МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЕКОНОМІКО-ПРАВОВИЙ ФАКУЛЬТЕТ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Допустити до захисту

В.о. завідувача кафедри

_____ Мнацаканян М.С.

(підпис) *(ПІБ завідувача кафедри)*

«___»_____20__ р.

РОЗРОБКА СИСТЕМНИХ ВИМОГ ДО УМОВ ПРОГРАМУВАННЯ

Кваліфікаційна робота
здобувача вищої освіти
першого (бакалаврського) рівня вищої освіти
освітньо-професійної програми
«____Комп'ютерні науки_____»
(назва освітньо-професійної програми)

Гапотченко Кирило Андрійович
(прізвище, імя, по батькові здобувача вищої освіти)

Науковий керівник:
_Мартинюк Г.В., к.т.н., доцент__
(прізвище, ініціали, науковий ступінь, вчене звання)

Рецензент:
__Охріменко Т.О., к.т.н., НАУ_
(прізвище, ініціали, науковий ступінь, вчене звання, місце роботи)

Кваліфікаційна робота
захищена з
оцінкою_____
Секретар
ЕК_____ «____»
_____20__ р.

Київ -2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МАРІУПОЛЬСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЕКОНОМІКО-ПРАВОВИЙ ФАКУЛЬТЕТ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Рівень вищої освіти *бакалавр*
Шифр та назва спеціальності *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Комп'ютерні науки»*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри__к.т.н.__
(науковий ступінь, вчене звання)

_____ ___Мнацаканян М.С.___
(підпис) *(ПІБ завідувача кафедри)*

«__» _____ 20__ р.

ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

_____
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка системних вимог до умов програмування»

керівник роботи_Мартинюк Г.В., к.т.н., доцент_____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом Маріупольського державного університету від «___» _____20__ р. №__

2. Строк подання здобувачем роботи

.

3. Вихідні дані до роботи (мета, об'єкт, предмет):

Об'єкт дослідження – різні програми для програмування.

Предмет дослідження – язики програмування для створення програми.

Мета кваліфікаційної роботи – при програмуванні на машинній мові програміст повинен тримати під своїм контролем кожну команду, використовувати всі можливості машинних операцій.

4. Зміст роботи

Розділ 1. Аналіз програмного забезпечення та його складових.

Розділ 2. Аналіз сучасних мов програмування і програмного середовища,яке для них використовується.

Розділ 3.Практична реалізація кваліфікаційної роботи.

5. Дата видачі завдання:05.06.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Аналіз предметної області, порівняння можливостей різних мов програмування. Написання першого розділу.	5 днів	
2.	Опис необхідного програмного забезпечення та функцій програм для програмування Написання другого розділу	3 дні	
3.	Розробка коду на мові програмування та його тестування. Написання третього розділу	4 дні	
4.	Редагування пояснювальної записки кваліфікаційної роботи.	1 день	
5.	Проходження перевірки на плагіат.	1 день	
6.	Підготовка презентації.	1 день	

Здобувач _____ _____
(підпис) *(прізвище та ініціали)*

Науковий керівник роботи _____ _____
(підпис) *(прізвище та ініціали)*

Зміст

1. Вступ

На сучасному етапі розвитку комп'ютерних технологій неможливо уявити якогось висококваліфікованого фахівця, що не володіє інформаційними технологіями. Для вільної орієнтації в інформаційних потоках сучасний фахівець будь-якого профілю повинен уміти одержувати, обробляти і використовувати інформацію, перш за все, за допомогою комп'ютерів, а також телекомунікацій та інших новітніх засобів зв'язку, у тому числі і вміти - спілкуватися мовами програмування. Мова програмування - це система позначень, що служить для точного опису програм або алгоритмів для ЕОМ. Мови програмування є штучними мовами. Від природних мов вони відрізняються обмеженим числом "слів" і дуже строгими правилами запису команд (операторів). Можна сформулювати ряд вимог до мов програмування і класифікувати мови за їх особливостям. Основні вимоги, до мов програмування: ·наочність - використання в мові по можливості вже існуючих символів, добре відомих і зрозумілих як програмістам, так і користувачам ЕОМ; ·єдність - використання одних і тих же символів для позначення одних і тих же або споріднених понять у різних частинах алгоритму. Кількість цих символів має бути по можливості мінімальним; ·гнучкість - можливість відносно зручного, нескладного опису розповсюджених прийомів математичних обчислень за допомогою наявного в мові обмеженого набору образотворчих засобів; ·модульність - можливість опису складних алгоритмів у вигляді сукупності простих модулів, які можуть бути складені окремо і використані в різних складних алгоритмах; ·однозначність - недвозначність записи будь-якого алгоритму. Відсутність її могло б призвести до неправильних відповідей при вирішенні завдань. На даний момен у світі існує кілька сотень мов програмування. Кожна має свою область застосування. У залежності від ступеня деталізації приписів зазвичай визначається рівень мови програмуаання - чим менше деталізація, тим вищий рівень мови. За цим критерієм можна виділити наступні рівні мов програмування:

-Машинні;

-Машинно-орієнтіровані (ассемблери);

-Машинно-незалежні (мови високого рівня).

При програмуванні на машинній мові програміст повинен тримати під своїм контролем кожну команду, використовувати всі можливості машинних операцій. Але процес написання програми на машинній мові дуже трудомісткий і виснажливий, програма виходить громіздкою. Коли потрібно мати ефективну програму- замість машинних мов використовують близькі до них машинно-орієнтовані мови (ассемблери).

РОЗДІЛ 1. АНАЛІЗ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ЙОГО СКЛАДОВИХ

Щоб розв'язати конкретну задачу, комп'ютер повинен виконати повністю визначений набір послідовних операцій. Ці операції являють собою набір дій, які виконує центральний процесор. Необхідні дії та послідовність їх виконання задаються програмою. Комплекс програм, що забезпечують можливість використання комп’ютера для розв’язання різноманітних завдань, становить програмне забезпечення комп’ютера [1-4].

Все програмне забезпечення, яке використовується комп'ютером, ділиться на три основні види (рис. 1.1.)

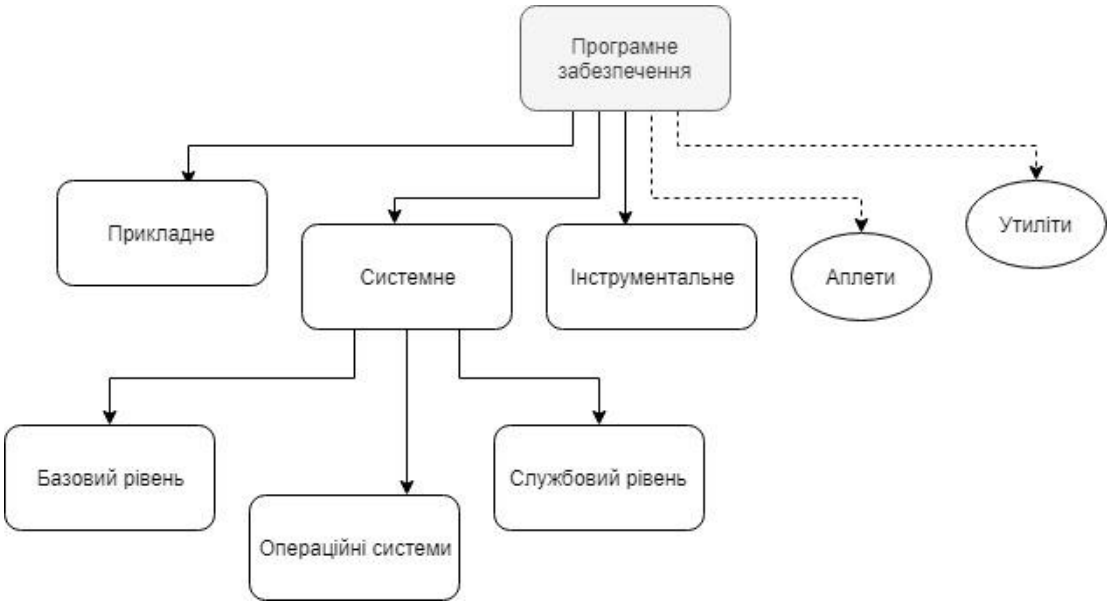


Рис. 1.1. Класифікація програмного забезпечення

Комп’ютер складається з двох однакових компонентів – апаратного та програмного забезпечення. Програмне забезпечення, на відміну від апаратного забезпечення, можна вважати змінною частиною комп'ютера. Програмне забезпечення поділяється на [2]:

- Системні:
  - Базовий рівень (прошивка) - драйвери

- Операційна система (ОС) — набір програм, що забезпечує базові програми та взаємодію інших програм із обладнанням
- Сервісний рівень - програми, що входять до ОС

- Прикладне — забезпечує виконання конкретних завдань на комп'ютері: захисту від вірусів, наукових, розважальних та інших. Наприклад, текстові та графічні редактори, файлові менеджери, веб-редактори, реєстратори даних, веб-браузери.
- Інструментальне (системне програмування) — програмне забезпечення, призначене для використання під час створення архітектури, розробки, оновлення та встановлення програм. Прикладом є середовище розробки.

