Умовний оператор

4 Вкладені умовні оператори

Потік управління

Порядок, в якому вираження виконуються в програмі, називається потоком управління (flow of control). У певному сенсі комп'ютер управляється єдиним виразом в кожен певний момент часу. Коли воно виконано, управління переходить до наступного виразу подібно до того, як передається естафетна паличка.

Потік управління зазвичай є послідовним. У місцях, де потрібно змінити напрямок потоку управління (зробити його непослідовним), використовуються керуючі структури (control structures). Це спеціальні оператори, які передають управління висловом, яка не є таким по порядку

Умови та логічні вирази

Якщо необхідно сформулювати твердження, що є істинним чи хибним. Комп'ютер оцінює (evaluate) це твердження, перевіряючи його на відповідність деякому внутрішньому умові (скажімо, значенням деякої змінної) і визначає істинність або хибність твердження

Логічні вирази

У мові С ++ твердження мають форму логічних (logical) виразів, які називаються також булеві (Boolean). Такі вирази складаються з логічних значень і операцій.

Операції порівняння

Одним із способів присвоєння значень логічним змінним є застосування оператора присвоювання:

```
Bool Start;
Start = true;
```

Логічні операції

До логічних операцій відносять операції
"AND" - &&
"OR" - ||
"NOT"!
С правої і лівої сторони можуть бути вирази або змінний (в простому випадку)
З логічних операцій можна будувати більш складні конструкції, які можуть містити більше однієї логічної операції

Рішення складних логічних виразів залежить від пріоритету операцій
1!
2 ==,! =
3 &&
4 ||

Логічний вираз можна перетворити, використовуючи їх еквівалентність
(Для цього необхідно звернутися до булевої алгебри)

приклад:
! (A == b) еквівалентно a! = B

3 Умовний оператор

Дізнавшись, як записувати логічні вирази, їх можна застосувати для зміни лінійного потоку управління в програмі.

Оператор If, або умовний (relational) оператор, дозволяє ветвіти потік управління. Інакше кажучи, за допомогою оператора If можна задати по ходу виконання програми якесь питання і в залежності від відповіді на нього виконати ті чи інші дії.

На псевдокоді це запишеться наступним чином:
IF (якщо) деякий умова вірна
THEN (тоді) виконати певну дію
ELSE (інакше) виконати деякі інші дії

Умовний оператор в формі If-Then-Else

У С ++ умовний оператор має дві форми: If-Then- Else і If-Then. Спочатку розглянемо його представлення у вигляді If-Then-Else. Нижче наведено синтаксичний шаблон для цієї форми:

If (вираз)

оператор 1А

Else

оператор 1Б

Вираз в дужках може бути будь-якого простого типу (майже завжди це логічний вираз). Якщо цей вислів має нульове значення (true), комп'ютер виконує оператор 1 А. При нульовому значенні умови (false) виконується оператор 1Б. Оператор 1А іноді називають then-твердженням (then clause), оператор 1Б - else -твердження (else-clause).

Умовний оператор в формі If-Then

Іноді можна зіткнутися з ситуацією, яка описується так: «Якщо якийсь умова вірна, то потрібно виконати певну дію; в іншому випадку ніяких дій робити не треба ». Цього можна досягти за допомогою порожнього оператора в else- затвердження:

If (умова)

оператор 1А

Тобто в разі, коли вираз в дужках «помилково» пропускається Оператор 1А або блок операторів (якщо він присутній) в фігурних дужках і виконується оператор, наступний за Оператором 1А або блоком

Вкладені умовні оператори

У С ++ не існує обмежень на тип виразів, що входять в умовний оператор. Отже, всередині одного оператора If може перебувати інший оператор If. Єдиним обмеженням є те, що програмісту важко розібратися в керуючої структурі, якщо вона надто громіздка. Не забувайте при цьому, що легкість сприйняття - це один з характерних ознак гарної програми.

При розміщенні If всередині If створюється так звана вкладена керуюча структура (nested control structure). Таке «гніздо» керуючих структур нагадує набір салатниць різного розміру, в якому менші за розміром миски вкладені в великі.

Розглянемо типовий приклад, записаний на псевдокоді.

```
IF today is Saturday or Sunday

IF it is raining
Continue to sleep
ELSE
Get up and go outside

ELSE
Go to work
```

Будь-яке завдання, яка включає в себе множину розгалуження (multi - way branch), тобто вибір з безлічі альтернативних дій, може бути запрограмована за допомогою вкладених умовних операторів. Наприклад, щоб надрукувати назву місяця по його заданому порядковому номеру, допустимо використовувати наступну послідовність умовних операторів

Лекція 6 «Введення і оформлення тексту»

План

1. Цикл for

2. Цикл while

3. Цикл do..while

4. Оператор break

5. Інструкція goto

6. Одномірні масиви

7. Сортування масиву

Цикл for

Оператор for служить для створення циклу. Він повинен виглядати так:

```
for (вираз ініціалізації; вираз умови; вираз циклу)
{
  оператори
}
```

Вираз ініціалізації зазвичай служить для завдання початкового значення лічильника циклу. Вираз умови дозволяє припинити виконувати цикл, коли умова перестане виконуватися, тобто прийме значення false. Вираз циклу зазвичай здійснює інкремент або декремент лічильника циклу. Будь-яке з цих виразів може бути пропущено, але відповідна точка з коми повинна стояти.

Цикл while

Оператор while схожий з оператором for, але він не виконує ініціалізацію і інкремент лічильника в своєму оголошенні. Синтаксис цього оператора наступний:

```
while (вираз умови)
{
  оператори
}
```

Якщо вираз умови в циклі while відразу помилково, то оператори не виконаються жодного разу.

Цикл do..while

Оператор do..while практично ідентичний оператору while, але, оскільки в ньому перевірка умови здійснюється в кінці, він гарантує виконання операторів принаймні один раз:

```
do
{
  оператори
} While (вираз умови)
```

Оператор break

За допомогою інструкції break можна організувати негайний вихід з циклу,опустивши виконання коду, що залишився в його тілі, і перевірку умовного виразу. При виявленні всередині циклу інструкції break цикл завершується, а управління передається інструкції, наступної, після циклу

Інструкція goto

Ця інструкція перебувала в немилості у програмістів, оскільки сприяла, з їх точки зору, створення "спагетті-коду". Однак інструкція goto як і раніше використовується, і іноді навіть дуже ефективно. Однак в будь-якій ситуації (в області програмування) можна обійтися без інструкції goto, оскільки вона не є елементом, що забезпечує повноту опису мови програмування. Разом з тим, в певних ситуаціях її використання може бути дуже корисним

Інструкція goto вимагає наявності в програмі мітки. Мітка - це дійсний в C ++ ідентифікатор, за яким поставлено двокрапка. При виконанні інструкції goto управління програмою передається інструкції, зазначеної за допомогою мітки. Мітка повинна знаходитися в одній функції з інструкцією goto, яка посилається на цю мітку. Мітка - це ідентифікатор, за яким стоїть двокрапка.

Одномірні масиви

Одновимірна масив - це список пов'язаних змінних.

Для оголошення одновимірного масиву використовується наступна форма запису.

тип імя_массіва [розмір];

Тут за допомогою елемента запису тип оголошується базовий тип масиву. Базовий тип визначає тип даних кожного елемента, що становить масив. Кількість елементів, які будуть зберігатися в масиві, визначається елементом розмір.

Наприклад, при виконанні наведеної нижче інструкції оголошується int-масив (що складається з 10 елементів) з ім'ям mas,

int mas [10];

Індекс ідентифікує конкретний елемент масиву.

Доступ до окремого елементу масиву здійснюється за допомогою індексу. Індекс описує позицію елемента всередині масиву. У С ++ перший елемент масиву має нульовий індекс. Оскільки масив mas містить 10 елементів, його індекси змінюються від 0 до 9. Щоб отримати доступ до елементу масиву за індексом, досить

Елементи масиву в пам'яті розташовані послідовно один за одним. Осередок з найменшим адресою відноситься до першого елементу масиву, а з найбільшим - до останнього. Схематично це можна показати так

У С ++ можна привласнити один масив іншому. У наступному фрагменті коду,

наприклад, присвоювання a = b; неприпустимо.

Вся відповідальність за дотримання меж масивів лежить тільки на програмістів, які повинні гарантувати коректну роботу з масивами. Іншими словами, програміст зобов'язаний використовувати масиви досить великого розміру, щоб в них можна було без ускладнень поміщати дані, але найкраще в програмі передбачити перевірку перетину кордонів масивів.

Сортування масиву

Однією з найпоширеніших операцій, виконуваних над масивами, є сортування. Існує безліч різних алгоритмів сортування. Широко застосовується, наприклад, сортування перемішуванням і сортування методом Шелла. Відомий також алгоритм Quicksort (швидке сортування з розбивкою вихідного набору даних на дві половини так, що будь-який елемент першої половини впорядкований щодо будь-якого елементу другої половини). Однак найпростішим вважається алгоритм сортування бульбашковим методом. Незважаючи на те що бульбашкова сортування не відрізняється високою ефективністю (і справді, його продуктивність неприйнятна для сортування великих масивів), його цілком успішно можна застосовувати для сортування масивів малого розміру.

Алгоритм сортування бульбашковим методом отримав свою назву від способу, використовуваного для упорядкування елементів масиву. Тут виконуються повторювані операції порівняння і при необхідності міняються місцями суміжні елементи. При цьому елементи з меншими значеннями поступово переміщуються до одного кінця масиву, а елементи з великими значеннями - до іншого. Цей процес нагадує поведінку бульбашок повітря в резервуарі з водою. Бульбашкова сортування виконується шляхом декількох проходів по масиву, під час яких при необхідності здійснюється перестановка елементів, які опинилися "не на своєму місці". Кількість проходів, які гарантують отримання відсортованого масиву, дорівнює кількості елементів в масиві, зменшеному на одиницю

Лекція 7 «Рядки. Функції..»

План

1 Рядки

2 Функції.

3 Функції бібліотеки cstring

Рядки

Найчастіше одномірні масиви використовуються для створення символьних рядків. У С ++ рядок визначається як символьний масив, який завершується нульовим символом ('\ 0'). При визначенні довжини символьного масиву необхідно враховувати ознака її завершення і задавати його довжину на одиницю більше довжини найбільшої рядки з тих, які передбачається зберігати в цьому масиві.

Наприклад, оголошуючи масив str, призначений для зберігання 10-символьного рядка, слід використовувати таку інструкцію.

```
char str [11];
```

Заданий тут розмір (11) дозволяє зарезервувати місце для нульового символу в кінці рядка.

Приклад строкових літералів (строковий літерал - це список символів, укладений в подвійні лапки.)

"STUDY"

""

Останній приклад, це приклад нульовий рядки останньої (""), називається нульовою. Вона складається тільки з одного нульового символу (ознаки завершення рядка). Нульові рядки використовуються для подання порожніх рядків.

Програмісту немає необхідності вручну додавати в кінець строкових констант нульові символи. С ++ - компілятор робить це автоматично.

S	T	U	D	Y	'\0'

Функції.

Будь-яка С ++ програма складається з "будівельних блоків", іменованих функціями. Функція-це підпрограма, яка містить одну або кілька С ++ - інструкцій і виконує одну або кілька завдань. Хороший стиль програмування на С ++ передбачає, що кожна функція виконує тільки одну задачу. Кожна функція має ім'я, яке використовується для її виклику. Своїми функціями програміст може давати будь-які імена за винятком імені main (), зарезервованого для функції, з якої починається виконання програми.

У С ++ жодна функція не може бути вбудована в іншу. На відміну від таких мов програмування, як Pascal, і деяких інших, які дозволяють використання вкладених функцій, в С ++ всі функції розглядаються як окремі компоненти. (Безумовно, одна функція може викликати іншу.)

Ім'я функції завершується парою круглих дужок. Наприклад, якщо функція має ім'я get, то її згадка в тексті позначиться як get (). Таким чином її легко відрізнити від змінних

Функція, що не поверта свої значення -Void

Прототип функції оголошує функцію до її визначення. Прототип дозволяє компілятору дізнатися тип значення, що повертається цією функцією, а також кількість і тип параметрів, які вона може мати. Компілятору потрібно знати цю інформацію до першого виклику функції. Тому прототип розташовується до функції main ().

Єдиною функцією, яка не вимагає прототипу, є main (), оскільки вона вбудована в мову C ++. Як бачите, функція myfunc () не містить інструкцію return. Ключове слово void, яке передує як прототип, так і визначення функції myfunc (), формально заявляє про те, що функція myfunc () не повертає ніякого значення.

```
void myfunc();// прототип функції myfunc()

int main()
{
  myfunc();
  myfunc();
  cout << "         Welcome" << endl;
  myfunc();
  myfunc();


_getch();


  return 0;
}

void myfunc()
{

  cout << "............................." << endl;

}
```

Аргументи функцій

Функції можна передати одне або кілька значень. Значення, що передається функції, називається аргументом.

Верхня межа числа прийнятих аргументів визначається конкретним компілятором. Відповідно до стандарту C ++ він дорівнює 256.

Аргумент - це значення, яке передається функції при виклику.

При створенні функції, яка приймає один або кілька аргументів, іноді необхідно оголосити змінні, які будуть зберігати значення аргументів. Ці змінні називаються параметрами функції.

Параметр - це обумовлена функцією змінна, яка приймає переданий функції аргумент.

Змінна, яка приймає аргумент, називається параметром. Функції, які приймають аргументи, називаються параметризрвані функціями.

Функції, які повертають значення

У С ++ багато бібліотечних функцій повертають значення. Наприклад, вже знакома функція abs () повертає абсолютну значення свого аргументу. Функції, написані програмістом, також можуть повертати значення.

В С ++ для повертання значення використовується інструкція return. Загальний формат цього інструкції такий:

Return значення

Функції бібліотеки cstring

Функція strlen ()

Загальний формат виклику функції strlen () такова:

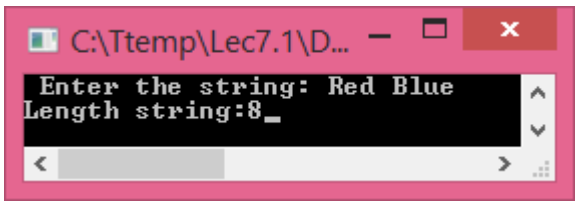
strlen (s);

Тут s- строка. Функція strlen () повертає довжину рядка, вказаного аргументом s.

При виконанні наступної програми буде показано довжина рядка, введена з клавіатури.

Лістінг 7.1

```
int main()
{
        char str[80];

        cout << " Enter the string: ";
        cin.getline(str, 80);
        cout <<"Length string:"<< strlen(str);
        _getch();
        return 0;
        }
```



Функция strcmp ()

Звернення до функції strcmp () має наступний формат:

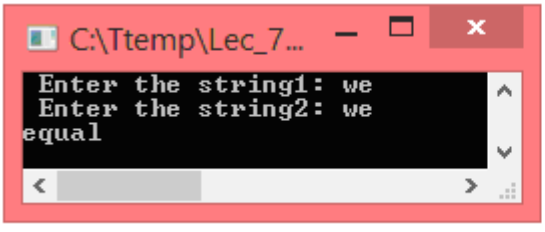
strcmp (s1, s2);

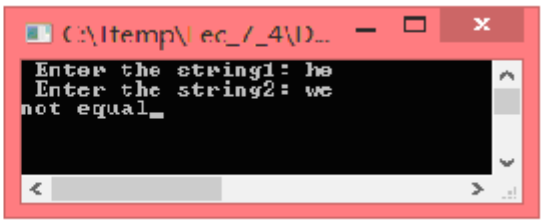
Функція strcmp () порівнює строку s2 з рядком s1 і повертає значення 0, якщо вони рівні. Если строка s1 лексикографически (тобто відповідно до алфавітного порядку) більше рядка s2, повертається позитивний номер. Якщо строка s1 лексикографически меньше строки s2, то негативне число повертається.