Утиліти та аплети додатково виділені. Службові програми — це невеликі корисні програми з обмеженими функціями. Деякі утиліти постачаються разом з операційною системою. Як і програми, утиліти зазвичай встановлюються окремо і можуть використовуватися незалежно від решти операційної системи. Аплети іноді постачаються з операційною системою як супутні програми. Вони також можуть бути створені самостійно в процесі за допомогою Java або інших мов програмування.

1.1. Системне програмне забезпечення

Системне програмне забезпечення - це комплекс програм, призначених для забезпечення роботи комп'ютерів і комп'ютерних мереж, а також організації взаємодії користувача з комп'ютером.

Системне програмне забезпечення включає [3]:

- архіватори;
- оболонки операційних систем;
- програми обслуговування дисків;
- операційні системи;

- антивірусні програми;
- драйвери,
- програми обслуговування комп'ютерних мереж тощо.

Операційна система — це набір програм, що забезпечує [4]:

• управління комп'ютерною технікою та обмін даними між ними;

• зберігання даних в оперативній пам'яті та зовнішніх носіях;

• реалізація інших програм;

• розподіл комп'ютерних ресурсів між різними програмами, що працюють разом;

• організація обміну даними між користувачем і комп'ютером.

Класифікацію операційних систем наведено на рис. 1.2

Рис. 1.2. Класифікація операційних систем, які використовуються в системному програмуванні

У сучасних комп’ютерах використовуються операційні системи Windows, Linux, Unix, MacOS, NetWare, Palm OS тощо.

Сервісне програмне забезпечення включає: [2]:

1. Файловий менеджер (File Manager). Вони використовуються для виконання більшості завдань підтримки файлової структури, таких як копіювання, переміщення, перейменування файлів, створення каталогів (папок), знищення об’єктів, пошук файлів і навігація файловою структурою. Базові програмні засоби є частиною програм системного рівня та встановлюються разом з операційною системою.

2. Засоби стиснення даних (архіватори). Призначені для створення архівів. Зросла щільність запису інформації в архівних файлах і, відповідно, більш ефективно використовуються носії інформації.

3. Діагностичні засоби. Призначені для автоматизації процесів діагностики програмного та апаратного забезпечення. Вони використовуються для виправлення помилок і оптимізації роботи комп'ютерних систем.

4. Інсталяційна програма. Призначена для керування додаванням нового програмного забезпечення до поточної конфігурації програмного забезпечення. Вони відстежують стан і зміни навколишнього програмного середовища, відстежують і записують утворення нових зв'язків, втрачених під час знищення певних програм. Прості елементи керування для встановлення та видалення програм включені в операційну систему, але можна використовувати додаткові утиліти.

5. Засоби зв'язку. Дозволяють встановлювати з’єднання з віддаленим комп’ютером, надсилати повідомлення електронної пошти, пересилати факси тощо.

6. Засоби перегляду та відтворення. Для роботи з більшістю файлів їх потрібно завантажити в «рідну» прикладну систему та внести необхідні зміни. Але, якщо редагування не потрібне, існують універсальні засоби для перегляду (у випадку тексту) або відтворення (у випадку аудіо чи відео) даних.

7. Засоби безпеки комп’ютера. До них належать засоби пасивного та активного захисту даних від пошкодження, несанкціонованого доступу, перегляду та модифікації даних. Інструменти пасивного захисту — це службові програми, призначені для резервного копіювання. Активний захист означає використання антивірусного програмного забезпечення. Для захисту даних від несанкціонованого доступу, перегляду та модифікації використовуються спеціальні системи, засновані на криптографії.

1.2. Класифікація прикладного програмного забезпечення

Прикладне програмне забезпечення призначене для користувачів, які зазвичай не створюють власних програм, а лише використовують програмні засоби для вирішення певних завдань. На відміну від програмістів, таких користувачів називають «кінцевими користувачами». Передбачається, що вони є реальними споживачами інформації, яка зосереджена в пам'яті комп'ютера або може бути згенерована під час роботи прикладних програм. Під час спілкування з прикладною системою користувачеві іноді потрібно виконати деякі прості операції – введення чисел і тексту, перегляд даних, відображення графіків і зображень на екранах і зовнішніх пристроях тощо. Прикладні програмні засоби створені таким чином, щоб створити користувачеві максимальний комфорт при виконанні дій і при цьому не вимагають надмірних навичок і спеціальних знань, які безпосередньо не пов'язані з його професійними інтересами. Існує два класи прикладного програмного забезпечення, що використовується на персональних комп’ютерах [5]:

- прикладні пакети і програми загального призначення;
- проблемно-орієнтовані пакети і програми.

Загальну класифікацію прикладного програмного забезпечення наведено на рис. 1.3.

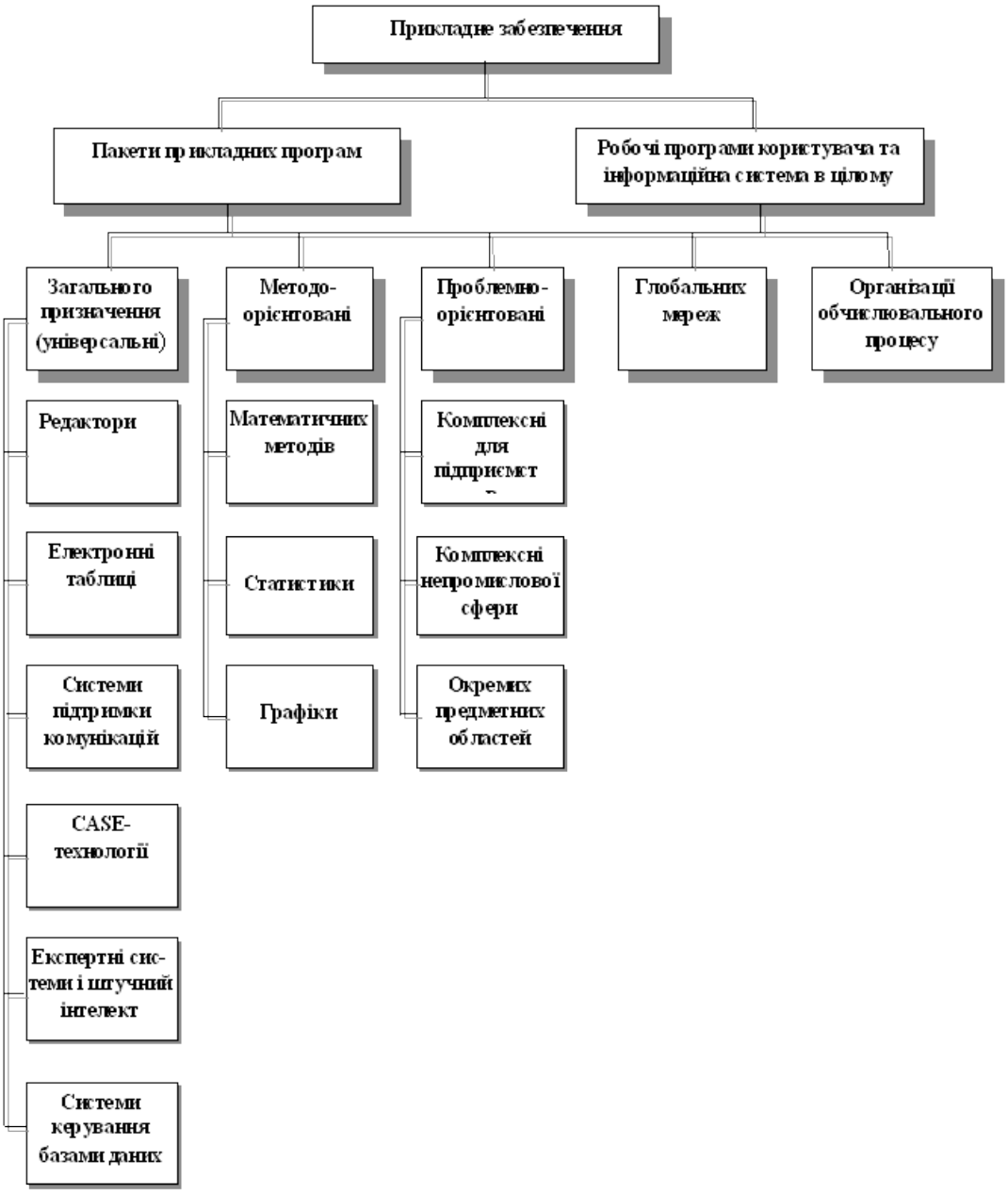


Рис. 1.3. Класифікація прикладного програмного забезпечення

У стандартному прикладному програмному забезпеченні загального застосування для управлінської діяльності слід виділити:

- *Система підготовки текстових документів (текстовий редактор, текстовий процесор, настільна видавнича система).* Призначені для

виготовлення управлінських документів і різноманітних інформаційних матеріалів текстового характеру.

- *Системи обробки фінансово-економічної інформації (універсальні табличні процесори, спеціальні бухгалтерські програми, спеціальні банківські програми для внутрішніх і міжбанківських розрахунків, спеціальні програми для фінансово-економічного аналізу та планування).* Вони призначені для обробки фінансово-економічної інформації, представленої у вигляді числових даних про різноманітні виробничо-економічні та фінансові явища та об'єкти, а також для підготовки відповідних управлінських документів та інформаційно-аналітичних матеріалів.
- *Системи керування базами даних* призначені для створення, зберігання та обробки великих обсягів даних. Різні системи цього класу відрізняються методами зберігання даних і обробки інформаційно-пошукових запитів, а також характером інформації, що зберігається в базі даних.
- *Персональна інформаційна система (програма-секретар).* Вони дозволяють планувати особистий час, своєчасно нагадувати про початок запланованих заходів, вести особисті та інші картки з можливістю автоматичного відбору інформації, телефонувати, записувати телефонні розмови. завдання Багатофункціональний телефон Ведіть особистий блокнот для збереження різної особистої інформації.
- *Системи підготовки презентацій* призначені для ефективної підготовки графічних і текстових матеріалів, які будуть використовуватися з метою презентації на презентаціях, ділових переговорах, конференціях. Сучасні технології поєднуються з традиційною графікою та текстом відео- та аудіоінформації, що дозволяє говорити про створення гіпермедійних технологій.

- *Система управління проектами* призначена для планування та управління різними видами ресурсів (матеріально-технічними, фінансовими, кадровими, інформаційними) при виконанні комплексних науково-дослідних і проектно-будівельних робіт.
- Для реалізації технологій інформаційного забезпечення процесів прийняття управлінських рішень на основі використання методів економіко-математичного моделювання та принципів штучного інтелекту створено *експертні системи та системи підтримки прийняття рішень*.
- *Системи інтелектуального проектування та вдосконалення управління,* призначених для впровадження технологій CASE (Computer Aided Systems Engineering*),* орієнтованих на автоматизовану розробку проектних рішень для створення та вдосконалення процесів формування нормативно-методичних матеріалів. підготовка та оформлення управлінських та інших документів у межах конкретного завдання із забезпечення управлінської діяльності; інструкційно-нормативні матеріали з експлуатації технічного обладнання (включаючи техніку безпеки праці та умови підтримання нормальної працездатності обладнання); інструктивні та нормативно-методичні матеріали з організації роботи керівного і технічного персоналу в рамках конкретних інформаційних технологій управлінської діяльності.
- *Системи підтримки зв'язку* необхідні для підключення до комп'ютера різних типів зовнішніх пристроїв, для організації зв'язку між комп'ютерами, а також для підтримки їх роботи в локальній мережі. Ці програми дозволяють задавати режим роботи послідовних каналів, а також виконувати більш складні функції - такі як пошук потрібного абонента в телефонному довіднику, автоматичний набір телефонних номерів, автовідповідь та інші.

Проблемно-орієнтовані пакети і системи, на відміну від програм загального призначення, мають або вузьку сферу застосування, або розраховані на спеціалістів певного профілю. Кількість таких програм для персональних комп'ютерів на даний момент становить кілька тисяч. Вони використовують спеціальні методи подання та обробки даних з урахуванням специфіки конкретних завдань.

Основні тенденції розвитку прикладного програмного забезпечення тісно пов'язані зі створенням і переходом до інформаційних систем четвертого покоління з ієрархічною структурою, в якій центр ваги переноситься з локальної мережі кінцевих користувачів на мережу локальних серверів. . , В основі інформаційних систем четвертого покоління лежить необхідність скорочення оперативних ресурсів при одночасному збільшенні масштабованості системи та розширенні діапазону її функціональних можливостей.

Остання обставина особливо важлива, оскільки спостерігається стійка тенденція до майже стовідсоткової інтеграції різних функціональних підсистем інформаційних технологій в єдину бізнес-модель підприємства, яка відповідає досить великій кількості вимог (іноді суперечливих) існування. передбачає. Для інформаційних систем з боку кінцевих користувачів, неоднорідних за своїми кваліфікаціями та професійними функціями.

Прикладні програми спеціального призначення використовуються для певної діяльності користувача.

Функції конкретних систем залежать від їх призначення. Наприклад, для освітніх систем це можуть бути допоміжні засоби для розробки комп’ютерних уроків (гіпермедійні та гіпертекстові системи, авторські та інші системи), програми імітаційного моделювання для навчальних цілей, програми для розробки та підтримки програм Навчальні програмні засоби різного призначення тощо.

До прикладних програм спеціального призначення також можна віднести пакети прикладних програм (ППП), які широко використовуються, наприклад, для статистичної обробки даних, бухгалтерського обліку, розрахунку будівельних конструкцій тощо. Наявність у комп'ютерах різноманітних ППП дозволяє розв'язувати значну частину простих прикладних задач практично без програмування. У цьому випадку завдання на вирішення тієї чи іншої задачі записується у вигляді інструкцій спеціальною проблемно-орієнтованою мовою і повідомляється в комп'ютер.

1.3. Класифікація систем програмування

Системи програмування є невід'ємною частиною сучасного світу інформаційних технологій. Вони дозволяють створювати складні програмні продукти, обробляти великі обсяги даних, автоматизувати процеси в різних сферах діяльності. У цій статті ми розглянемо основні аспекти систем програмування: їх види, функціональні можливості та призначення.

Системи програмування **—** це набір програмних і апаратних засобів, призначених для розробки, тестування, налагодження та обслуговування програмного забезпечення [3]. Вони містять різні інструменти та ресурси для створення програмних продуктів, таких як середовища розробки, компілятори, налагоджувачі та тестові структури.

Існує багато різних типів систем програмування, кожна з яких призначена для виконання певних завдань. Наприклад, існують системи програмування для розробки веб-додатків, мобільних додатків, системи управління базами даних тощо.

Вони важливі в сучасній індустрії програмного забезпечення, оскільки дозволяють більш ефективно створювати складні програмні продукти та скорочувати час, витрачений на розробку. Вони допомагають розробникам

спростити процес створення програм і підвищити їх якість завдяки різноманітним інструментам і функціям, доступним у цих системах.

Існує велика кількість різних систем програмування, і точну кількість визначити важко. Крім того, кожен день з'являються нові системи програмування, оскільки розробники прагнуть спростити і прискорити процес розробки програмного забезпечення, а також підвищити його якість.

Наприклад, система програмування Microsoft Visual Studio використовується для розробки додатків для операційної системи Windows, а система програмування Xcode використовується для розробки додатків для iOS і macOS.

Система програмування складається з кількох компонентів, включаючи мову програмування, компілятор, інтерпретатор, відладчик, текстовий редактор, систему контролю версій та інші засоби розробки. Мова програмування визначає синтаксис і семантику мови, компілятор перетворює вихідний код мови програмування в машинний код, інтерпретатор виконує вихідний код мови програмування, а налагоджувач виявляє помилки в коді програмування. Допомагає знайти та виправити.

Для написання коду використовується текстовий редактор, а система контролю версій дозволяє відстежувати зміни коду та співпрацювати над проектом. Усі ці компоненти разом утворюють структуру програмування, яка допомагає розробникам створювати якісне та ефективне програмне забезпечення.

До основних інструментів відносять [5]:

- Інтегроване середовище розробки (IDE) є найпопулярнішим типом системи програмування. IDE — це комплексний програмний пакет, який поєднує в собі редактор вихідного коду, компілятор, налагоджувач і багато інших інструментів, необхідних для розробки програмного забезпечення. Приклади таких систем: Visual Studio, Eclipse, NetBeans тощо.