Використання функцій strcat () демонстрируется в наступній програмі

Лістінг 7.2

```
int main()
{

	char str1[80];
	char str2[80];

	cout << " Enter the string1: ";
	cin.getline(str1, 80);   // Считываем строку1 с клавиатуры.

	cout << " Enter the string2: ";
	cin.getline(str2, 80);   // Считываем строку2 с клавиатуры.

	//cout << " Here is your strings: " << endl;
	//cout << str1;
	// cout << str2;

	if (strcmp(str1, str2))
		cout << "not equal";
	else cout << "equal";


	_getch();
	return 0;
	}
```





Функція strcat ()

Звернення к функциям strcat () має наступний формат.

Strcat (s1, s2);

Функція strcat () присоединяет строку s2 до кінця рядка s1 при цьому рядку s2 не змінюється. Строки повинні завершуватися нульовим символом.

Функція strcpy ()

Загальний формат виклику функції strcpy () так:

strcpy (до, з);

Функція strcpy () копіює вміст рядка з в рядку до. Помните, що масив, що використовується для зберігання рядків, має бути достатньо великим, щоб в нього було можливо помістити рядок із масиву з. В противному випадку масив до переполнится, т.е. відбувається вихід за його кордони, що може призвести до руйнування програми.

Лекція 8 «N-вимірні масиви

1 Двувимірні масиви
2 Багатовимірні масиви
3 Безрозмірна ініціалізація масивів
4 Массиви строк

Двувимірні масиви

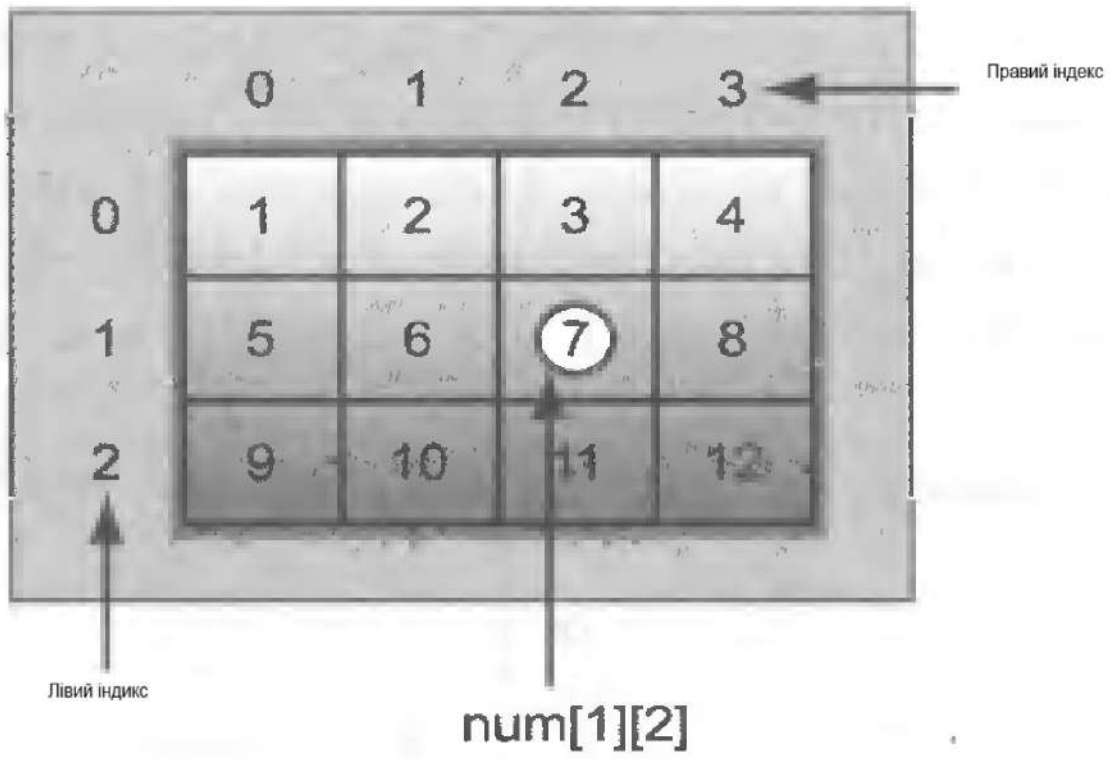
В С ++ можна використовувати багатомірні масиви. Найпростіший багатомірний масив - двомірний. Двувимірний масив, по суті, представляє собою список одновимірних масивів. Щоб оголосити двовимірний масив цільнозначних значень розміром 10x20 з ім'ям mas, досить записати наступне:

```
int mas [10] [20];
```

В відмінність від багатьох інших програм програмування, в яких при оголошенні масиву значення розмірів розділяються зап'ятими, в С ++ кожна розмірність складається в власній пари квадратних скобок.

Щоб отримати доступ до елементу масиву з координатами 3,5, необхідно використовувати запис мас [3] [5].

У двувимірному массиві позиция любого елементу визначається двома індексами. Якщо уявити двувимірний массив у вигляді таблицы даних, то один індекс означає строку, а другий — стовбець. З цього виходить, що, якщо доступ до елементів масиву надати в порядку, в якому вони реально зберігаються в пам'яті, то правий індекс змінюватиметься швидше, ніж лівий.



Необхідно пам'ятати, що місце зберігання для усіх елементів масиву визначається під час компіляції. Крім того, пам'ять, виділена для зберігання масиву, використовується впродовж усього часу існування масиву. Для визначення кількості байтів пам'яті, займаної двовимірним масивом, використайте наступну формулу: число байтів = число рядків x число стовпців x розмір типу у байтах

Багатовимірні масиви

В C++, окрім двовимірних, можна визначати масиви трьох і більше вимірів. От як оголошується багатовимірний масив. тип ім'я [розмір 1] [розмір 2] ...[розмір N] ;

Наприклад, за допомогою наступного оголошення створюється тривимірний цілочисельний масив розміром 4x10x3. int multidim[4][10][3]; Як згадувалося вище, пам'ять, виділена для зберігання усіх елементів масиву, використовується впродовж усього часу існування масиву. Масиви з числом вимірів, що перевищує три, використовуються нечасто, хоч би тому, що для їх зберігання потрібно великий об'єм пам'яті

Ініціалізація масивів

В C++ передбачена можливість ініціалізації масивів. Формат ініціалізації масивів подібний до формату ініціалізації інших змінних.

тип ім’я_масива [розмір] = { список_значень};

Тут елемент список_значений є списком значень ініціалізації елементів масиву, розділених комами. Тип кожного значення ініціалізації має бути сумісний з базовим типом масиву (елементом тип). Перше значення ініціалізації буде збережено в першій позиції масиву, друге значення - в другій і так далі. Зверніть увагу на те, що крапка з комою ставиться після закриваючої фігурної дужки (}).

Для символьних масивів, призначених для зберігання рядків, передбачений скорочений варіант ініціалізації, який має таку форму.

char имя_массива[розмір] = "рядок";

Наприклад, наступний фрагмент коду ініціалізував масив str фразою "привіт,

char str[7] = "привіт";

Це рівнозначно поелементній ініціалізації.

char str[7] = {'п', 'р', 'и', 'в', 'і', 'т', '0'};

Багатовимірні масиви ініціалізувалися по аналогії з одновимірними. Наприклад, в наступному фрагменті програми масив sqrs ініціалізувався числами від 1 до 10 і квадратами цих чисел.

Лістінг 8.1

```
int sqrs[10][2] = {
                        1, 1,
                        2, 4,
                        3, 9,
                        4, 16,
                        5, 25,
                        6, 36,
                        7, 49,
                        8, 64,
                        9, 81,
                        10, 100
                    };
```

При ініціалізації багатовимірного масиву список ініціалізаторів кожної розмірності (підгрупу ініціалізаторів) можна взяти у фігурних дужок. Ось, наприклад, як виглядає ще один варіант запису попереднього оголошення.

Лістінг 8.2

```
int sqrs[10][2] = {
                        {1, 1},
                        {2, 4},
                        {3, 9},
                        {4, 16},
                        {5, 25},
                        {6, 36},
                        {7, 49},
                        {8, 64},
                        {9, 81},
                        {10 , 100}
                    };
```

Безрозмірна ініціалізація масивів

Припустимо, що ми використовуємо наступний варіант ініціалізації масивів для побудови таблиці повідомлень про помилки.

```
char err1[13] = " Ділення на 0\n";
char err2[12] = "Кінець файла\n";
char err3[19] = " У доступі відмовлено \n";
```

Неважко припустити, що вручну незручно підраховувати символи в кожному повідомленні, щоб визначити коректний розмір масиву. На щастя, в C++ передбачена можливість автоматичного визначення довжини масивів шляхом використання їх "безрозмірного" формату. Якщо в інструкції ініціалізації масиву не вказаний його розмір, C++ автоматично створить масив, розмір якого буде достатнім для зберігання усіх значень ініціалізаторів. При такому підході попередній варіант ініціалізації масивів для побудови таблиці повідомлень про помилки можна переписати так.

```
char err1[] = " Ділення на 0\n";
char err2[] = "Кінець файла\n";
char err3[] = " У доступі відмовлено \n";
```

Окрім зручності в первинному визначенні масивів, метод "безрозмірної" ініціалізації дозволяє змінити будь-яке повідомлення без перерахунку його довжини. Тим самим усувається можливість внесення помилок, викликаних випадковим прорахунком. "Безрозмірна" ініціалізація масивів не обмежується одновимірними масивами. При ініціалізації багатовимірних масивів вам необхідно вказати усі дані, за винятком крайньої ліворуч розмірності, щоб З++-компилятор міг належним чином індексувати масив. Використовуючи "безрозмірну" ініціалізацію масивів, можна створювати таблиці різної довжини, дозволяючи компілятору автоматично виділяти область пам'яті, достатню для їх зберігання.

Масиви рядків

Існує спеціальна форма двовимірного символьного масиву, яка є масивом рядків. У використанні масивів рядків немає нічого незвичайного. Наприклад, в програмуванні баз даних для з'ясування коректності команд, що вводяться користувачем, вхідні дані порівнюються з вмістом масиву рядків, в якому записані допустимі в цьому застосуванні команди. Для створення масиву рядків використовується двовимірний символьний масив, в якому розмір лівого індексу визначає кількість рядків, а розмір правого - максимальну довжину кожного рядка. Наприклад, при виконанні наступної інструкції оголошується масив, призначений для зберігання 30 рядків завдовжки 80 символів.

```
char str _ array[30][80];
```

Масив рядків - це спеціальна форма двовимірного масиву символів.

Отримати доступ до окремого рядка досить просто: досить вказати тільки лівий індекс. Наприклад, наступна інструкція викликає функцію gets () для запису третього рядка масиву

```
str _ array. gets (str _ array[2]);
```

Лекція 9 «Покажчики»

1 Покажчики і змінні
2 Покажчики і масиви
3. Покажчики і рядки

Покажчики і змінні

Покажчики, без сумніву, - один з найважливіших і складніших аспектів C++. Значною мірою потужність багатьох засобів C++ визначається використанням покажчиків. Наприклад, завдяки ним забезпечується підтримка зв'язних списків і динамічного виділення пам'яті, і саме вони дозволяють функціям змінювати вміст своїх аргументів.

При розгляді теми покажчиків доведеться використати такі поняття, як розмір базових C++-типов даних. Треба припустити, що символи займають в пам'яті один байт, цілочисельні значення - чотири, значення з плаваючою точкою типу float - чотири, а значення з плаваючою точкою типу double - вісім (ці розміри характерні для типового 32-розрядного середовища).

Оператори, використовувані з покажчиками

З покажчиками використовуються два оператори : "&" і "*". Оператор "&" - унарний. Він повертає адресу пам'яті, по якій розташований його операнд. Другий оператор роботи з покажчиками (*) служить доповненням до першого (&). (*)

Покажчик - це змінна, яка містить адресу іншої змінної. Покажчики - це змінні, які зберігають адреси пам'яті. Найчастіше ці адреси означають місце розташування в пам'яті інших змінних.

Наприклад, якщо x містить адресу змінної y, то про змінну, x говорять, що вона "вказує" на y.

Змінні-покажчики (чи змінні типу покажчик) мають бути відповідно оголошені. Формат оголошення змінної-покажчика такий:

тип * ім'я _ змінної;

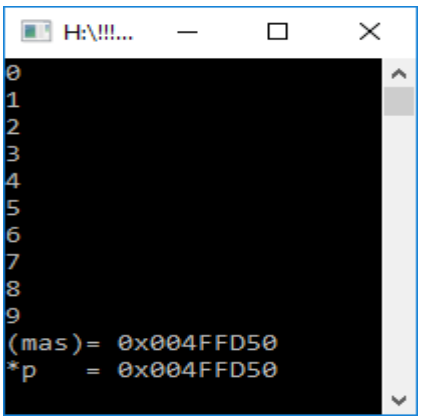
Тут елемент тип означає базовий тип покажчика, причому він має бути допустимим З++-типом. Елементом ім'я __ змінної є ім'я змінної-покажчика.

Покажчики і масиви

У покажчику зберігається адреса першого елементу масиву (перевіримо)

Приклад 9.1

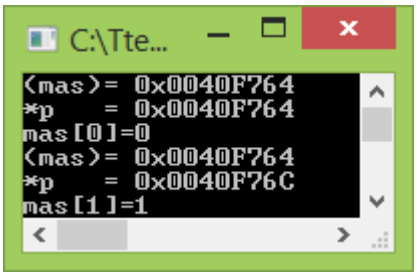
```
int main()
{
	double mas[10];
	double *p = mas;

	for (int i = 0; i < 10; i++)
	{
		mas[i] = (double)i;
		cout << i << endl;
	}
	cout << "(mas)= 0x" << mas << endl;
	cout << "*p   = 0x" << p << endl;

	_getch();
	return 0;
}
```



Виведення першого і другого елементу масиву за допомогою покажчика (разимєнованіе) і його адреси

Приклад 9.2

```
int main()
{
	double mas [10];
	double *p = mas;

	for (int i = 0; i < 10; i++)
		mas[i] = (double)i;

	cout << "(mas)= 0x" << mas << endl;
	cout << "*p   = 0x" << p << endl;
    cout << "mas[0]=" << *p << endl;
	p++;
	cout << "(mas)= 0x" << mas << endl;
	cout << "*p   = 0x" << p << endl;
	cout << "mas[1]=" << *p << endl;

	_getch();
return 0;
}
```

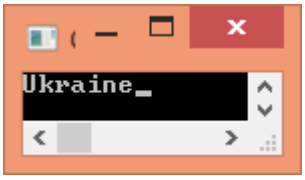


Покажчики і рядки

Рядок це масив, тому з рядками покажчики працюють як і з масивами (у покажчик записується адреса початку рядка) Тому при виведенні рядка досить вказати за допомогою покажчика початкову адресу рядка, що виводиться, і далі буде виводиться рядок до ознаки кінця рядка

Приклад 9.3

```
int main()
{

    char *p ="Ukraine";
    cout << p;

       _getch();
return 0;
}
```



Лекція 10 «Порозрядні оператори, Робота з файлами»

План

1 Порозрядні оператори

2 Оператори зсуву

3 Робота з файлами

Порозрядні оператори

Оскільки C++ націлений на те, щоб дозволити повний доступ до апаратних засобів комп'ютера, важливо, щоб він мав можливість безпосередньо впливати на окремі біти у рамках байта або машинного слова. Саме тому C++ і містить порозрядні оператори. Порозрядні оператори призначені для тестування, установки або зрушення реальних бітів у байтах або словах, які відповідають символьним або цілочисельним З++-типам. Порозрядні оператори не використовуються для операндів типу bool, float, double, long double, void або інших ще складніших типів даних. Порозрядні оператори (вони перераховані в таблиці. 10.1) дуже часто використовуються для вирішення широкого кола завдань програмування системного рівня, наприклад, при опитуванні інформації про стан пристрою або її формування.

Таблиця. 10.1

Оператор	Значення
&	Порозрядне І (AND)
|	Порозрядне АБО (OR)
^	Порозрядне виключаючеАБО (X0R)
>>	Зрушення управо
<<	Зрушення вліво
~	Доповнення до 1 (унарний оператор НІ)

У наступній таблиці показаний результат виконання кожної порозрядної операції для усіх можливих поєднань операндів (нулів і одиниць).

Таблиця. 10.2

x	y	x & y	x | y	x ^ y	~x
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	1
1	1	1	1	0	0

Специфікатор register

Спочатку це специфікатор застосовувався тільки до змінних типу char і int, але тепер його можна застосовувати до змінних будь-якого типу. Даний специфікатор вказує компілятору зберігати значення змінної не в пам'яті, а в регістрі процесора. Інший трактуванням специфікатор register служить підказка компілятору, що даний об'єкт використовується дуже інтенсивно. Зрозуміло в регістрах зможуть помістится тільки дані дуже обмеженого обсягу, такі як int і char, а більш великі об'єкти в регістри не вмістяться, але отримають більш високою пріоритет обробки ..

Оператори зсуву

Оператори зсуву призначені для зсуву бітів в рамках цілочисельного значення.

Оператори зсуву, ">>" і "<<", зрушують все біти в значенні вправо або вліво. Загальний формат використання оператора зсуву вправо виглядає так.

значення »кількість _ зсувів

А оператор зсуву вліво використовується так.

значення «кількість _ зсувів

Тут елемент колічество _ зсувів вказує, на скільки позицій має бути зрушене значення. При кожному зсуві вліво все біти, що становить значення, зсуваються вліво на одну позицію, а в молодший розряд записується нуль. При кожному зсуві вправо все біти зсуваються, відповідно, вправо. Якщо зрушення вправо піддається значення без знака, в старший розряд записується нуль. Якщо ж зрушення вправо піддається значення зі знаком, значення знакового розряду зберігається і не зрушується. Як ви пам'ятаєте, негативні цілі числа представляються установкою старшого розряду числа рівним одиниці. Таким чином, якщо зрушувати значення негативно, при кожному зсуві вправо в старший розряд записується одиниця, а якщо позитивно - нуль. Не забувайте, зрушення, що виконується операторами зсуву, не є циклічним, тобто при зсуві як вправо, так і вліво крайні біти втрачаються, і вміст втраченого біта дізнатися неможливо.

Оператори зсуву працюють тільки зі значеннями цілочисельних типів, наприклад, символами, цілими числами і довгими цілими числами. Вони неспроможні до значень з плаваючою точкою.

Специфікатор register

Спочатку це специфікатор застосовувався тільки до змінних типу char і int, але тепер його можна застосовувати до змінних будь-якого типу. Даний специфікатор вказує компілятору зберігати значення змінної не в пам'яті, а в регістрі процесора. Інший трактуванням специфікатор register служить підказка компілятору, що даний об'єкт використовується дуже інтенсивно. Зрозуміло в регістрах зможуть поміститися тільки дані дуже обмеженого обсягу, такі як int і char, а більш великі об'єкти в регістри не вмістяться, але отримають більш високою пріоритет обробки ..

Робота з файлами

Вивід (запис) в файл

fstream (скорочення від «FileStream») - заголовки зі стандартної бібліотеки C ++, що включає набір класів, методів і функцій, які надають інтерфейс для читання / запису даних з / в файл. Для маніпуляції з даними файлів використовуються об'єкти, звані потоками («stream»).

Функції, включені в даний файл, дозволяють зчитувати з файлів як побайтово, так і блоками, і записувати так само. У комплект включені всі необхідні функції для управління послідовністю доступу до даних файлів, а також безліч допоміжних функцій

Для того щоб працювати з файловими потоками необхідно підключити бібліотеку #include "fstream" У цій бібліотеці є такий клас ofstream (висновок інформації в файл), але в цьому класі немає об'єктів введення виведення (cout cin) тому ми повинні створити об'єкт ось таким чином ofstream fout

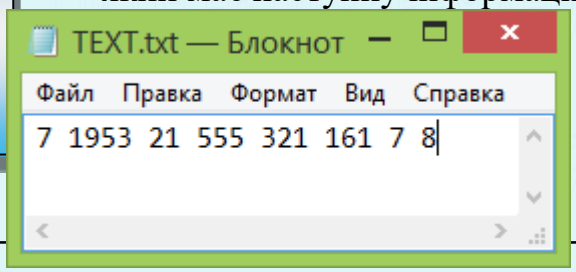
(Fout -довільний ім'я - в нашому випадку вибираємо збігається зі змістом - висновок.

При використанні цього об'єкта через (.) Необхідно вказати метод (функція) open (), яка повинна прийняти текст в текстовий файл, ім'я якого необхідно вказати в дужках

При повторній передачі інформації, остання переписується в файлі.

Введення (читання) з файлу

Перш за все такий файл повинен існувати - нехай це файл - TEXT.txt, який має наступну інформацію.



Увага: після останнього числа пробіл не ставити

Тепер необхідно створити для класу ifstream об'єкт, який буде зчитувати з інформацію з файлу, підключивши метод open (ім'я файлу) fin.open ("TEXT.txt");

Перед доступом до файлу необхідно переконатися, що такий файл існує для цього можна скористатися функцією is_open, яка повертає значення «true» якщо файл відкрився

Лістінг 10

```
using std::ofstream;
using std::ios;
using std::ifstream;


int main()
{
	ifstream  fin;
	fin.open("TEXT.txt");

    if (!fin.is_open())

	{
		cout << "File is absent ";
		return -1;
	}

	int mas[7];

	for (int i=0; i < 7; i++)
	{
		fin >> mas[i];
		cout << mas[i];
	}

	_getch;
	return 0;
}
```

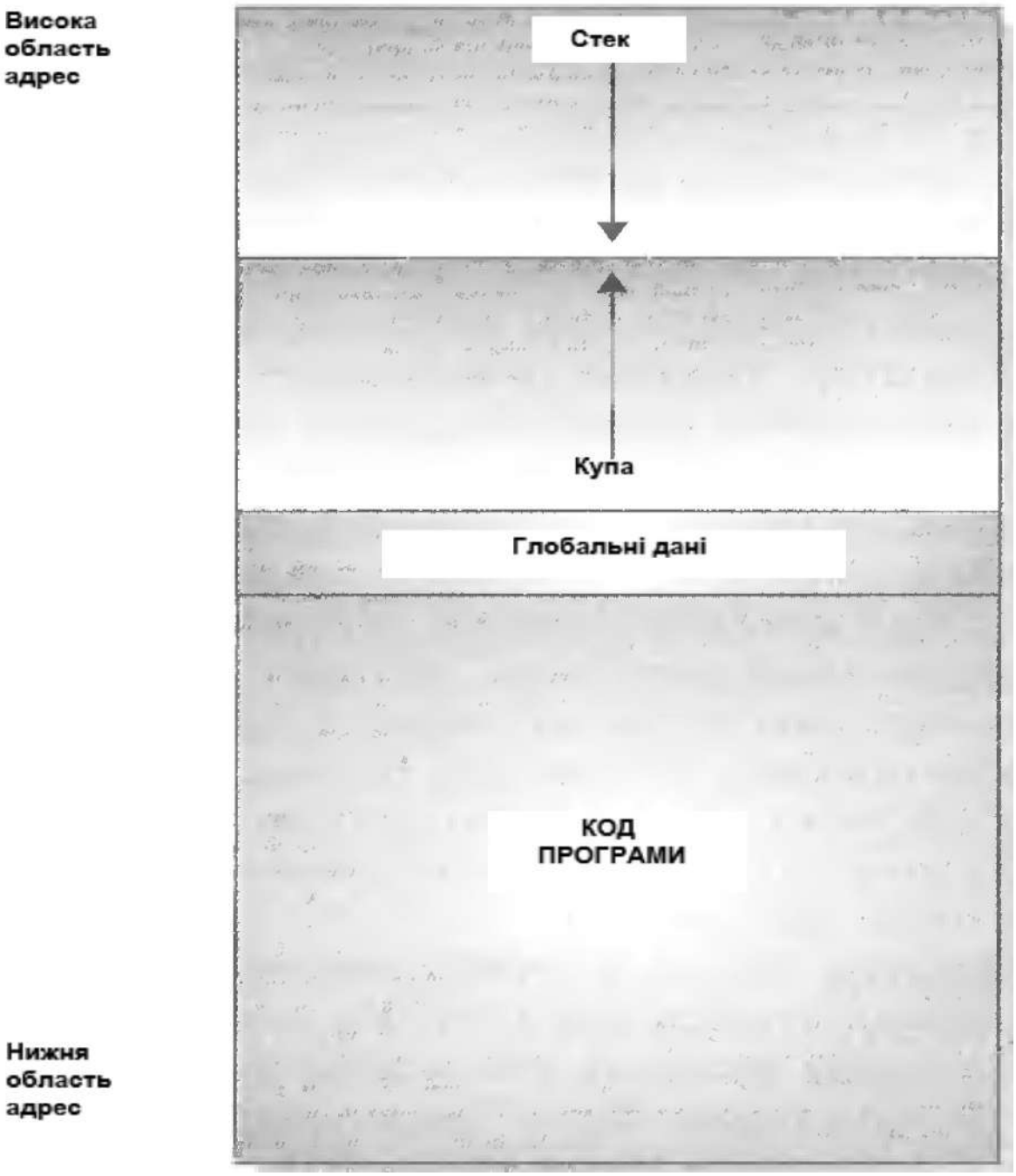
Лекція 11 «Динамічна пам'ять, і ще про оператори»

План

1 Динамічний розподіл пам'яті з використанням операторів new і delete
2 Оператор "знак питання"
3 Складові оператори присвоювання
4 Оператор "кома"
5 Використання ключового слова Sizeof

Динамічний розподіл пам'яті

Для С ++ - програми існує два основних способи зберігання інформації в основний пам'яті комп'ютера. Перший полягає у використанні змінних. Область пам'яті, що надається змінним, закріплюється за ними під час компіляції і не може бути змінена при виконанні програми. Другий спосіб полягає у використанні С ++ - системи динамічного розподілу пам'яті. У цьому випадку пам'ять для даних виділяється в міру необхідності з розділу вільної пам'яті, який розташований між вашою програмою (і її постійної областю зберігання) і стеком. Цей розділ називається "купою" (heap). (Розташування програми в пам'яті схематично показано на рис.)


Висока область адрес
Стек
Купа
Глобальні дані
КОД ПРОГРАМИ
Нижня область адрес

Система динамічного розподілу пам'яті - це засіб отримання програмою деякої області пам'яті під час її виконання.}

Щоб задовольнити запит на динамічне виділення пам'яті, використовується так звана "купа". Неважко припустити, що в деяких надзвичайних ситуаціях вільна пам'ять "купи" може вичерпатися. Отже, незважаючи на те, що динамічний розподіл пам'яті (у порівнянні з фіксованим) забезпечує більшу гнучкість, але і в цьому випадку воно має свої межі.

Мова C ++ містить два оператора, new і delete, які виконують функції по виділенню і звільнення пам'яті.

Їх загальний формат.

```
int * p;
змінна-вказівник = new тіп_переменной;
p = new int

delete змінна-вказівник;
delete p;
```

Оператор new дозволяє динамічно виділити область пам'яті.

Оператор delete звільняє раніше виділену динамічну пам'ять.

Ініціалізація динамічно виділеної пам'яті

Використовуючи оператор new, динамічно виділяється пам'ять можна форматувати. Для цього після імені типу необхідно задати початкове значення, уклавши його в круглі дужки. Наприклад, в наступній програмі область пам'яті, що адресується покажчиком p, инициализируется числом 99.

Листинг 11.1

```
#include "stdafx.h"
#include <iostream>
using namespace std;


int main()
{
	int *p;
	p = new int(100);
	cout << *p;
	delete p;

	system("pause");
	return 0;
	}
```

Виділення пам'яті для масивів

За допомогою оператора new можна виділяти пам'ять і для масивів. Ось як виглядає загальний формат операції виділення пам'яті для одновимірного масиву:

Змінна - покажчик = new тип [розмір];

Тут елемент розмір задає кількість елементів в масиві.

Щоб звільнити пам'ять, виділену для динамічно створеного масиву, використовуйте такий формат оператора delete:

delete [] змінна - покажчик;

Тут елемент змінна-вказівник є адреса, отриманий при виділенні пам'яті для масиву (за допомогою оператора new). Квадратні дужки означають для C ++, що динамічно створений масив видаляється, а вся область пам'яті, виділена для нього, автоматично звільняється.

Оператор "знак питання"

Одним з найбільш чудових операторів C ++ є оператор "?". Оператор "?" можна використовувати в якості заміни if-else-інструкцій, що вживаються в наступному загальному форматі.

```
if (умова)
вираз 1;
else
вираз2;
```

Оператор "?" називається тернарного, оскільки він працює з трьома операндами.

Ось його загальний формат запису:

```
Вираз1? Вираз2: вираз3;
```

Всі елементи тут є виразами. Зверніть увагу на використання і розташування двокрапки.

Значення? - вираз визначається наступним чином. Обчислюється Вираз1. Якщо воно виявляється істинним, обчислюється Вираз2, і результат його обчислення стає значенням за все? -виражена. Якщо результат обчислення елемента Вираз 1 виявляється хибним, значенням за все? - виражена стає результат обчислення елемента Вираженіе3.

Оператор "кома"

Оператор "кома" може становити частину виразу. Його призначення в цьому випадку - зв'язати певним чином кілька виразів. Значення списку виразів, розділених комами, визначається в цьому випадку значенням крайнього праворуч вираження. Значення інших виразів відкидаються. Отже, значення виразу справа стає значенням всього виразу-списку. Наприклад, при виконанні цієї інструкції

var = (count = 19, incr ++, count + 1); змінної count спочатку присвоюється число 19, змінної incr - число 10, а потім до значення змінної count додається одиниця, після чого змінної var присвоюється значення крайнього праворуч вираження, тобто count + 1, що дорівнює 20.

Круглі дужки тут обов'язкові, оскільки оператор "кома" має більш низький пріоритет, ніж оператор присвоювання.

Лекція № 12 «Структури даних.»

План

1 Загальний формат оголошення структур

2 Масиви структур

3 Створення функцій, які працюють зі структурою

Загальний формат оголошення структур.
Структури

У С ++ структура являє собою колекцію об'єднаних загальним ім'ям змінних, яка забезпечує зручний засіб зберігання родинних даних в одному місці. Структури - це сукупні типи даних, оскільки вони складаються з декількох різних, але логічно пов'язаних змінних. З тих же причин їх іноді називають складовими або конгломератного типами даних.

Перш ніж буде створено об'єкт структури, повинен бути визначений її формат. Це робиться за допомогою оголошення структури. Оголошення структури дозволяє зрозуміти, змінні якого типу вона містить. Змінні, що становлять структуру, називаються її членами. Члени структури також називають елементами або полями

Член (поле) структури - це змінна, яка є частиною структури.

Загальний формат оголошення структури виглядає так.