- Текстові редактори (програми, призначені для написання вихідного коду програм певною мовою програмування) дозволяють редагувати вихідний код, виділяти синтаксичні структури різними кольорами, використовувати автозаповнення коду тощо. Приклади текстових редакторів: Sublime Text, Atom, Notepad++ тощо.
- Компілятори — це програми, які перетворюють вихідний код програми в машинний код на мові програмування, який можна виконати на комп'ютері. Прикладами компіляторів є GCC, Clang, Microsoft Visual C++ тощо.
- Бібліотеки — це набори програмних модулів, які надаються для повторного використання. Вони містять функції та класи, які можна викликати з програмного коду.
- Інтерпретатори — це програми, які виконують вихідний код програми мовою програмування, переводячи його в машинний код під час виконання програми. Інтерпретатори не вимагають попередньої компіляції вихідного коду, що дозволяє швидше отримувати результати. Приклади інтерпретаторів: Python, Ruby, Perl тощо.
- Система контролю версій — використовується для контролю версій вихідного коду програми та дозволяє відстежувати зміни коду, створювати гілки розробки, об'єднувати зміни з різних гілок і багато іншого. Прикладами систем контролю версій є Git, SVN, Mercurial тощо.

Існує багато систем програмування, кожна з яких призначена для вирішення певних завдань.

Системи програмування призначені для полегшення процесу створення програмного забезпечення та спрощення життя програмістів. Вони дозволяють швидше писати та налагоджувати код, керувати проектами та ділитися кодом з іншими програмістами. Без системи програмування створення складного програмного забезпечення було б більш трудомістким і дорогим процесом.

Однією з основних цілей систем програмування є підвищення продуктивності процесу розробки та зниження вартості створення програмного забезпечення. Крім того, система програмування також:

- дає можливість швидко створювати та тестувати нові додатки, а також ефективно підтримувати існуючі;
- використовується для розробки програмного забезпечення різної складності - від невеликих скриптів до великих додатків, які працюють на багатомільйонну аудиторію (наприклад, система програмування Eclipse часто використовується для створення великих додатків у банківському та фінансовому секторах);
- можна використовувати для різних цілей, таких як створення ігор, веб-додатків, наукових програм тощо.

Важливо відзначити, що використання систем програмування дає змогу прискорити та спростити процес створення програмного забезпечення, знизити ризик помилок та підвищити його якість. Тому ці системи є невід'ємною частиною сучасної індустрії програмного забезпечення та важливим інструментом для розробників і програмістів усіх рівнів.

Важливим напрямком розвитку обчислювальної техніки є розробка та вдосконалення систем програмування. Системи програмування дозволяють створювати і підтримувати великі проекти, прискорюють процес розробки, підвищують якість і надійність програмного забезпечення. Завдяки постійному вдосконаленню систем програмування розробники програмного забезпечення зможуть вдосконалювати свої продукти, створювати нові продукти, розширювати можливості комп'ютерної техніки в цілому.

1.4 Висновки до розділу 1

Ознайомлення с програмами за їх складовими для наступних робот с мовами програмування

РОЗДІЛ 2. АНАЛІЗ СУЧАСНИХ МОВ ПРОГРАМУВАННЯ І ПРОГРАМНОГО СЕРЕДОВИЩА, ЯКЕ ДЛЯ НИХ ВИКОРИСТОВУЄТЬСЯ

Мова програмування – це штучна мова, призначена для написання комп'ютерних програм. Мови програмування класифікуються за такими основними критеріями [6-8]:

- за рівнем абстракції:
  - мови програмування низького рівня (машинно-орієнтовані);
  - мови програмування високого рівня;
- за областю застосування:
  - універсальні;
  - спеціалізовані;
- за парадигмами програмування, які підтримуються
  - об'єктно-орієнтовані,
  - логічні,
  - функціональні тощо.

Даний розділ присвячений більш детальному опису мов програмування та інтерфе

2.1. Мови програмування та їх функціональні можливості

2.1.1. Мови програмування низького рівня

Мови програмування низького рівня (*машинно-орієнтовані мови*) - такі, у яких принципи керування і структура даних безпосередньо відображають архітектуру ЕОМ. Тобто, такі мови орієнтовані на конкретний тип процесора і враховують його особливості.

Група мов низького рівня включає машинні мови (машинні коди), *мови символічного кодування* (Асемблери) та ряд інших. Програми, написані на

таких мовах програмування, представляють собою лінійні послідовності елементарних операцій з регістрами, в яких зберігаються дані.

Зокрема, мова Асемблера представляє кожну команду машинного коду у вигляді спеціальних символьних позначень, які називаються *мнемоніками*, а в якості операндів використовує символічні імена, а не конкретні адреси (табл. 2.1). Це допомагає програмісту легше запам'ятовувати смисловий зміст операції та забезпечує суттєве зменшення кількості помилок при складанні програм.

Таблиця 2.1 Приклади команд на мові асемблера

Операція	**Операнди**	**Примітка**
MOV	AX,	Записати значення, що міститься за адресою WORDB,
MOV	BX,	Записати значення, що міститься за адресою WORDC,
ADD	AX, BX	Додати значення регістру BX до значення регістру
MULL	WORDD	Помножити значення регістру AX на значення, що
MOV	WORDA,	Записати значення регістру AX за адресою WORDA

У табл. 2.1 наведено фрагмент машинної програми асемблера для обчислення виразу a = (b + c) d

Оскільки різні моделі мікропроцесорів мають різні системи інструкцій, специфічна архітектура комп’ютера відповідає власній мові асемблера, і програми, написані на ньому, можна використовувати лише на тих комп’ютерах, для яких він розроблений. Тому асемблери вважаються машинно-орієнтованими мовами програмування.

Асемблери використовуються професійними системними програмістами з метою використання всіх можливостей комп'ютерної техніки та отримання ефективних програм за часом виконання та необхідному об'єму пам'яті. На цих мовах зазвичай розробляються порівняно невеликі програми, які є частиною системного програмного забезпечення: драйвери, утиліти тощо..

2.1.2. Мови програмування високого рівня

Мови програмування високого рівня - ті, в яких засоби управління та роботи з даними відображають потреби програміста, а не можливості апаратної частини. Програми, написані на таких мовах, являють собою послідовності операторів, структурованих відповідно до правил мови. Вони працюють із ближчими та зрозумілішими людині сутностями, такими як змінні, функції тощо. Специфіка конкретної архітектури комп'ютера не враховується, тому програми, розроблені на цих мовах, можуть виконуватися на інших комп'ютерах тієї ж платформи.

Набагато легше розробляти програми на мовах програмування високого рівня за допомогою зрозумілих команд, і при їх розробці допускається дуже мало помилок. При цьому програмісти отримали можливість не детально описувати обчислювальний процес на рівні машинних команд, а зосередитися на основних особливостях алгоритму. Такі мови програмування часто називають алгоритмічними.

Наприклад, сегмент програми C/C++ для обчислення виразу $a = (b+c)\cdot d$:

```
a = (b + c)*d;
```

Ключові відмінності між мовою програмування високого рівня та машинно-орієнтованою мовою [10]:

- алфавіт алгоритмічної мови набагато ширший, ніж алфавіт машинно-орієнтованого, що значно підвищує наочність тексту такої програми;
- множина допустимих операцій в алгоритмічній мові не залежить від множини машинних операцій, а вибирається для зручності формулювання послідовності дій;
- формат речень в алгоритмічній мові є досить гнучким, що дозволяє одним реченням визначити досить змістовний етап обробки даних;

- необхідні операції в алгоритмічній мові задаються в зручному для людини вигляді, наприклад, за допомогою загальноприйнятих математичних записів;
- у алгоритмічній мові можна надати більший набір типів даних, ніж набір машинних типів даних.

Чіткість і потужні можливості мов програмування високого рівня визначили їх використання для вирішення різноманітних прикладних завдань.

Мови програмування, які призначені для опису алгоритму, що виконується комп'ютером, називаються алгоритмами. Наприклад, наступні мови алгоритмічного програмування високого рівня.

Мова програмування С. Вона була створена Д. Річі на початку 1970-х років для розробки операційної системи UNIX. Засоби для роботи безпосередньо з пам'яттю. Вона була задумана як мова системного програмування на заміну асемблеру, здатна створювати настільки ж ефективні та менші програми, але не залежати від конкретного процесора. Це найпопулярніша мова для створення системного програмного забезпечення. Однак великий набір операцій і типів даних, сучасний дизайн і високий ступінь машинної незалежності зробили його привабливою мовою програмування загального призначення. Незважаючи на те, що С не розроблявся для початківців, він активно використовується для навчання програмуванню. Пізніше синтаксис мови Сі став основою багатьох інших мов. Багато прикладних і системних програм і багато відомих ОС (зокрема, Unix) написані на мові Сі.