лістинг 12.1

```
struct імя_ тіпа_ структури
{
тип імя_елемента1;
тип імя_елемента2;
тип ім'я елемента3;
тип імя_елементаM;
} Структурны_змінні (екземпляр);
```

Доступ до членів (полях, атрибутам) структури

Щоб звернутися до поля екземпляра структури, потрібно скористатися оператором "точка", який розділяє екземпляр структури і будь-яке поле з цієї структури. Так здійснюється доступ до всіх полів екземпляра структури. Загальний формат доступу записується так.

екземпляр структури.поле = значення;

Лістінг 12.2

```
student stud_1;


stud_1.lname = "First";
stud_1.progress = 85;
stud_1.absent = 10;
```

```
student stud_1; // создание первого экземпляра структуры

stud_1.

_getch(
return
}
```

absent
lname
progress

public : int student::absent
Файл: l12.cpp

Масиви структур

Структури можуть бути елементами масивів. І справді, масиви структур використовуються досить часто. Щоб оголосити масив структур, необхідно спочатку визначити структуру, а потім оголосити масив елементів цього структурного типу. Наприклад, щоб оголосити 100-елементний масив структур student), досить записати наступне. Struct Student d [100];

Щоб отримати доступ до конкретної структурі в масиві структур, необхідно індексувати ім'я структури.

Наприклад, щоб відобразити на екрані вміст поля absent третьої структури, використовуйте таку інструкцію. cout << d [2]. absent;

Подібно до всіх змінним масивів, у масивів структур індексування починається з нуля

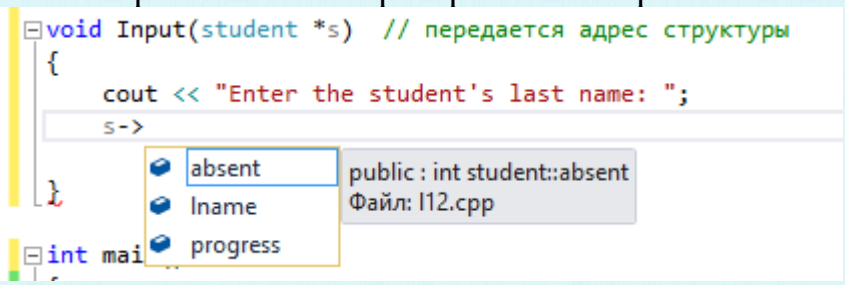
.

Створення функцій, які працюють зі структурою

Функції введення

Створення функції введення примірників структури, де будемо використовувати покажчик - * s

Скористаємося оператором -> Вибір елемента за вказівником



Скористаємося оператором -> Вибір елемента за вказівником

Лістінг 12.3

```
void Input(student *s)  )
{
	cout << "Enter the student's last name: ";
	s->lname = new char[20];

	cin.get();
	cin.getline(s->lname, 20);

	cout << "Enter the student's academic performance: ";
	cin>> s->progress;

	cout << "Enter the missing student activities: ";
	cin >> s->absent;
	cout << endl;
}
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ

МЕТОДИЧНІ ВКАЗІВКИ
до практичних занять студентів з дисципліни

«Програмування ч.1»

для студентів усіх форм навчання
напряму 6.050903 «Телекомунікації»

Електронний документ

ЗАТВЕРДЖЕНО
кафедрою ІМІ.
Протокол № 1
від 31.08.2017

Харків 2017

Методичні вказівки до практичних занять з дисципліни «Програмування ч.1» для студентів усіх форм навчання напряму 6.050903 «Телекомунікації» [Електронний документ] / Упоряд. ОП.Малінін. – Харків: ХНУРЕ, 2017. – 11 с.

Упорядник: О.П. Малінін

ЗМІСТ

1 Заняття №1 Тема «Алгоритмізація»…….109
Практичне заняття №2 «Цикли і Масиви»……..113

Заняття №1 Тема «Алгоритмізація»

Мета Заняття : Вивчення графічного способу опису алгоритму розв'язання задачі.

Завдання заняття:
- ознайомитися з основними способами подання алгоритмів;
- освоїти графічний спосіб опису алгоритмів

На початку заняття студенти повинні відповісти на наступні питання:

1. Мета алгоритмізації
2. Дайте визначення алгоритму.
3. Якими властивостями повинна володіти алгоритм.
4. Назвіть способи запису алгоритмів найбільш поширених на практиці
5. Назвшть три основних види графічних алгоритмів алгоритмів:
6 Охарактеризуйте - лінійний алгоритм
7 Охарактеризуйте- розгалужується алгоритм.
8 Охарактеризуйте- циклічний алгоритм.
9. Назвіть види блоків і їх призначення. (При графічної реалізації алгоритмів).

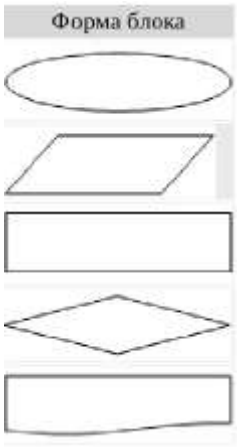


Рисунок 1.1

Для запису алгоритму будь-якої складності досить трьох базових структур:

слідування - позначає послідовне виконання дій (рис. 1.2, а);

розгалуження - відповідає вибору одного з двох варіантів дій (рис. 1.2, б);

цикл-поки - визначає повторення дій, поки не буде порушено умова, виконання якого перевіряється на початку циклу (рис. 1.2, в).

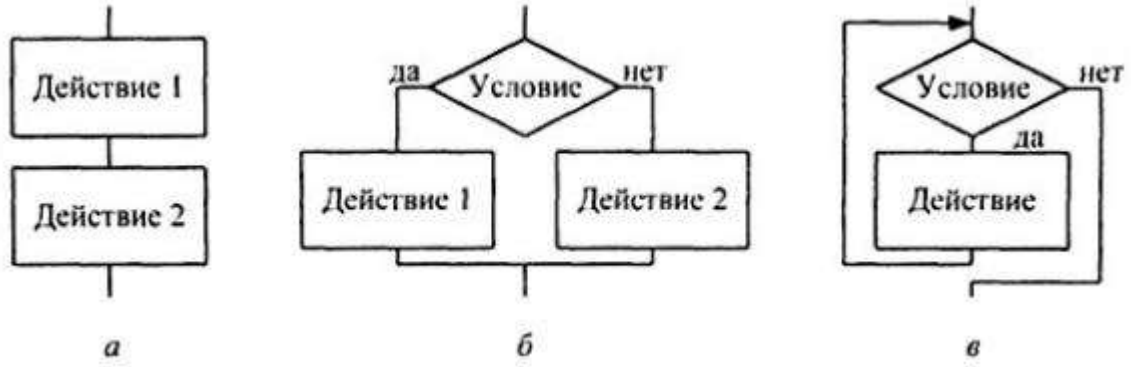


Рисунок 1.2 - Базові алгоритмічні структури

Приклады рішення завдань

Завдання №1

Розробити блок схему алгоритму определяния площади и периметра прямоугольника со сторонами a и b

Рішення :

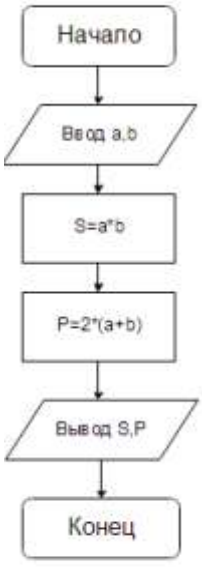


Рисунок 1.3

Завдання №2

Розробити блок схему алгоритму для наступної задачі:

Швидкість першого автомобіля - V1 км / год, другого - V2 км / год, відстань між ними S км. Яка відстань буде між ними через T годин, якщо автомобілі рухаються в різні боки? Значення V1, V2, T і S задаються з клавіатури.

Рішення :

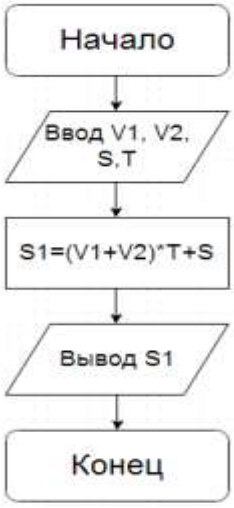


Рисунок 1.4

Завдання №3

Розробити блок схему алгоритму виведення цифри, якщо вона перевищує певний поріг. Значення цифр задаються з клавіатури.

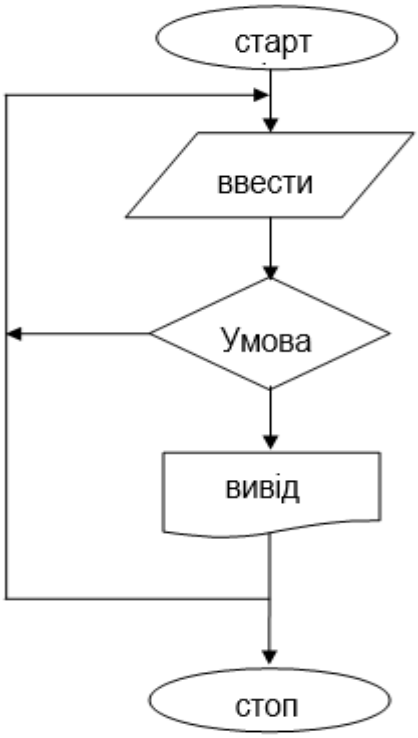


Рисунок 1.5

Завдання яке необхідно виконати.

Розробити блок схему алгоритму визначення кількості цифри «5» в масиві (Mass) з 10 чисел, а також організувати масив (Poin), елемент якого вказував би порядковий номер цифри «5» в масиві (Mass). Вивести отримані результати

Практичне заняття №2 «Цикли і Масиви»

Мета Заняття : Закріпити матеріал який вивчається в лекції «Цикли. Масиви »,і вміти використовувати отримані теоретичні знання на практиці

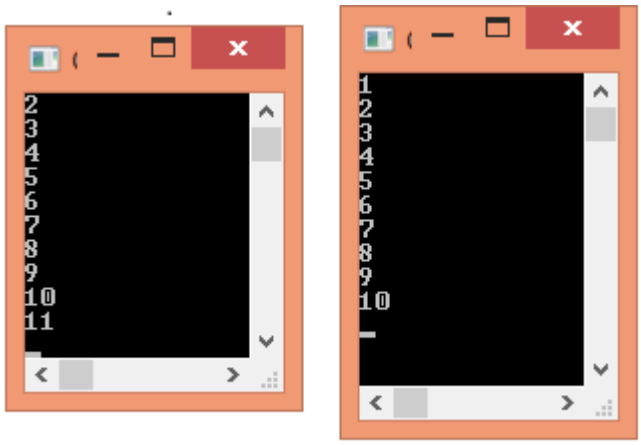
Цикл While

Відповісти на наступні питання:
1. Зобразіть синтаксичний шаблон циклу While
2. Наведіть приклади з використанням оператора While:
- Керований лічильником цикл.
- Керований подією цикл
3. Що виводить наступний цикл

Лістінг 1

```
int main ()
{
int number;
number = 1;
while (number <11)
{
number ++;
cout << number << endl;
}

_getch ();

return 0;
}
```

4 Зміна порядок операторів (але не їх форму) в циклі з попереднього вправи, таким чином щоб програма виводила числа від 1 до 10.



5 Нижче наведено простий цикл, керований лічильником:

Лістінг 2

```
int main()
{

      int count;
      count = 1;
      while (count < 20)
      {
            cout << count << "  ";
            count++;
   }
      _getch();
      return 0;
}
```

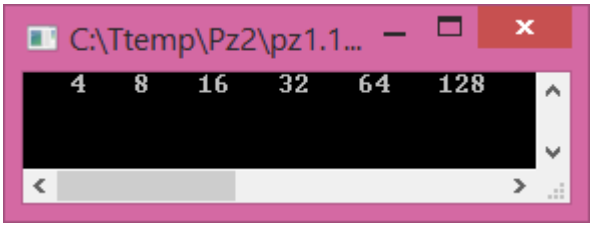
а). Перерахуйте три способи зміни циклу, щоб він виконувався 20 разів, а не 19.
б). Які з цих способів дозволяють змінної count пробігати значення від 1 до 21?

```
1 count = 0; while (count < 20)
2 count = 1; while (count < 21)
3 count = 0; while (count != 20)
```

6 Скільки ітерацій робить цикл при виконанні наступного коду?

Лістінг 3

```
int main()
{

      int number;
      number = 2;
      bool done;
      done = false;
      while (!done)
      {
            number = number * 2;
            if(number > 64)
                  done = true;
            cout << "   " << number ;
      }

      _getch();

      return 0;
```



7 Передбачається, що наступний код повинен надрукувати всі парні числа в діапазоні від 1 до 15, якщо це завдання нездійсненна за допомогою даного коду то чому (якщо є помилки в даному коді виправте їх).

Лістінг 4 a

```
int n;
	n = 2;
	while (n != 15)
{
		n = n + 2;
	cout <<   n  << " ";
}

	_getch();

	return 0;
}
```

Вірний код

Лістінг 4 b

```
int main()
{

	int n;
	n = 0;
	while (n <= 13)
{
		n = n + 2;
	cout <<   n  << " ";
}

	_getch();

	return 0;
}
```

Цикл DO While

Відповісти на наступні питання

1. У чому основна відмінність циклів While і Do- While?
2. Зобразіть синтаксичний шаблон циклу While
3. Що буде надруковано в результаті роботи наступного фрагмента програми, якщо вводиться значення 0?

Лістінг 5

```
int main()
{
	int i, n;
	cin >>   n;
	i = 1;
	 do
	{
		cout <<  i;
		i++;
	} while (i <= n);

	_getch();

	return 0;
}
```

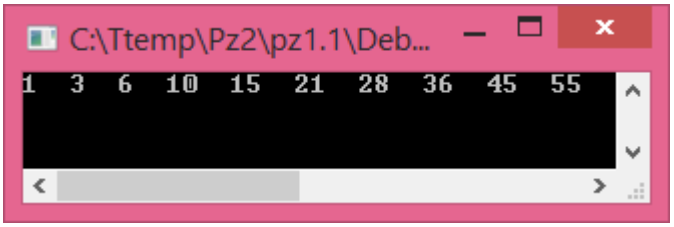
Оператор For

Відповісти на наступні питання

1. Зобразіть синтаксичний шаблон циклу *For*

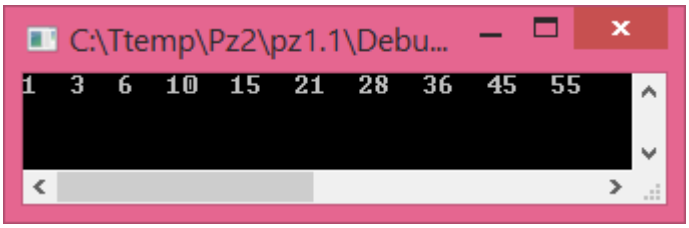
2. Перепишіть наступний фрагмент програми, використовуючи цикл For.

```
int main()
{
	int sum, count;
	sum = 0;
	count = 1;
	while (count <= 10)
	{
		sum = sum + count;
		count++;
		cout << sum << "  ";
	}

	_getch();

	return 0;
}
```



Лістінг 6

```
int main()
{
	int sum, count;
	sum = 0;

	for (count = 1; count <= 10;count++)
	{
		sum = sum + count;

		cout << sum << "  ";
	}

	_getch();

	return 0;
}
```



3. У чому полягає відмінність інструкцій break і continue

Одновимірний масив

Відповісти на наступні питання

1. Зобразіть синтаксичний шаблон оголошення одновимірного масиву
2. Зобразіть синтаксичний шаблон ініціалізації масиву
3. Що буде надруковано в результаті роботи наступного фрагмента програми

Лістінг 7

```
int main()
{
	int mas[5] = {5, 10, 15, 20, 25}; // ініціализація масиву


	for (int i = 1; i < 6; i++)
	{

		cout << "mas[i]=  " << mas[i] << endl;

	}

	_getch();

	return 0;
	 }
```

Увага !!!! Завдання

В коді є помилка, необхідно її знайти і виправити

4. Запропонувати блоковий алгоритм сортування масиву бульбашковим методом, якщо словесний алгоритм описаний нижче, а лістингом 9 показано реалізацію цього алгоритму (див. Лекцію 6)

Алгоритм сортування бульбашковим методом отримав свою назву від способу, використовуваного для упорядкування елементів масиву. Тут виконуються повторювані операції порівняння і при необхідності міняються місцями суміжні елементи. При цьому елементи з меншими значеннями поступово переміщуються до одного кінця масиву, а елементи з великими значеннями - до іншого. Цей процес нагадує поведінку бульбашок повітря в резервуарі з водою. Бульбашкова сортування виконується шляхом декількох проходів по масиву, під час яких при необхідності здійснюється перестановка елементів, які опинилися "не на своєму місці". Кількість проходів, які гарантують отримання відсортованого масиву, дорівнює кількості елементів в масиві, зменшеному на одиницю.

лістінг 8

```
int main()
{

	int nums[10];
	int a, b;
	int size;
	int i;

	size = 10; // Кількість елементів, що підлягають сортуванню.
	for (i = 0; i<size; i++)
	nums[i] = rand()%10;                        // Розміщуємо в масив випадкові числа.

	for (i = 0; i < size; i++)
		cout << nums[i] << " ";           // Відображаємо вихідний масив.

	// Реалізація методу бульбашкового сортування.
	for (a = 1; a<size; a++)
		for (b = size - 1; b >= a; b--)
		{
			if (nums[b - 1] > nums[b])
			{
				i = nums[b - 1];
				nums[b - 1] = nums[b];
				nums[b] = i;
			}
		}
	// Кінець бульбашкового сортування.

	cout << endl;
	cout << endl;
	cout << "array is sorted: ";
	cout << endl;
	cout << endl;

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

		cout << nums[i] << " ";    // Відображаємо відсортований масив.


_getch();
```

```
    return 0;
}
```

Завдання для самоперевірки

Створіть масив з 15 випадкових цілих чисел з відрізка [0; 9]. Виведіть масив на екран. Підрахуйте скільки в масиві парних елементів і виведете це кількість на екран на окремому рядку.

Створіть масив з 8 випадкових цілих чисел з відрізка [1; 10]. Виведіть масив на екран у рядок. Замініть кожен елемент з непарним індексом на нуль. Знову виведете масив на екран на окремому рядку.

Користувач повинен вказати з клавіатури натуральне число більше двох, а програма повинна створити масив зазначеного розміру з випадкових цілих чисел з [-5; 5] і вивести його на екран у рядок. Якщо користувач введе невідповідний число, то програма повинна вимагати повторного введення до тих пір, поки не буде вказано коректне значення.

Створіть 2 масиву з 5 випадкових цілих чисел з відрізка [0; 5] кожен, виведіть масиви на екран в двох окремих рядках. Порахуйте середнє арифметичне елементів кожного масиву і повідомте, для якого з масивів це значення виявилося більше (або повідомте, що їх середнє арифметичне рівні).

Створіть два масиви з 10 цілих випадкових чисел з відрізка [1; 9] і третій масив з 10 дійсних чисел. Кожен елемент з i-им індексом третього масиву повинен дорівнювати відношенню елемента з першого масиву з i-им індексом до елемента з другого масиву з i-им індексом. Вивести всі три масиву на екран (кожен на окремому рядку).

Створіть масив з 4 випадкових цілих чисел з відрізка [10; 99], виведіть його на екран у рядок. Визначити і вивести на екран повідомлення про те, чи є масив строго зростаючої послідовністю.

Створіть масив з 12 випадкових цілих чисел з відрізка [-15; 15]. Визначте який елемент є в цьому масиві максимальним і повідомте індекс його останнього входження в масив.

Створити масив з 20 випадкових цілих чисел з відрізка [0; 13] і вивести його на екран. Створити другий масив потрібного розміру тільки з парних елементів першого масиву, якщо вони там є, і вивести його на екран.

Створіть масив з 11 випадкових цілих чисел з відрізка [-1; 1], виведіть масив на екран у рядок. Визначте який елемент зустрічається в масиві найчастіше і виведіть про це повідомлення на екран. Якщо два якихось елемента зустрічаються однакову кількість разів, то не виводьте нічого.

Програма повинна створити масив розміру 12 з випадкових цілих чисел з [-6; 6] і вивести його на екран у рядок. Після цього програма повинна визначити і повідомити користувачеві про те, сума модулів який половини масиву більше: лівої чи правої, або повідомити, що ці суми модулів рівні.

Електронний навчальний документ

МЕТОДИЧНІ ВКАЗІВКИ
до практичних занять

«ПРОГРАМУВАННЯ ч.1»

для студентів усіх форм навчання
напряму 6.050903 Телекомунікації

Упорядник: МАЛІНІН Олександр Петрович

Відповідальний випусковий В.М. Безрук

Авторська редакція

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з дисципліни

«ПРОГРАМУВАННЯ Ч.1»

для студентів усіх форм навчання
напряму 6.050903 «Телекомунікації»

Електронний документ

ЗАТВЕРДЖЕНО
кафедрою ІМІ.
Протокол № 1 від 31.08.2017

Харків 2017

123

Методичні вказівки до лабораторних робіт з дисципліни «Програмування ч.1» для студентів усіх форм навчання напряму 6.050903 «Телекомунікації» [Електронний документ] / Упоряд. О.П. Малінін . – Харків: ХНУРЕ, 2017. – 27с.

Упорядники: О.П. Малінін

ЗМІСТ

Загальні положення……125
1 Знайомство з середовищем розробки додатків - Visual Studio ……..…..126
2 Арифметичні вирази. Організація інтерактивного введення і виведення…..132
3 Умови та цикли …….....136
4 Функції, Рядки і двовимірні масиви ……..…145
5 Покажчики та оператори ……..…...151

ЗАГАЛЬНІ ПОЛОЖЕННЯ

С ++ - надзвичайно потужна мова програмування , яка містить засоби створення ефективних програм практично будь-якого призначення, від низькорівневих утиліт і драйверів до складних програмних комплексів самого різного призначення. Зокрема:

Підтримуються різні стилі і технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, метапрограмування (шаблони, макроси).

Передбачуване виконання програм є важливою перевагою для побудови систем реального часу. Весь код, неявно генерується компілятором для реалізації мовних можливостей (наприклад, при перетворенні змінної до іншого типу), визначений в стандарті. Також строго визначені місця програми, в яких цей код виконується. Це дає можливість заміряти або розраховувати час реакції програми на зовнішнє подія.

Функції для-оператори дозволяють коротко і ємко записувати вирази над користувацькими типами у природному алгебраїчній формі.

Використовуючи шаблони, можливо створювати узагальнені контейнери і алгоритми для різних типів даних, а також спеціалізувати і обчислювати на етапі компіляції.

Можливість створення вбудованих предметно-орієнтованих мов програмування.

Тематика лабораторних їхній обсяг визначаються відповідно до робочої програми.

Правила виконання лабораторних робот.

Перед виконанням лабораторної роботи кожен студент зобов’язаний: – знати відповідний розділ теоретичного курсу і мати чітке уявлення про

досліджувані процеси;

– вивчити методичні вказівки та інструкції до відповідної лабораторної роботи, методику проведення вимірювань необхідних величин та методику аналізу результатів;

– ознайомитися з необхідною для роботи вимірювальною апаратурою, правилами її вмикання та експлуатації;

– подати оформлений звіт про попередню лабораторну роботу.

Лабораторні роботи виконуються кожним студентом індивідуально або у складі бригади.

Перед початком лабораторної роботи викладач проводить співбесіду зі студентами з метою з’ясування їхньої підготовки до роботи. Під час співбесіди з’ясовується знання студентами фізичних процесів, що відбуваються в окремих функціональних ланках або у пристрої в цілому, знання методики проведення експериментів і очікуваних результатів, уміння користуватися необхідною вимірювальною апаратурою.

Лабораторна робота №1 Тема «Знайомство з середовищем розробки додатків - Visual Studio»

Самостійна підготовка: Повторити матеріал, який викладено в лекціях №3

Мета роботи: освоїти середовище розробки Додатків - Visual Студіо і продуктивно використовувати його ресурси

Організація інтерактивного введення і виведення

Хід виконання роботи

1.1 Запустити Visual Studio. Відкриється наступне вікно (рис 1.1)

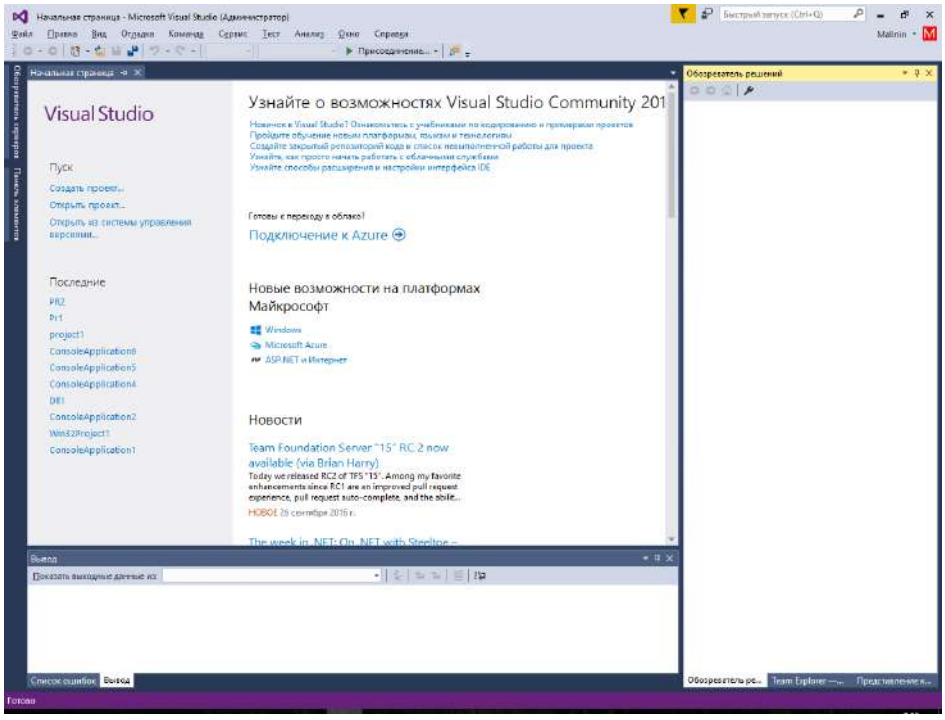


рис 1.1

1.2 Створити новий проект.

Створити новий проект можна декількома способами:

- натиснути клавіші «Ctrl» + «Shift» + «N»;
- пройти шлях Файл / Створити / Проект;
- у вікні «Початкова сторінка» вибрати рядок «Створити проект ...».

Відкриється вікно «Створення проекту» рис 1.2

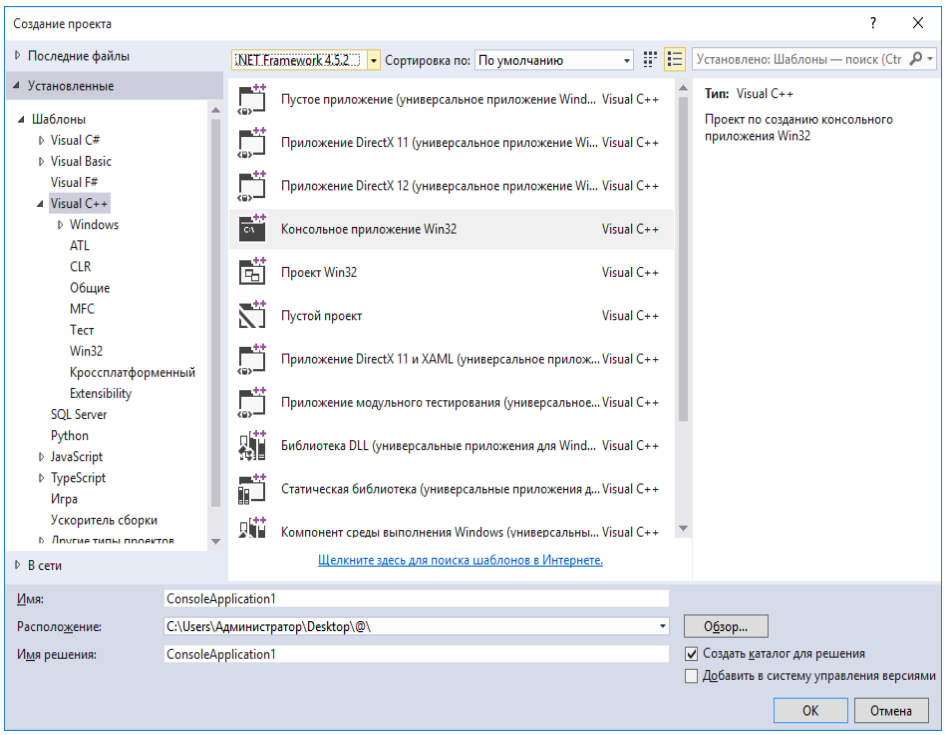


рис 1.2 вікно «Створення проекту»

1.3 У вікні, вибрати:

- мова програмування, в нашому випадку це Visual C ++;
- робочу платформу (.NET Framework ...) залишити обраної за замовчуванням;
- так як проект буде створюватися для консолі, вибрати «консольний додаток»;
- в рядку «Ім'я» вказати ім'я створюваного додатка, при цьому «Рішення» приймає таке ж ім'я, що і проект. Рішення може містити кілька проектів (в нашому випадку один);
- в рядку «Розташування» вказати місце дислокації створюваного додатка (консультація у адміністратора РС):
- натиснути на клавішу «ок». Відкриється вікно рис 1.3, де необхідно натиснути на кнопку «Готово».

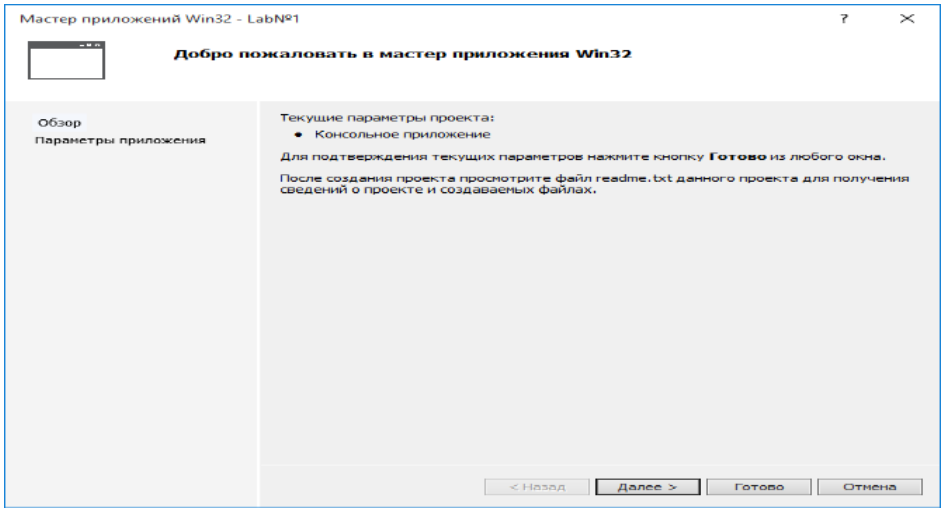


рис 1.3

на малюнку 1.4 показаний вид середовища програмування, вікна якої відкриються після попередньої дії (для версії 2015)

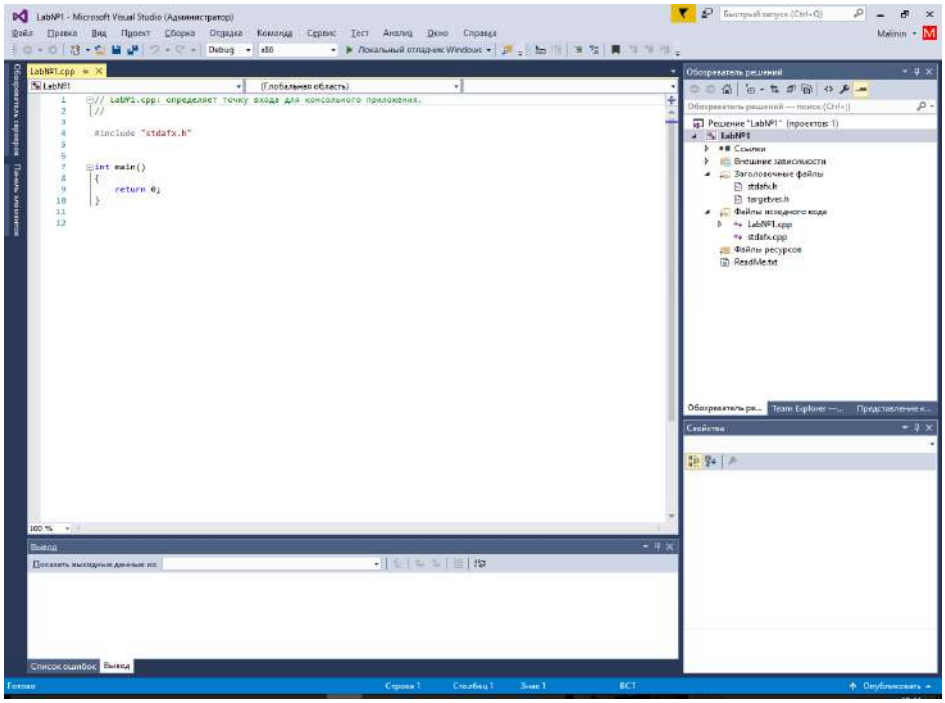


Рис 1.4

У лівій частині середовища програмування знаходиться вікно текстового редактора, в якому необхідно написати текст створюваної програми. Якщо у версії Visual Studio, яку Ви використовуєте, текстовий редактор не активний, його необхідно запустити наступним чином:

- З закладки «Вид», яка знаходиться на панелі головного меню, викликати «Обозреватель рішень» і у вікні «Обозревателя рішень» для створюваного проекту в папку «Файли вихідного коду» помістити файл з розширенням cpp і з ім'ям відповідним вашому проекту

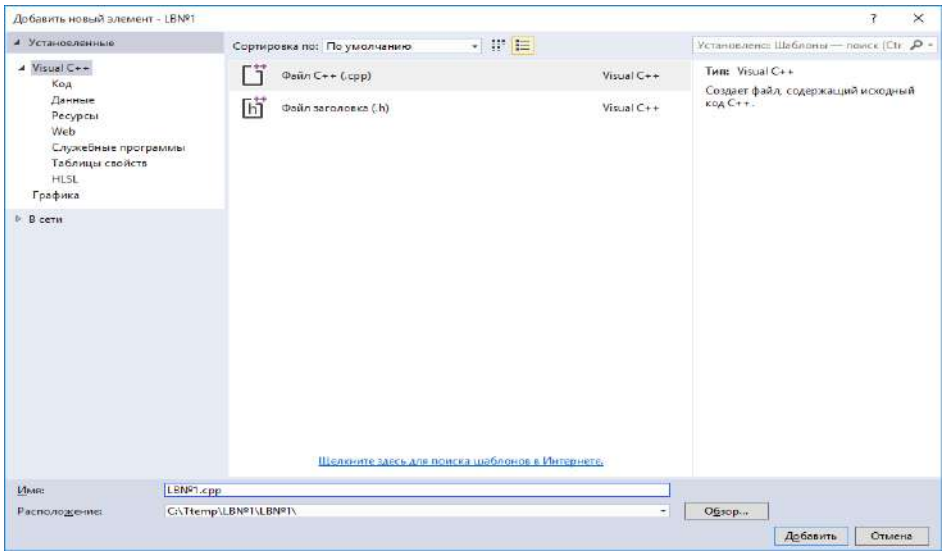


Рис 1.5

1.4 Створення програми

Будь-яка програма на C ++ повинна містити функцію з ім'ям main. З цієї функції починається виконання програми. Можна уявити, що main - це господиня, а інші функції її помічники.

- Набрати код програми, який показаний на малюнку 1.4

- Виконати складання створюваного проекту, для чого на панелі головного меню вибрати закладку «Збірка» і натиснути на рядок «Побудувати ім'я проекту»

- Подібним чином виконати налагодження проекту.

Результат на рис 1.6

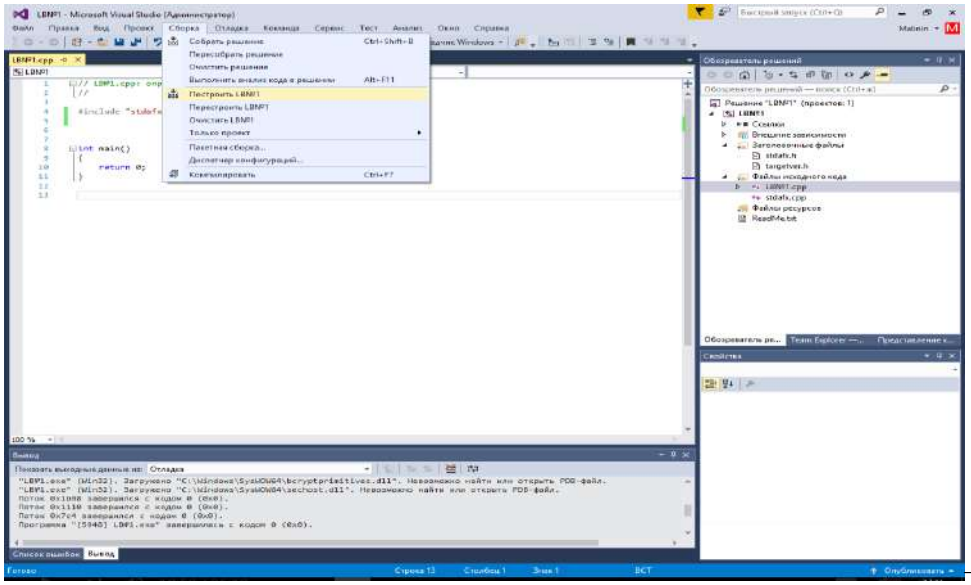


Рис 1.6

- Відкрити папку, яка була створена для зберігання створюваного проекту. Далі відкрити папку «Debug» і запустити файл з розширенням «exe» Прокоментувати отриманий результат

- Для того щоб зафіксувати результат необхідно зупинити закриття консолі, для цього можна скористатися функцією - _getch () введення символу, але до цього необхідно підключити бібліотеку, яка містить цю функцію. За допомогою функції препроцесора #include "conio.h" (В деяких версіях Visual Studio замість "" необхідно використовувати <>) організувати виклик необхідної бібліотеки. Помістити функцію _getch () в тіло програми см pic1.7. Налагодити і виконати програму. Прокоментувати результат.

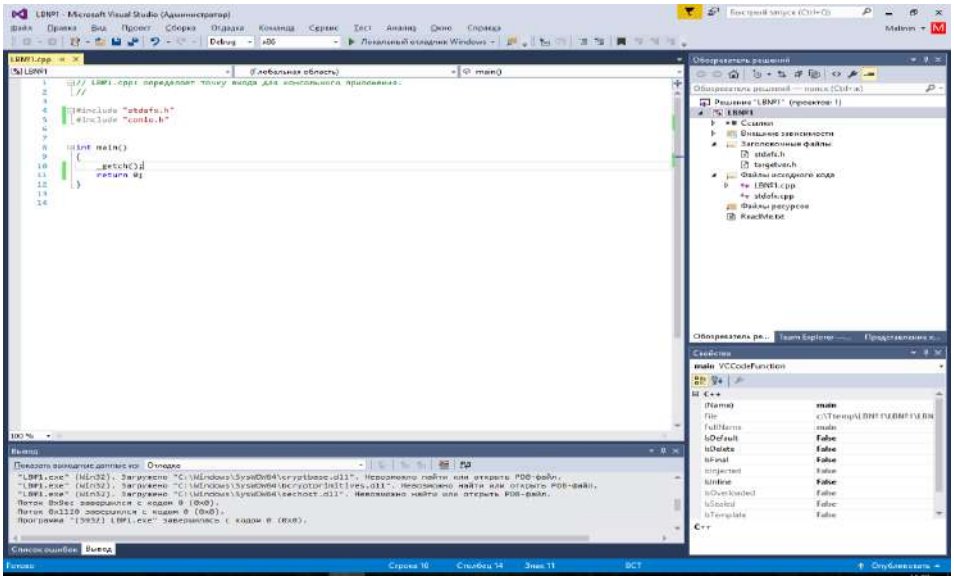


рис1.7

- Вивести текстову фразу на екран монітора за допомогою терміналу. Перед цим необхідно підключити препроцесором стандартну бібліотеку мови С ++ "iostream" і за допомогою директиви using оголосити об'єкти cout і endl в просторі імен std (стандартне простір імен) як ті імена які будуть використовуватися програмою (див рис 1.8)

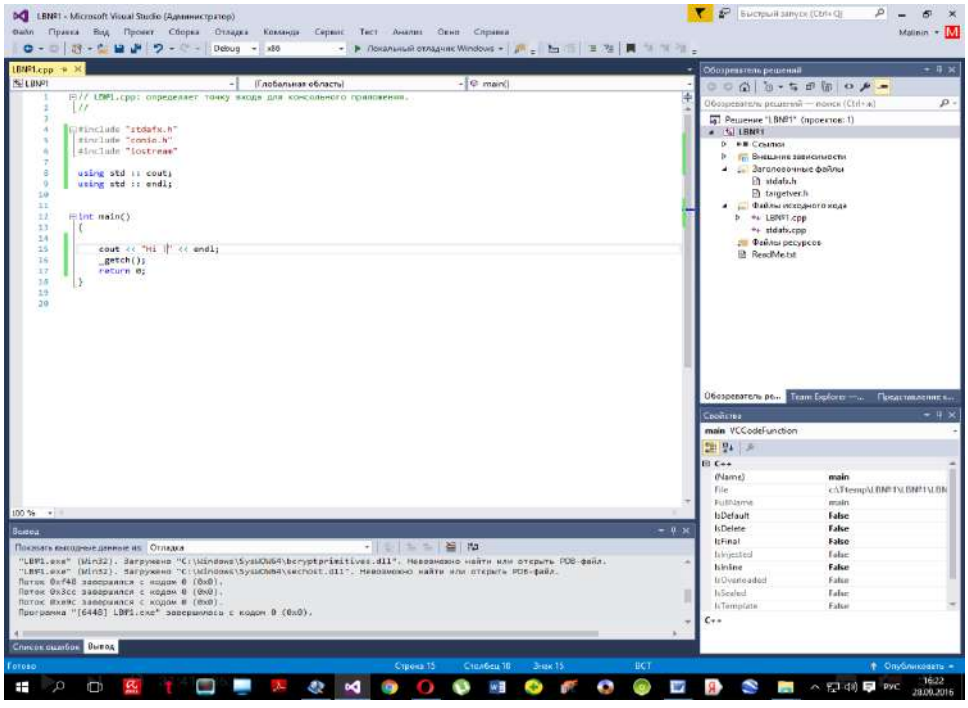


рис1.8

Налагодити і виконати програму

Завдання №1

Написати програму, яка виводила б наступну інформацію:
Name: вказати Name
FirstName: вказати FirstName
Year of birth: рік
group Number: номер
Phone number: номер
Skipe: Nick
Email: Phone number
При цьому необхідно використовувати табуляцію (\ tPeter)

Примітка: переклад на новий рядок можна організувати за допомогою керуючого символу new line- \ n ("текст \ n").

С допомогою функції Setlocale можна в об'єкті cout мови С ++ використовувати кирилицю.

Набрати в тілі програми setlocale (LC_ALL, "Russian");

Завдання №2

Виконати завдання №1 з використанням кирилиці.

1.5 Зміст звіту

Звіт з лабораторної роботи має містити:

1 Мету й постановку завдання;

2 Результати виконання завдань (скріншоти, отримані в результаті лабораторної роботи);

3 Висновки.

Лабораторна робота №2 Тема «Арифметичні вирази. Організація інтерактивного введення і виведення »

Самостійна підготовка: Повторити матеріал, який викладено в лекціях №4
Мета роботи: Закріплення матеріалу за темами:
«Арифметичні вирази»

Завдання №1

Набрати програму, лістинг якої наведено нижче, описати призначення кожного рядка. Виконати цю програму і якщо виникне помилка (змінна «byte» прийме значення нуль см. Pic_2.2) пояснити виникнення даної помилки і виправити її. Якщо дана помилка не відбудеться, привласнити значення змінної «GB» значення 1000 запустити програму і пояснити виникнення помилки (ок)