Більш загальні мови програмування, не обов’язково алгоритмічні за своєю природою, є подальшим розвитком ідеї алгоритмічної мови. Подібно до алгоритмічних мов, такі мови також спрямовані на кінцеве отримання машинних програм, але в багатьох випадках їх тексти допускають певну свободу у виконанні і, як правило, лише для синтезу шуканих алгоритмів.надають зміст, а не самі алгоритми.

Мова програмування C++. У 1980 році створене Б. Страуструпом об’єктно-орієнтоване розширення мови С. Вона поєднує в собі функції мов високого та низького рівня. На основі використання класів і об'єктів.

Створюючи мову С++, він намагався зберегти її синтаксис сумісним з мовою С. Більшість програм, написаних мовою С, добре працюють із компілятором мови С++. Інновації мови С++ порівняно з мовою С включають підтримку об’єктно-орієнтованого програмування через класи та об’єкти, підтримку узагальненого програмування через шаблони, доповнення до стандартної бібліотеки, додаткові типи даних, обробку винятків), простори імен, вбудовані модулі. -in функції, незалежні від операторів і назв функцій, посилання та перевизначення операторів для управління розподіленою пам'яттю.

Мова програмування Java. Вона була створена компанією Sun Microsystems на початку 90-х на основі С++. Він розроблений, щоб спростити розробку додатків на С++, видаливши з нього функції низького рівня. Головна особливість мови полягає в тому, що він переводить програми не в машинний код, а в платформно-незалежний байт-код (кожна команда займає один байт). Цей байт-код можна виконати за допомогою інтерпретатора - віртуальної Java-машини JVM (Java Virtual Machine), версії якої створюються для будь-якої апаратної платформи. Завдяки цьому Java-програми можна портувати не тільки на рівні вихідних текстів, а й на рівні двійкового байт-коду. Це дозволяє створювати незалежні програмні модулі, здатні працювати на серверах глобальної та локальної мереж з різними ОС.

Мова програмування C#. Належить до сімейства мов із С-подібним синтаксисом, з яких його синтаксис найбільш близький до С++ і Java.

Мова програмування Perl. Інтерпретована мова програмування загального призначення високого рівня. Головною особливістю мови є можливість роботи з текстом, у тому числі реалізованим за допомогою регулярних виразів..

Мова програмування Python. Мова програмування загального призначення з акцентом на продуктивність розробника та читабельність коду. Синтаксис ядра Python мінімалістичний. Також стандартна бібліотека містить велику кількість корисних функцій.

Мова програмування Ruby. Інтерпретована мова програмування високого рівня для швидкого та зручного об'єктно-орієнтованого програмування. Мова має незалежну від операційної системи реалізацію багатопоточності та багато інших функцій. За синтаксисом Ruby близький до Perl, а за об'єктно-орієнтованим підходом до Smalltalk. Крім того, деякі особливості мови були взяті з Python, Lisp та інших.

Мови програмування також можна розділити на універсальні та спеціалізовані: універсальні мови використовуються для вирішення різноманітних завдань; Спеціалізовані - призначені для вирішення одного або кількох видів завдань. Наприклад, робота з базами даних, веб-програмування, написання скриптів для адміністрування операційної системи.

Мова програмування PHP. Мова сценаріїв загального призначення, яка інтенсивно використовується для розробки веб-додатків. Наразі підтримується більшістю хостинг-провайдерів і є однією з мов програмування, що використовується для створення динамічних веб-сайтів.

Мова програмування ActionScript. Об’єктно-орієнтована мова програмування, яка додає інтерактивність, обробку даних тощо до вмісту програм Flash. ActionScript виконується віртуальною машиною ActionScript, яка є компонентом Flash Player. ActionScript компілюється в байт-код, який міститься у файлі SWF.

2.1.3. Покоління засобів програмування

Сьогодні люди вже говорять про різні техніки програмування та відповідні засоби підтримки мови. Загалом, з точки зору функціональності, виділяють п'ять поколінь засобів програмування [8]:

1. Асемблер, побудований за принципом "одна інструкція - один рядок".
2. Символічний асемблер, в якому з'явилося поняття змінних. Вона стала першою повноцінною мовою програмування. Завдяки його появі значно зросла швидкість розробки та надійність програм.
3. Універсальні мови високого рівня, які можна використовувати для вирішення будь-якої прикладної задачі. Вони характеризуються відносною простотою, незалежністю від конкретного ПК, можливістю використання потужних синтаксичних структур (Fortran, COBOL, A1go1, PL/1, BASIC, Pascal, C/C++, Java, ...).
4. Проблемно-орієнтовані мови, призначені для реалізації великих проектів, підвищення їх надійності та швидкості створення. Як правило, вони мають вбудовані потужні оператори, які дозволяють одним рядком описати функціональність, для реалізації якої в молодших поколіннях мов потрібні тисячі рядків коду. зосереджені на окремих сферах застосування, де хороших результатів можна досягти, використовуючи мови, які працюють з конкретними концепціями вузької предметної області (Prolog - мова для логічного програмування, SQL - мова програмування бази даних, HTML - мова розмітки) Інтернет, UML - мова графічного моделювання).
5. Система автоматизованого створення прикладних програм за допомогою засобів візуальної розробки (середовище RAD). Характеризується можливістю автоматичного формування результуючого тексту на універсальних мовах програмування (Delphi, Borland C, MS Visual Studio та ін.)..

Сьогодні кількість мов програмування вже вимірюється тисячами і продовжує зростати.

2.2. Проблеми вибору середовища програмування

Технічні характеристики обчислювальної техніки надзвичайно важливі для вибору програмного середовища. Якщо комп'ютерні класи в школі мають низькі характеристики, то на них не можна встановити сучасну операційну систему, яка вимагає потужних ресурсів. Сучасні програмні середовища, наприклад, Microsoft Visual Studio версій 2013, 2015 і 2016 років, мають високі системні вимоги. Для комфортної роботи в цьому програмному середовищі комп'ютер повинен мати досить високі апаратні характеристики, і це негативно впливає на вибір даного програмного середовища - на третині комп'ютерів це середовище використовувати недоцільно. Комп'ютер буде виконувати завдання компіляції повільно, «зависати», втрачати програми. Як правило, це спричиняє зниження інтересу учнів до вивчення мови програмування, а отже, і ефективності процесу навчання програмуванню.

Доцільніше вибрати програмне середовище, яке має менші системні вимоги. На мою думку, системні вимоги до середовища програмування повинні бути приблизно такими: процесор 2-2,2 ГГц, до 1 ГБ оперативної пам'яті, до 1 ГБ вільного місця на диску, операційна система Microsoft Windows XP / Seven / 8.1 / 10. вищі системні вимоги цих комп’ютерів значно зменшують межі, на яких ви можете комфортно працювати в середовищі програмування. Низькі системні вимоги - це, як правило, середовища програмування з обмеженою функціональністю. Цей фактор зменшує можливості використання в навчальному процесі потужних середовищ програмування з високими системними вимогами.

Важливу роль відіграє і другий фактор - наявність операційної системи і додаткового програмного забезпечення. Більшість програмних середовищ вимагають наявності додаткового програмного забезпечення, встановленого на комп’ютері – Microsoft Framework Program Execution Environment

(Microsoft VisualStudio, SharpDevelop, Delphi) або віртуальної машини Java Development Kit (Eclipse, NetBeans, JBuilder та інші). Додаткове програмне забезпечення зазвичай знаходиться у вільному доступі на сайті компанії - виробника програмного середовища або операційної системи. Однак можливі конфлікти між різними програмами, що вільно поширюються.

Важливу роль при виборі програмного середовища відіграє його функціональність. Щоб функціонувати як ефективний інструмент навчання, функціональність програмного забезпечення повинна відповідати змісту навчання. Однак бувають випадки, коли більша функціональність програмного середовища стимулюватиме студентів до подальшого самостійного вивчення програмування. Наприклад, повна версія Microsoft Visual Studio Standard Edition платна, і ціна її досить висока. Безкоштовна версія Microsoft Visual Studio Express Edition має деякі обмеження щодо функціональності, але ці обмеження не відіграють великої ролі під час навчання, оскільки стосуються лише тих функцій, які зазвичай не вивчаються в школі.

Обмежений функціонал програмного середовища SharpDevelop майже не впливає на його дидактичні можливості, оскільки він стає важливим лише під час розробки складних програм, які взаємодіють з базами даних тощо. Однак функціональності версій SharpDevelop, починаючи з версії 4.0 і вище, цілком достатньо для навчання програмуванню на мові C# і створення програм зі складним алгоритмом. Іншим прикладом є Eclipse і NetBeans. У середовищі Eclipse ви можете розробляти повнофункціональні програми на Java та конвертувати програму у файл EXE, але неможливо розмістити компоненти на екрані у візуальному режимі. Середовище NetBeans має візуальний режим роботи з екранними формами, але немає функції конвертації програми у файл EXE.

При виборі програмного середовища слід також враховувати особливості інтерфейсу. Має включати функції [10]:

− тип інтерфейсу (графічний або командний рядок);

- раціональне компонування інтерфейсу. Доступ до основних функцій і підбазових компонентів має бути максимально простим;

- мова інтерфейсу значно полегшує процес засвоєння прийомів роботи в програмному середовищі;

- наявність підказки значно спрощує процес виправлення помилок. Тому студентам-початківцям доцільно використовувати програмне середовище з україномовним інтерфейсом. Для студентів, які мають навички програмування, можна використовувати програмне середовище з англомовним інтерфейсом;

- наявність документації на програмне середовище. Цей фактор впливає на швидкість засвоєння викладачем програмного середовища;

- поширеність програмного середовища в навчальних закладах залежить від наявності відповідного методичного забезпечення.

Однією з особливостей програмних середовищ є можливість роботи з візуальними компонентами, що дозволяє розробляти програми у візуальному режимі. Використання консольного введення-виведення ускладнює процес навчання студентів основам роботи в середовищі програмування та значно уповільнює процес розробки програм. Це негативно впливає на рівень інтересу студентів до програмування. Навпаки, графічний інтерфейс сприяє активізації роботи учнів. Учні змінюють розташування компонентів у формі й одразу бачать внесені зміни. Це допомагає підтримувати інтерес студентів до програмування на необхідному рівні та надихає на подальшу роботу.

Раціональне компонування інтерфейсу позитивно впливає на вивчення програмного середовища та процес розробки програм студентами. Графічний інтерфейс середовища Delphi можна вважати одним із найбільш раціонально розроблених. Основні компоненти екранної форми розміщені на вкладках, їх властивості доступні у вікні Інспектора об’єктів. У процесі оформлення форми на екрані студент може встановити форму на екрані так, як йому

зручно. У середовищах Microsoft Visual Studio та SharpDevelop положення форм на екрані не можна змінити під час розробки програми, а доступ до компонентів форми та їх властивостей утруднений. Це зменшує цінність цих середовищ програмування для навчання студентів програмуванню.

Українськомовний інтерфейс значно спрощує процес вивчення основ роботи в середовищі програмування та знижує поріг входження до мови програмування. Крім того, мова інтерфейсу не настільки важлива для програмного середовища, як для прикладного програмного забезпечення, наприклад, текстових редакторів. У середовищі програмування учень зазвичай працює з програмним кодом, а не з форматуванням тексту. Кількість елементів інтерфейсу досить обмежена, тому запам'ятати їх досить просто. Тому мова інтерфейсу є скоріше початковим бар'єром, після подолання якого вплив мови інтерфейсу на успішність вивчення мови програмування поступово зменшується.

Наявність підказки значно підвищує цінність програмного середовища як засобу навчання. У процесі навчання прийомам написання програм учні припускаються значної кількості помилок. Дуже важливо, щоб вони навчилися самі виправляти помилки. Українськомовна чи російськомовна підказка спрощує процес розвитку навичок пошуку та виправлення помилок у програмах. Це знижує бар’єр входу під час вивчення мови програмування.

Документація для програмного середовища може мати форму інструкцій користувача або навчального посібника з використання цього середовища. У більшості випадків наявність документації важливіша для викладача, ніж для студентів. Більшість студентів не користуються документацією. Для вчителя наявність такої документації дає змогу швидше опанувати середовище програмування і, відповідно, швидше навчити студентів програмуванню. Для оволодіння учнями навичками, необхідними для роботи в середовищі програмування на просунутому рівні, важлива також

документація від розробника, оскільки підручники та навчальні посібники зазвичай не розкривають весь функціонал програмних засобів.

2.3. Класифікація сучасного середовища програмування

Важливим компонентом у процесі розробки програмного забезпечення є вибір правильної IDE, яка залежить не лише від платформи, але й від важливості особистого навчання. Інтегроване середовище розробки, ICP (Integrated Development Environment-IDE) — набір програмних засобів, що використовуються програмістами для розробки програмного забезпечення [1].

Вважається, що впровадження ICP для розробки програмного забезпечення є прямим контрастом до методології, у якій різні інструменти, такі як текстові редактори, компілятори тощо, поєднуються з простими інтерфейсами користувача. На відміну від дискретних програм розробки, це дозволяє розробнику виконувати менше дій для перемикання між різними режимами. Але оскільки ICP є досить складним програмним комплексом, то середовище розробки зможе якісно прискорити процес розробки програмного забезпечення лише після спеціального навчання. Щоб знизити вхідний бар’єр, багато з них досить інтерактивні, а для полегшення переходу від одного до іншого інтерфейс виробника є максимально наближеним, аж до використання ICP.

ICP зазвичай є окремою програмою, у якій виконується вся розробка. Зазвичай він складається з кількох функцій для створення, модифікації, компіляції, розгортання та налагодження програмного забезпечення. Метою інтегрованого середовища є об'єднання різних утиліт в єдиний модуль, який дозволить абстрагуватися від виконання допоміжних функцій, дозволить програмістам сконцентруватися на вирішенні реальної алгоритмічної задачі, а не витрачати час на виконання конкретних технічних дій. наприклад, виклик компілятора). Таким чином підвищується продуктивність розробника. Також

вважається, що більш тісна інтеграція завдань розробки може ще більше підвищити продуктивність, дозволяючи вводити додаткові завдання на проміжних етапах роботи. Наприклад, ICP дозволяє аналізувати код і таким чином надавати негайний зворотний зв'язок і повідомляти про синтаксичні помилки.

Більшість сучасних ICP є графічними. Але перші IC використовувалися ще до того, як операційні системи з графічним інтерфейсом набули широкого поширення - вони базувалися на текстовому інтерфейсі з використанням функціональних і гарячих клавіш для виклику різних функцій (наприклад, Turbo Pascal від Borland).

Microsoft Visual Studio — інтегроване середовище розробки, ціна якого коливається від 699 до 2900 доларів. Існує кілька версій цієї IDE, здатної створювати будь-які програми, від веб-додатків до мобільних додатків і відеоігор. Ця лінія програмного забезпечення включає різноманітні інструменти перевірки сумісності. Завдяки своїй гнучкості Visual Studio є чудовим інструментом для студентів і професіоналів. Один з найстаріших програмних продуктів для створення консольних додатків і має графічний інтерфейс. Додавання сторонніх плагінів дозволяє серйозно розширити функціональність середовища, включаючи кросплатформне позиціонування. Підтримувані мови: Ajax, ASP.NET, DHTML, JavaScript, JScript, Visual Basic, Visual C#, Visual C++, Visual F#, XAML тощо. Інтерфейс Microsoft Visual Studio показаний на рис. 2.1 [2].

Особливості Microsoft Visual Studio:

– величезна бібліотека розширень, яка постійно поповнюється;

- Intellisense;

- регулюється панель і кріпляться вікна;

- простий робочий процес та ієрархія файлів;

- статистика моніторингу продуктивності в режимі реального часу;

- засоби автоматизації;

- легкий рефакторинг і злиття фрагментів коду;

- підтримка розділеного екрану;

- список помилок, що спрощує налагодження;

- перевірка під час розгортання програми за допомогою ClickOnce, інсталятора Windows або майстра публікації.