```
4   #include "stdafx.h"
5
6   #include "conio.h"
7   #include "math.h"
8   #include "iostream"
9
10  using std::cout;
11  using std::endl;
12
13
14
15
16
17  int main()
18  {
19
20      const int Size = 1024; // описание и инициализация константы
21      int GB;
22      int MB;
23      int Kb;
24      int byte;
25
26      GB = 1000;
27      MB = GB*Size;
28      Kb = MB*Size;
29      byte = Kb*Size;
30      cout << "Gigabytes = " << GB << endl;
31      cout << "Megabytes = " << MB << endl;
32      cout << "Kilobytes = " << Kb << endl;
33      cout << "Bytes = " << byte << endl;
34
35
36      _getche();
37      return 0;
38  }
39
```

Рис 2.1

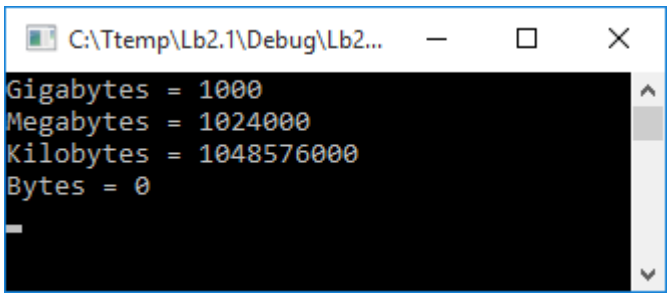


Рис 2.2

Заданння №2

Написати програму, яка знаходить витрата бензину (у кілометрах на літр) під час автомобільної поїздки.

Дано кількість палива при заправці. Під час поїздки машину заправляли чотири рази (11.7, 14 .3, 12.2 і 8.5 літрів бензину), а також поточна й кінцеве значення автопробігу (67308.0 і 68750.5 кілометрів відповідно). Вважається, що перша заправка була перед початком поїздки (бак був порожній до заправки) і що в кінці поїздки бензин скінчився.

«Організація інтерактивного введення і виведення»

Інтерактивної (inter active) називається програма, за допомогою якої користувач обмінюється інформацією безпосередньо з комп'ютером.

Для того щоб інтерактивна програма могла отримати дані, треба спочатку

надрукувати так звану вхідну підказку (input prompt) - повідомлення, що пояснює користувачеві, які дані потрібно ввести. Без підказки користувач

швидше за все, не впізнає, що саме очікує програма. Програма повинна також роздруковувати всі величини, отримані із зовні, з тим, щоб користувач міг переконатися в тому, що вони введені правильно. Відображення вхідних величин на пристрої виведення називається ехопечатью (echo printing).

Фрагмент програми, що ілюструє правильне використання підказок:

```
cout << "Enter the Personnel number";
cin >> Personnel_No;
```

Для введення даних з клавіатури використовується операція витягання «>>»

Оператор вилучення використовує два аргументи. Аргумент зліва від «>>» є потоковим виразом (в найпростішому випадку це потоковая змінна, така як cin). Правим аргументом укзивается ім'я тієї змінної значення, якої буде введено з клавіатури.

Змінна типу istream - cin, може бути використана в програмі тільки після того як директивою «using» буде описаний об'єкт «cin» в просторі імен «std».

Завдання 3.

Напишіть програму, в якій був би використаний інтерактивний введення - висновок. І яка виводила б таблицю з шапкою, показаної на рис 2.3, а значення рядка було б задано в процесі роботи програми, яка використовує об'єкт cin

Personnel No	Salary	Gender	Presence

Рис 2.3

При цьому типи змінних:
Personnel No - int;
Salary -float;
Gender - char;
Presence - bool

Завдання 4

Напишіть програму, яка перетворює значення температури, виражене в градусах Цельсія (° C), в значення за шкалою Фаренгейта (° F). Формула перетворення має такий вигляд:

T (° F) = (9/5) T ('C) + 32.

Використовуйте іменовану константу для позначення температури за Цельсієм для того, щоб її значення можна було легко міняти. Програма повинна виводити з відповідними позначками як значення температури за Цельсієм, так і еквівалент цього значення в градусах Фаренгейта. Включити в програму коментарі, виберіть осмислені позначення змінних.

Зміст звіту

Звіт з лабораторної роботи має містити:

1 Мету й постановку завдання;

2 Результати виконання завдань (скріншоти, отримані в результаті лабораторної роботи);

3 Висновки.

Лабораторна робота №3 Тема «Умови та цикли»

Самостійна підготовка: Повторити матеріал, який викладено в лекціях №5

Мета роботи: Закріплення матеріалу за темами:
1 «Умови»
2 «Цикли»

Частина 1УМОВИ

3.1 Операції порівняння

1. Запустіть в VS і для Visual C ++ виберіть консольний додаток
2. Наберіть код, який представлений на малюнку 3.1

```
#include "stdafx.h"
#include "conio.h"
#include "math.h"
#include "iostream"

using std::cout;
using std::endl;
using std::cin;


int main()
{
    int a, b;

    cout << "Enter a,b = " << endl;
    cin >> a >> b;

    cout << "a="<<  a <<",b="<<b << endl;

    cout << "a==b   " << (a == b) << endl;
    cout << "a!=b   " << (a != b) << endl;
    cout << "a>b    " << (a < b) << endl;
    cout << "a<b    " << (a < b) << endl;
    cout << "a>=b   " << (a >= b) << endl;
    cout << "a<=b   " << (a <= b) << endl;

   _getch();

    return 0;
}
```

Рис 3.1

3. Запустіть програму. Задайте значення: a = 6, b = 6. В результаті роботи програми буде отримано повідомлення схоже на те, яке показано на малюнку 3.2

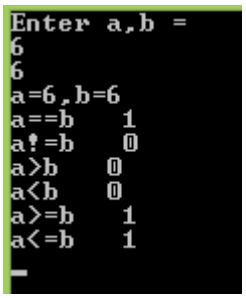


Рис 3.2

4. При оформленні звіту поясніть, чому були отримані такі результати

5. Запустіть цю програму кілька разів, змінюючи значення a і b. Прокоментуйте результати

Умовний оператор в формі If-Then-Else

1. Наберіть код, який представлений на малюнку 3.5.
2. Запустіть програму і надайте для N значення спочатку рівне 3, а потім нерівний 3, переконайтеся, що програма працює правильно.
3. Змініть знак «==» в умови if (N == 3) на знак «=», запустіть програму і надайте для N значення спочатку рівне 3, а потім нерівний 3.
4. Поясніть, в якому випадку, виникає помилка і з якої причини. Для пояснення з'явилася помилки можна скористатися програмою, яка показана на рис 3.1. додавши ще один оператор, `cout << "a=b " << (a = b) << endl;`, (як ви розумієте він з помилкою, але для експерименту можна спробувати)

```
19  int main()
20  {
21      int N;
22      cout << "enter N=";
23      cin  >>   N;
24
25      if (N == 3)
26          cout <<   "N   equals  3";   //  equals - равно
27      else
28          cout  <<   "N  doesn't   equal  3";
29
30  _getch();
31
32
33      return 0;
34  }
35
```
Рис 3.5

Вкладені умовні оператори зі структурою If-Then- Else-If:

Для вирішення завдання 3.1 можна використовувати програму з вкладеними умовними операторами, лістинг такої програми показаний на рис 3.6

```
using std::cout;
using std::endl;
using std::cin;


int main()
{

    int month;

        cout << " number month = ";
        cin >> month;
        cout << " number month = " << month << endl;
        if (month == 1)
            cout << "   January";
        else if (month == 2)
            cout << "   February";
         else if (month == 3)
            cout << "   March";
         else if (month == 4)
             cout << "   April";
         else if (month == 5)
             cout << "   May";
         else if (month == 6)
             cout << "   June";
         else if (month == 7)
             cout << "   July";
         else if (month == 8)
             cout << "  August";
         else if (month == 9)
             cout << "   September";
         else if (month == 10)
             cout << "   October";
         else if (month == 11)
             cout << "  November";
          else  cout << " December";
_getch();
        return 0;
}
```

Рис 3.6

5. Наберіть код, який представлений на малюнку 3.6 і виконайте програму.

6. Пояснити в чому перевага даної програми від програми, показаної на рис 3.3

На рис 3.7 реалізований один із запитів.

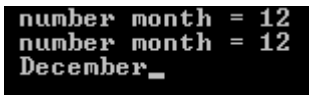


Рис 3.7

3.3 Інструкція switch

Інструкція switch- це інструкція багатонаправленого розгалуження, яка дозволяє вибрати одну з безлічі альтернатив. Хоча багатонаправлені тестування можна реалізувати за допомогою послідовності вкладених if-інструкцій як ми робили вище, у багатьох ситуаціях інструкція switch виявляється більш ефективним рішенням. Вона працює в такий спосіб. Значення виразу послідовно порівнюється з константами із заданого списку. При виявленні збігу для однієї з умов порівняння виконується послідовність інструкцій, пов'язана з цією умовою.

Загальний формат запису інструкції switch такий:
switch (выражение)

{
case значение1:
операторы
[break]
case значение2:
операторы
[break]

..
default:
операторы
[break]
}

Ця інструкція дає можливість передбачити оператор за замовчуванням (default :), який виконується, якщо не знайдено жодне відповідність, см. Pic3.8.

7. Виконати завдання 3.1, використовуючи інструкцію switch додавши, оператор default:

default: cout << "Month with this number does not exist";

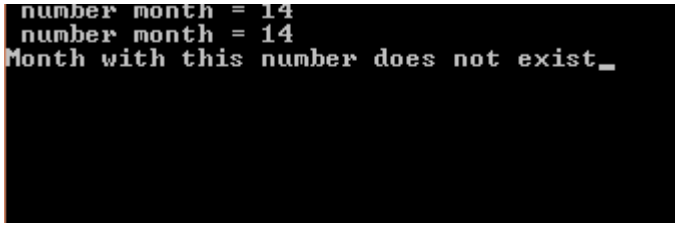


Рис 3.8

Частина 2 ЦИКЛИ

3.4 Оператор While
Загальна форма циклу while має такий вигляд:

while (вираз) інструкція;

Тут під елементом інструкція розуміється або одиночна інструкція, або блок інструкцій. Роботою циклу управляє елемент «вираз», який являє собою будь-який допустимий C ++ - вираз. Елемент інструкція виконується до тих пір, поки умовний вираз повертає значення true. Як тільки цей вислів стає хибним, управління передається інструкції, яка слідує за цим циклом.

Цикл керований лічильником

Завдання 3.2. Вивести всі порядкові номери місяців року.

1. Наберіть код, який представлений на малюнку 3.9.
2. Запустіть програму, проаналізуйте її роботу

```
int main()
{
    int month;
    month = 1;

    while (month <= 12)
    {
        cout << month << endl;
        month ++;

    }

    _getch();
    return 0;
}
```

```
1
2
3
4
5
6
7
8
9
10
11
12
```

Рис 3.10 Рішення завдання 3.2

3.5 Оператор Do - While

Цикл do-while - це єдиний цикл, який завжди робить ітерацію хоча б один раз.

На відміну від циклу while, в якому умова перевіряється при вході, цикл do -while перевіряє умова при виході з циклу. Це означає, що цикл do - while завжди виконується хоча б один раз. Його загальний формат має такий вигляд:

Do
{
инструкции;
}
while (выражение);

Цикл do- while виконується до тих пір, поки залишається істинним елемент вираз, який представляє собою умовний вираз.

Завдання 3.2 «Гра з комп'ютером» Комп'ютер «задумує» число, а Ви повинні його вгадати. Використовуючи цикл do-while, програма "не випустить" Вас з циклу вгадування, доки Ви не вгадаєте задумане комп'ютером число.

```
15  int main()
16  {
17      int my_number; // моё число
18      int your_namb; // вариант пользователя
19      my_number = rand()%10; // rend()- генератор случайных чисел от 0 до 10.
20
21      do {
22          cout << "Enter your version number: " << endl;
23          cin >> your_namb;
24
25          if (your_namb == my_number)
26          {
27              cout << "** Ok ** " << endl;
28              cout << my_number << " -  it is the  my_number." << endl;
29          }
30          else
31          {
32              cout << "SORRY, but you're wrong." << endl;
33              if (your_namb > my_number)
34                  cout << " Your version exceed the my_numberr." << endl;
35                  else cout << "Your version is less than the my_number." << endl;
36          }
37      } while (your_namb != my_number);
38
39      _getch();
40
41      return 0;
42  }
```

Рис 3.11 Код програми «Гра з комп'ютером»

3. Напишіть код програми (Рис 3.11) і запустіть програму.

4. У звіті дайте Ваші коментарі щодо кожного рядка коду програми «Гра з комп'ютером»

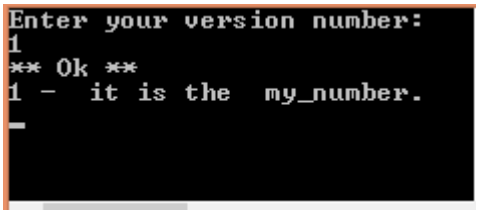


Рис 3.11 Результат роботи програми «Гра з комп'ютером»

3.6 Цикл for - універсальний цикл C ++.

Загальний формат запису циклу for для багаторазового виконання однієї інструкції має такий вигляд.

for (ініціалізація; вираз інкремент) інструкція;

Якщо цикл for призначений для багаторазового виконання не однієї інструкцій, а
програмного блоку, то його загальний формат виглядає так.

for (ініціалізація; вираз; інкремент)
{
послідовність інструкцій
}

Елемент ініціалізація зазвичай являє собою інструкцію присвоювання, яка встановлює керуючу змінну циклу рівною початкового значення. Ця змінна діє в якості лічильника, який керує роботою циклу. Елемент вираз являє собою умовний вираз, в якому тестується значення керуючої змінної циклу. Результат цього тестування визначає, виконається цикл for ще раз чи ні. Елемент інкремент- це вираз, який визначає, як змінюється значення керуючої змінної циклу після кожної ітерації.

Зверніть увагу на те, що всі ці елементи циклу for повинні відділятися крапкою з комою. Цикл for буде виконуватися до тих пір, поки обчислення елемента вираз дає істинний результат. Як тільки це умовний вираз стане хибним, цикл завершиться, а виконання програми продовжиться з інструкції, наступної за циклом for

Наберіть текст програми і проаналізуйте її виконання.

Листинг 3.1

```
#include "stdafx.h"
#include "conio.h"
#include "math.h"
#include "iostream"
using std::cout;
using std::endl;
```

```
using std::cin;

int main()
{
	for (int i = 0; i <= 10; i++)
	{
		cout << "i=  " << i << endl;
	}
		_getch();
	return 0;
	}
```

Завдання 3.3. Напишіть програму, використовуючи оператор For, яка виводила б по порядку всі місяці року з їх порядковими номерами.

3.7 Зміст звіту

Звіт з лабораторної роботи має містити:

1 Мету й постановку завдання;

2 Результати виконання завдань (скріншоти, отримані в результаті лабораторної роботи);

3 Висновки.

Лабораторна робота №4 Тема «Функції, Рядки і двовимірні масиви»

Самостійна підготовка: Повторити матеріал, який викладено в лекціях №7 та № 8

Мета роботи: Закріплення матеріалу за темами:
«Типи даних»

Хід виконання лабораторної роботи
функції
1. Запустити середовище програмування «Visual Studio»
2. Ввести і налагодити програму лістинг, якої показаний на малюнку 4.1. Ця програма в залежності від поточного часу, яка вводиться за допомогою клавіатури, виводить одну з фраз:
- з 6 до 12 «Good morning»;
- з 12 до 16 «Good afternoon»;
- з 16 до 20 Good evening;
- з 20 до 6 Good night;
Вкажіть в звіті, чому дана конструкція використання послідовності операторів if не є ефективною.
3. Розробіть програму, яка вирішувала б цю задачу з використанням функцій виду void (див. Макет програми рис 4.2)

```
4   #include "stdafx.h"
5
6
7   #include "conio.h"
8   #include "math.h"
9   #include "iostream"
10
11  using std::cout;
12  using std::endl;
13  using std::cin;
14
15
16  float time;
17
18  int main()
19  {
20      cout << "Enter the current time: ";
21      cin >> time;
22
23
24      if (time >= 6.1 && time <= 12)                    // с 6 до 12  «Good morning»;
25          cout << "Good morning" << endl;
26      if (time >= 12.1 && time <= 16)                  // с 12 до 16 «Good afternoon»;
27          cout << "Good afternoon" << endl;
28      if (time >= 16.1 && time <= 20)                  //  с 16 до 20 Good evening;
29          cout << "Good evening" << endl;
30      if (time >= 20.1 || time <= 6)                   //  с 20 до 6 Good night;
31          cout << "Good night" << endl;
32
33      _getch();
34
35      return 0;
36  }
37
```

Рис 4.1

```
int main()
{
	cout << "Enter the current time: ";
	cin >> time;
                                        // с 6 до 12 «Good morning»
                                        // с 12 до 16 «Good afternoon»
                                        // с 16 до 20 «Good evening»
                                        // с 20 до 6 «Good night»
	    _getch();

	return 0;
}

void morning()
{

}

void afternoon()
{

}

void evening()
{

}

void night()
{

{
```

Рис 4.2 Макет програми

4.1 Наберіть код програми, який показаний на рис 4.3 і запустіть програму.

4.2 сторопів словесний алгоритм роботи даної програми

4.3 Вкажіть, чим відрізняється робота функції типу void у програмі рис 4.3 від функції типу void з програми рис 4.2