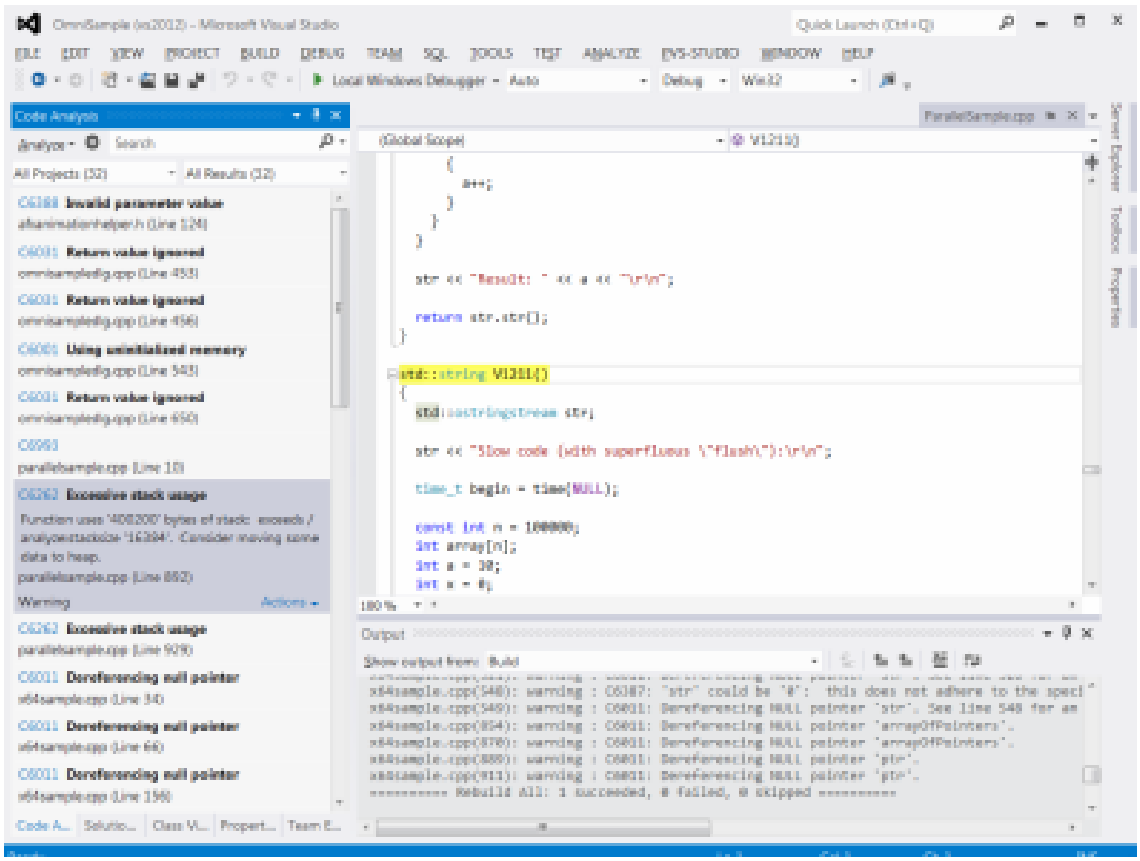


Рис. 2.1. Інтерфейс програмного середовища Microsoft Visual Studio

Недоліки включають те, що, оскільки Visual Studio є складною IDE, вона вимагає значних ресурсів для відкриття та запуску програм. Тому внесення простих змін на деяких пристроях може зайняти багато часу. Для простих завдань рекомендується використовувати редактор Compact або засоби розробки PHP. Початківцю неможливо розібратися в Visual Studio самостійно без проходження спеціалізованих курсів і читання літератури. Цей

продукт більше для досвідчених розробників, які звертають увагу на якість і функції тестування редактора.

Далі розглянемо середовище NetBeans. Безкоштовне середовище розробки з відкритим кодом. Підходить для редагування існуючих проектів або створення нових [3]. NetBeans надає простий інтерфейс перетягування, який постачається з великою кількістю зручних шаблонів проектів. Середовище в основному використовується для розробки програм Java, але ви можете інсталювати пакети, які підтримують інші мови. Підтримувані мови програмування: C, C++, C++ 11, Fortran, HTML 5, Java, PHP та інші. Інтерфейс NetBeans показаний на малюнку. 2.2 [3].

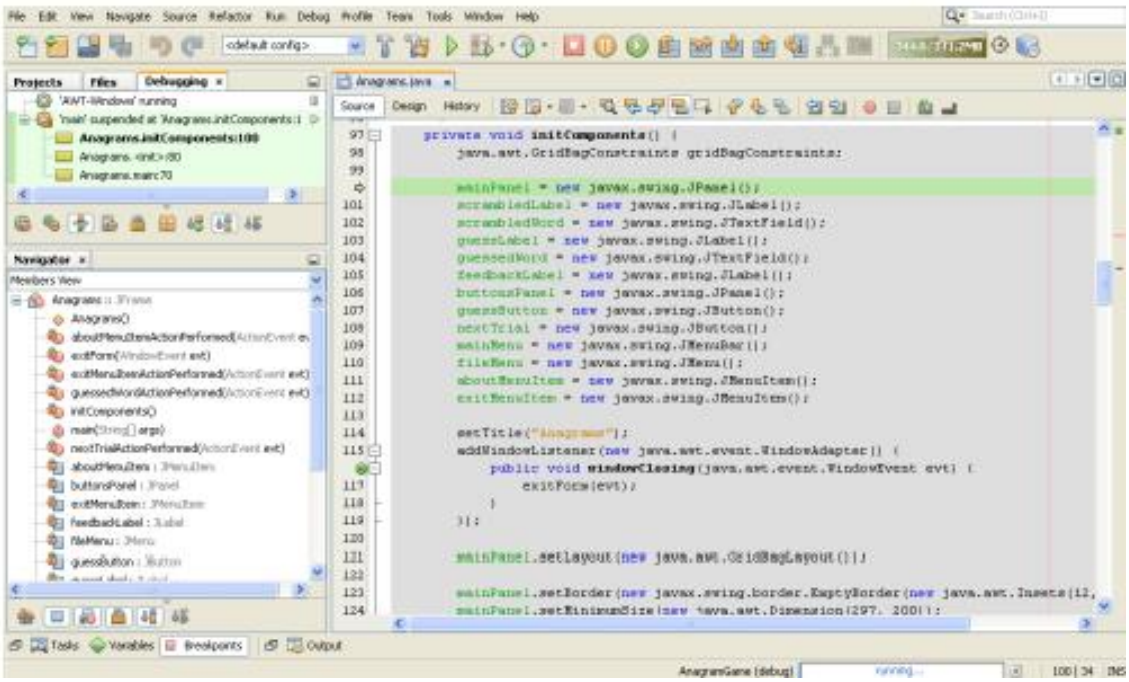


Рис. 2.2. Інтерфейс середовища NetBeans

Особливості середовища NetBeans:

- інтуїтивно зрозумілий інтерфейс перетягування;
- динамічні та статичні бібліотеки;
- інтеграція кількох сеансів GNU Debugger з підтримкою коду;
- можливість віддаленого розгортання;
- сумісність з платформами Windows, Linux, OS X і Solaris;

- підтримка Qt Toolkit;

- підтримка Fortran і Assembler;

- підтримка кількох компіляторів, включаючи Clang/LLVM, Cygwin, GNU, MinGW і Oracle Solaris Studio..

До недоліків можна віднести те, що це безкоштовне середовище розробки споживає багато пам’яті, тому на деяких ПК воно може працювати повільно.

Далі розберемо PyCharm.

PyCharm розроблено командою JetBrains. Користувачі отримують безкоштовний Community Edition, 30-денну безкоштовну пробну версію Professional Edition і річну підписку за $213 - $690 для Professional Edition. Розширена підтримка коду та аналіз роблять PyCharm найкращим IDE для програмістів на Python. Підтримувані мови: AngularJS, CoffeeScript, CSS, Python, HTML, JavaScript, Node.js, Python, TypeScript. Інтерфейс PyCharm показаний на рис. 2.3 [4].

Рис. 2.3. Інтерфейс середовища PyCharm

Особливості середовища PyCharm:

- сумісність з операційними системами Windows, Linux і Mac OS;

- поставляється з Django IDE;

- легко інтегрується з Git, Mercurial і SVN;
- настроюваний інтерфейс з емуляцією VIM;
- відладчики Javascript, Python і Django;
- підтримка Google App Engine.

До недоліків можна віднести те, що користувачі скаржаться на те, що в цьому середовищі розробки Python є деякі помилки, наприклад, час від часу не працює функція автозаповнення, що може викликати деякі незручності.

Таким чином, можна зробити висновок, що середовище Visual Studio найкраще підходить для розробки інтерфейсу автоматизованої системи інтерфейсу користувача, не в останню чергу тому, що воно має зручний та інтуїтивно зрозумілий інтерфейс та підтримку мови програмування C#, а також спрощений робочий процес та ієрархія файлів.

2.4. Висновки до розділу 2

Розбор інтерфейсу програми та мов програмування для наступних робіт з ними

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

3.1. Розробка програми на мові програмування Python

Для демонстрації реалізації проекту було обрано середовище програмування PyCharm від JetBrains і відповідну мову програмування Python.