```
#include "cctype"
#include "conio.h"
#include "math.h"
#include "iostream"

using std::cout;
using std::endl;
using std::cin;


  void mul(int x, int y);

  int main()
  {
	mul(10, 20);
	mul(5, 6);
	mul(8, 9);
	_getch();
	return 0;
  }
  void mul(int x, int y)
  {
	cout << x * y << endl;
  }
```

Рис 4.3

Рядки

Розібратися в програмі (рис 4.4) і написати коментарі для кожної її рядки. Дана програма підраховує кількість слів у введеній фразі з допомогою клавіатури.

Примітка:

Мінлива numberWords використовується для підрахунку слів у введеній фразі

Str -контейнери для введеного рядка

isalpha () - функція, яка перевіряє поточний символ на букву і повертає

- false якщо не буква

- true якщо буква

inWord - маркер, який приймає значення:

- false, якщо поточний символ не буква,

- true якщо - буква

Для того щоб можна було скористатися функцією isalpha () необхідно підключити бібліотеку <cctype>

```
15  int main()
16  {
17
18      char str[80];
19      int numberWords = 0;
20      bool inWord = false;
21
22      cout << "Enter string: ";
23      cin.getline(str, 80);
24
25      cout << "Str:    "<< str << "\n";
26
27      for (int i(0); str[i] != '\0'; i++)
28      {
29          if (isalpha(str[i]) && !(inWord))
30          {
31              numberWords++;
32              inWord = true;
33          }
34          if (!isalpha(str[i]))
35              inWord = false;
36      }
37      cout << "number Words= " << numberWords << endl;
38      _getch();
39      return 0;
40
41  }
42
```

Рис 4.4

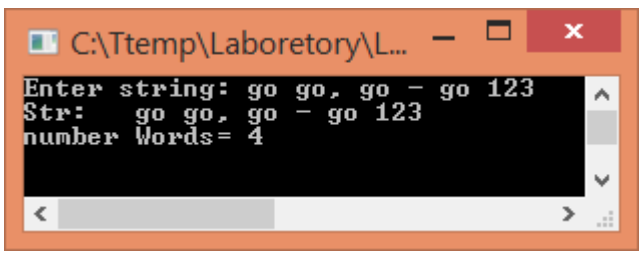


Рис 4.5

Двувимені масиви

```
5
6   #include "conio.h"
7   #include "math.h"
8   #include "iostream"
9
10  using std::cout;
11  using std::endl;
12  using std::cin;
13
14  int main()
15  {
16      int s, c, num[3][4];
17      int n = 1;
18
19      for (s = 0; s < 3; ++s)
20      {
21          for (c = 0; c < 4; ++c)
22          {
23              num[s][c] = n++;
24              cout << num[s][c] << " ";
25          }
26          cout << '\n';
27      }
28
29      _getch();
30      return 0;
31  }
32
```

Рис 4.6

1 У програмі, яка показана на рис 4.6 змініть так код, щоб програма формувала елементи масиву таким чином, щоб число, записане в комірку масиву, відповідало її номеру (див. Рис 4.7)

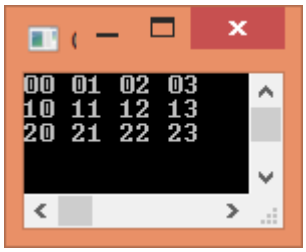


Рис 4.7

Ініціалізація масивів

Завдання: Розробіть програму, яка спочатку виконує пошук заданого числа в масиві (від 1 до 10), а потім виводить відповідне йому значення в ступені три.

Масиви рядків

Завдання: Введіть програму, лістинг якої показаний на рис 4.8 і запустіть її на виконання. Напишіть коментар до кожної її рядку.

Результат роботи цієї програми помістіть в звіт

```
{
	int  i;
	char str[][80] = {
	                   {"first line"},
	                   {"second line "},
	                   {"third line "}
	                 };

	//for (i = 0; i < sizeof(str); i++)
	 for ( i = 0; i < 3; i++)
	{
		cout << str[i] << '\n';

	}

		_getch();

		return 0;

}
```

4.5 Зміст звіту

Звіт з лабораторної роботи має містити:

1 Мету й постановку завдання;

2 Результати виконання завдань (скріншоти, отримані в результаті лабораторної роботи);

3 Висновки.

Лабораторна робота №5 Тема «Покажчики та оператори»

Методичні вказівки з організації самостійної роботи студентів

Повторити теми «Покажчики», «порозрядному оператори, робота з файлами» «Динамічна пам'ять, і ще про операторів»

1. Завдання на тему «Покажчики»

1.1. Використовуючи покажчики стеків написати код, який дозволив би визначити довжину наступних типів даних

short int
long int
long long
double

1.2. Написати програму, яка могла б порахувати кількість букв «А» в тексті, який знаходиться в пам'яті PC

Під час написання даної програми можна скористатися таблицею №5.1

2. Завдання на тему «порозрядному оператори, Робота з файлами»

2.1 Написати програму перекодування заголовної (великий) букви в прописну (див. Таблицю 5.1)

2.2 Написати програму яка виконувала б «швидку» операцію множення на 2, 4 і 8

```
int root, tor, right;
right =15;
root = 10;
tor =27;
```

3.2 Правильність роботи кодів із завдання 1.1 перевірити за допомогою ключового слова sizeof

множення на 2, 4 і 8

3. Завдання на тему «Динамічна пам'ять, і ще про операторів»

3.1. Використовуючи оператор «Кома» в залежності від умови, "одночасно" змінити значення (в довільному вигляді) змінних root, tor, right, якщо відомо:

Таблица5.1 «Символы ASCII»

Dec	Hex	Char	Cmd	Dec	Hex	Char	Cmd	Dec	Hex	Char	Cmd	Dec	Hex	Char	Cmd
0	00		NUL	32	20		(sp)	64	40	@		96	60	`	
1	01	☺	SOH	33	21	!		65	41	A		97	61	a	
2	02	☻	STX	34	22	"		66	42	B		98	62	b	
3	03	♥	ETX	35	23	#		67	43	C		99	63	c	
4	04	♦	EOT	36	24	$		68	44	D		100	64	d	
5	05	♣	ENQ	37	25	%		69	45	E		101	65	e	
6	06	♠	ACK	38	26	&		70	46	F		102	66	f	
7	07	•	BEL	39	27	'		71	47	G		103	67	g	
8	08	◘	BS	40	28	(		72	48	H		104	68	h	
9	09	○	TAB	41	29	)		73	49	I		105	69	i	
10	0A	◙	LF	42	2A	*		74	4A	J		106	6A	j	
11	0B	♂	VT	43	2B	+		75	4B	K		107	6B	k	
12	0C	♀	FF	44	2C	,		76	4C	L		108	6C	l	
13	0D	♪	CR	45	2D	-		77	4D	M		109	6D	m	
14	0E	♫	SO	46	2E	.		78	4E	N		110	6E	n	
15	0F	☼	SI	47	2F	/		79	4F	O		111	6F	o	
16	10	►	DLE	48	30	0		80	50	P		112	70	p	
17	11	◄	DC1	49	31	1		81	51	Q		113	71	q	
18	12	↕	DC2	50	32	2		82	52	R		114	72	r	
19	13	‼	DC3	51	33	3		83	53	S		115	73	s	
20	14	¶	DC4	52	34	4		84	54	T		116	74	t	
21	15	§	NAK	53	35	5		85	55	U		117	75	u	
22	16	▬	SYN	54	36	6		86	56	V		118	76	v	
23	17	↨	ETB	55	37	7		87	57	W		119	77	w	
24	18	↑	CAN	56	38	8		88	58	X		120	78	x	
25	19	↓	EM	57	39	9		89	59	Y		121	79	y	
26	1A	→	SUB	58	3A	:		90	5A	Z		122	7A	z	
27	1B	←	ESC	59	3B	;		91	5B	[		123	7B	{	
28	1C	∟	FS	60	3C	<		92	5C	\		124	7C	|	
29	1D	↔	GS	61	3D	=		93	5D	]		125	7D	}	
30	1E	▲	RS	62	3E	>		94	5E	^		126	7E	~	
31	1F	▼	US	63	3F	?		95	5F	_		127	7F	⌂	DEL

5.4 Зміст звіту

Звіт з лабораторної роботи має містити:

1 Мету й постановку завдання;

2 Результати виконання завдань (скріншоти, отримані в результаті лабораторної роботи);

3 Висновки.

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт

«ПРОГРАМУВАННЯ ч.1»

для студентів усіх форм навчання
напряму 6.050903 Телекомунікації

Упорядник: МАЛІНІН Олександр Петрович

Відповідальний випусковий В.М. Безрук

Авторська редакція

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра ІМІ

КОНТРОЛЬНІ ЗАВДАННЯ

з дисципліни «Програмування ч.1»

для студентів усіх форм навчання
напряму 6.050903 «Телекомунікації»

Електронний документ

ЗАТВЕРДЖЕНО
кафедрою ІМІ
протокол № 1
від 31.08.2017

Харків 2017 р.

Контрольні завдання з дисципліни «Програмування ч.1» для студентів усіх форм навчання напряму 6.050903 «Телекомунікації»
[Електронний документ] / Упоряд.: О.П. Малінін. Харків: ХНУРЕ, 2017. 6 с.

Упорядник О.П. Малінін

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 1

1. Що таке мова програмування
2. Поняття строки
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____ Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил. — Парал. тит. англ.

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____1_____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 2

1. Поняття алгоритму. Види алгоритмів
2. Функції.
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____ Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил. — Парал. тит. англ.

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____1_____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 3

1. Блок-схеми. Графічна реалізація алгоритмів
2. Функції бібліотеки «***cstring»***
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил. — Парал. тит. англ.

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____1_____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 4

1. Елементи програм на С ++. Структура програми
2. Двовимірні масиви
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил.

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____1_____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 5

1. Синтаксис і семантика. Ідентифікатори
2. Багатовимірні масиви
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____ Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил. — Парал. тит. англ.

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____1_____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 6

1. Дані та їх типи
2. Безрозмірна "ініціалізація масивів
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____ Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил.

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____1_____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 7

1. Описи змінних. Основні арифметичні операції
2. Масиви рядків
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил. — Парал. тит. англ.

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____1_____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 8

1. Препроцесор
2. Пркажчики та змінні, організація адресного простору в обчислювальних системах
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил.

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____1_____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 9

1. Арифметичні вираження
2. Покажчики та масиви
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____ Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил. — Парал. тит. англ.

_____ ХТУРЕ _____ Форма № У-Я.09
(назва вищого навчального закладу)

Спеціальність _____ _____ Семестр _____1_____

Навчальний предмет «Програмування»

КОНТРОЛЬНА РОБОТА 10

1. Потік управління. Умови та логічні вираження. Оператор “***SWITCH”***
2. Покажчики та рядки. методи адресації
3. Розробити алгоритм і написати код для програми

Видано «» _____2017 р. Видав _____ Малінін О.П.
(Підпис) (прізвище, ініціали)

Основна література:

1.Алгоритмизация Введение в язык программирования С++, 2-е издание Белоцерковская и др. Национальный открытый университет «ИНТУИТ» 2016
2. Шилдт, Герберт. C++: базовый курс, 3-е издание. : Пер. с англ. — М. : Издательский дом "Вильяме", 2010. — 624 с.: ил.

Харківський національний університет радіоелектроніки

Кафедра ІМІ

ЕКЗАМЕНАЦІЙНІ ЗАПИТАННЯ

з дисципліни «Програмування ч.1»

для студентів усіх форм навчання
напряму 6.050903 «Телекомунікації»

Електронний документ

ЗАТВЕРДЖЕНО
кафедрою ІМІ
протокол № 1
від 31.08.2017

Харків 2017 р.

Екзаменаційні запитання з дисципліни «Програмування ч.1» для студентів усіх форм навчання напряму 6.050903 «Телекомунікації»
[Електронний документ] / Упоряд.: О.П.Малінін. Харків: ХНУРЕ,
2017. 5 с.

Упорядник О.П.Малінін

ЗМІСТ

ВСТУП 2
1. Теоретична частина 2
2. Практична частина 3

ВСТУП

Для оцінювання роботи студента протягом семестру підсумкова рейтингова оцінка Осем розраховується як сума оцінок за різні види занять та контрольні заходи.

Формою підсумкового контролю для дисципліни БД є письмовий (комбінований) іспит. При цьому виді контролю підсумкова оцінка Рп обчислюється за формулою: Рп=0,6 · Осем+0,4 · Оісп, де Осем – оцінка за семестр у 100-бальній системі, Оісп – оцінка за іспит у 100-бальній системі.

Білет для іспиту складається з двох теоретичних запитань та однієї задачі.

Теоретичні запитання оцінюються по 30 балів, а задача – у 40 балів (в сумі – 100 балів).

1. ТЕОРЕТИЧНА ЧАСТИНА

1 Що таке мова програмування
2 Поняття алгоритму. Види алгоритмів
3 Блок-схеми. Графічна реалізація алгоритмів
4 Елементи програм на С ++. Структура програми
5 Синтаксис і семантика. Ідентифікатори
6 Дані та їх типи
7 Описи змінних. Основні арифметичні операції
8 Препроцесор
9 Арифметичні вираження
10 Потік управління. Умови та логічні вираження. Оператор “***SWITCH”***
11 Умовний оператор в формі ***«If-Then-Else»***
12 Умовний оператор в формі ***«If-Then»***
13 Вкладені умовні оператори
14 Висячі ***«else»***
15 Цикл ***«for»***
16 Цикл ***«while»***
17 Цикл ***«do..while»***
18 Оператор ***«break».*** Інструкція ***«goto»***
19 Одномірні масиви
20 Сортування масиву
21 Строки
22 Функції.
23 Функції бібліотеки «***cstring»***
24 Двовимірні масиви
25 Багатовимірні масиви
26 Безрозмірна "ініціалізація масивів
27 Масиви рядків
28 Покажчики та змінні, організація адресного простору в обчислювальних системах
29 Покажчики та масиви
30. Покажчики та рядки. методи адресації
31 Побітові операції

32 Робота з файлами
33 Динамічний розподіл пам'яті з використанням операторів ***«new»*** і ***«delete»***
34 Оператор ***"знак питання"***
35 Складові оператори присвоювання
36 Оператор ***"кома"***
37 Використання ключового слова ***«Sizeof»***
38 Загальний формат оголошення структур
39 Масиви структур
40 Створення функцій, які працюють зі структурою

2. ПРАКТИЧНА ЧАСТИНА
(приклади задач)

Завдання 1

Знайти максимальне число в тій частині матриці (8Х8), яке знаходиться вище головної її діагоналі і вивести його на екран монітора

Завдання 2

Знайти порядковий номер максимального числа, який знаходиться вище головної її діагоналі і вивести його на екран монітора

Завдання 3

Розробіть програму, яка виділила динамічну пам'ять для 10-елементного масиву типу double, який потім заповнюється значеннями від 10 до 19, після чого вміст цього масиву відображається на екрані.

Завдання 4

Знайти кількість «5» в матриці (8Х8) (8Х8), які знаходиться вище головної її діагоналі і цю кількість вивести на екран монітора

Завдання 5

Знайти мінімальне число в масиві М, яке знаходиться нижче головної її діагоналі і вивести його значення на екран монітора

Завдання 6

Знайти суму чисел, що входять в масив М, які знаходиться нижче головної її діагоналі і цю суму вивести на екран монітора

Завдання 7

Вивести кожен третій елемент масиву М. Вивід можливий тільки в тому випадку, якщо між виведеним числом і числом яке має бути виведено нема «0»

Завдання 8

Знайти суму 2 і 5 елементу якщо між ними немає «0», в іншому випадку знайти суму перших трьох чисел масиву М і цю суму вивести на екран монітора

Завдання 9

Переставити місцями перший і останній елементи масиву М, тільки в тому випадку якщо між ними є «0» в іншому випадку масив ранжувати, результат вивести на екран монітора.

Завдання 10

Написати програму, яка могла б порахувати кількість букв «А» в тексті, який знаходиться в пам'яті РС.