Для початку поставимо собі завдання розробити програму з прогнозування цін на квартиру спираючись на початкову вибірку, де:

- є робота з масивами та складні обчислення в масиві;
- є робота із файлами;
- Реалізований графічний інтерфейс;
- програма буде самонавчати спираючись на отриманий результат при кожному прогнозуванні;

Вхідні дані:

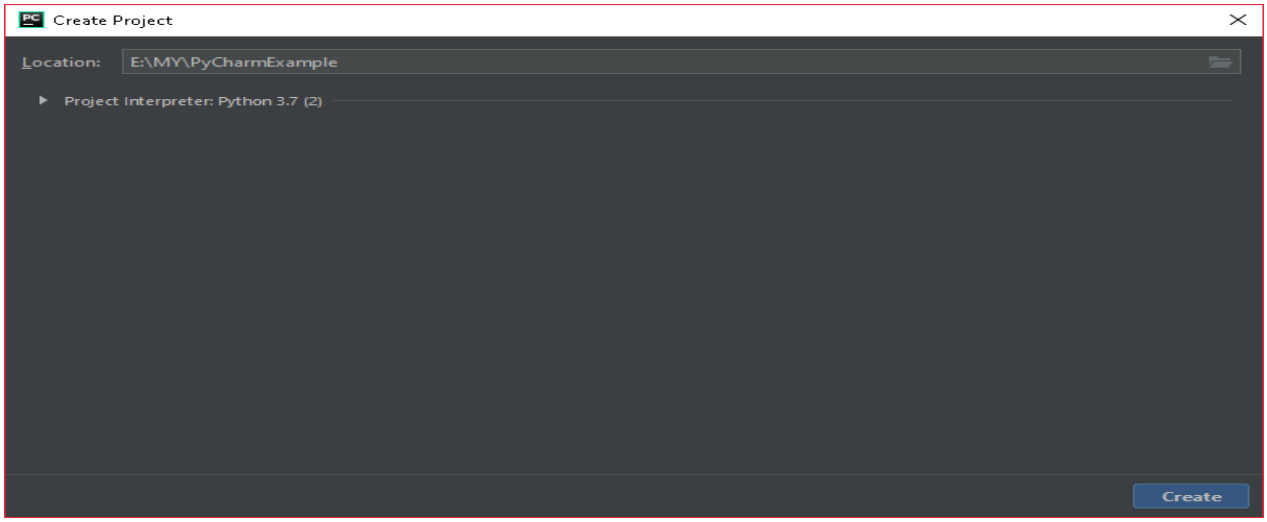
- кількість кімнат;
- Площа, м2;
- Поверх;
- Наявність ліфта;

Вихідні дані:

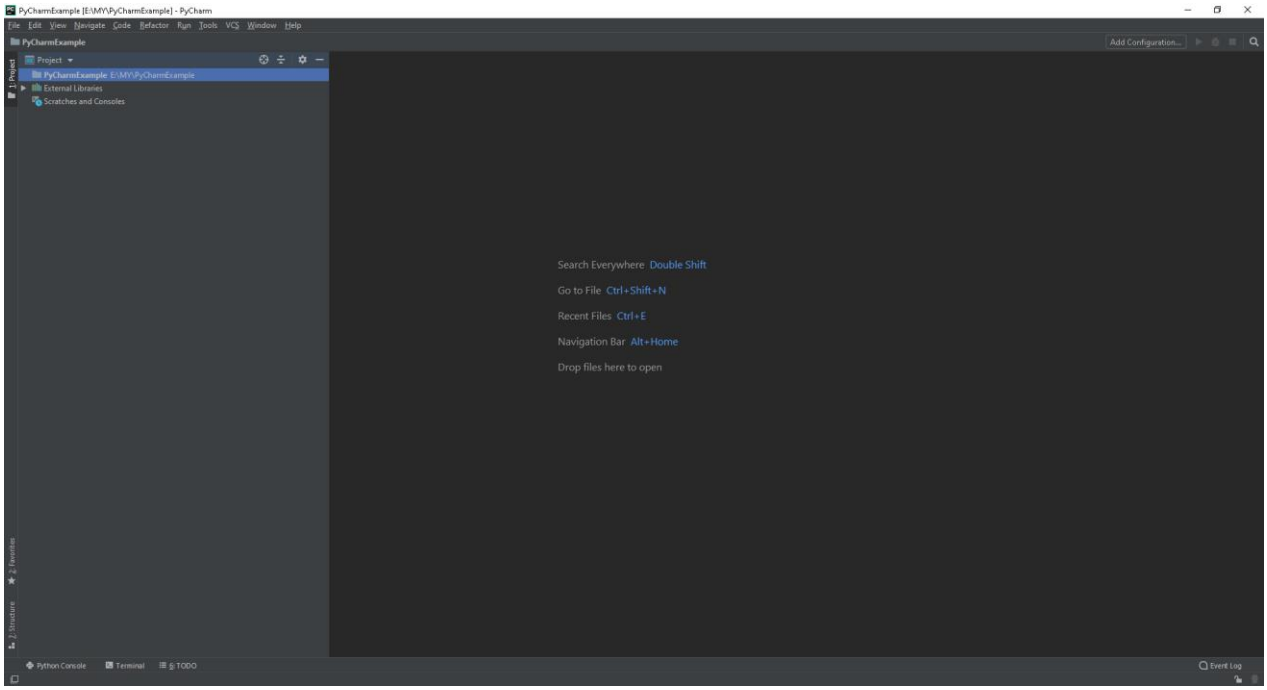
- спрогнозована вартість квартири;

Насамперед потрібно запустити середовище розробки PyCharm та створити новий проект

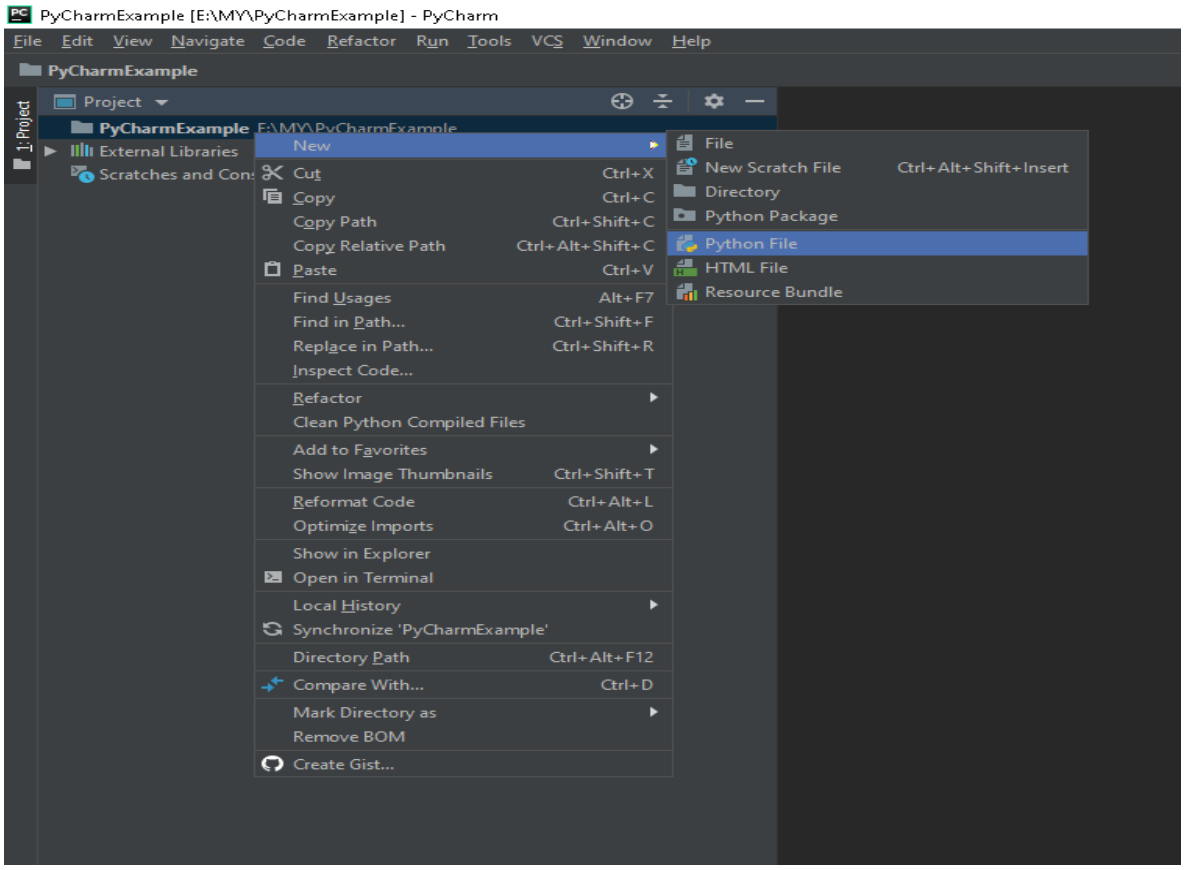
Для цього виберемо File-> New Project.(малюнок №1)-створення проекту



Далі у вікні створення проекту виберемо шлях до його розташування на диску.В результаті було отримано вікно порожнього проекту (мал. №2)



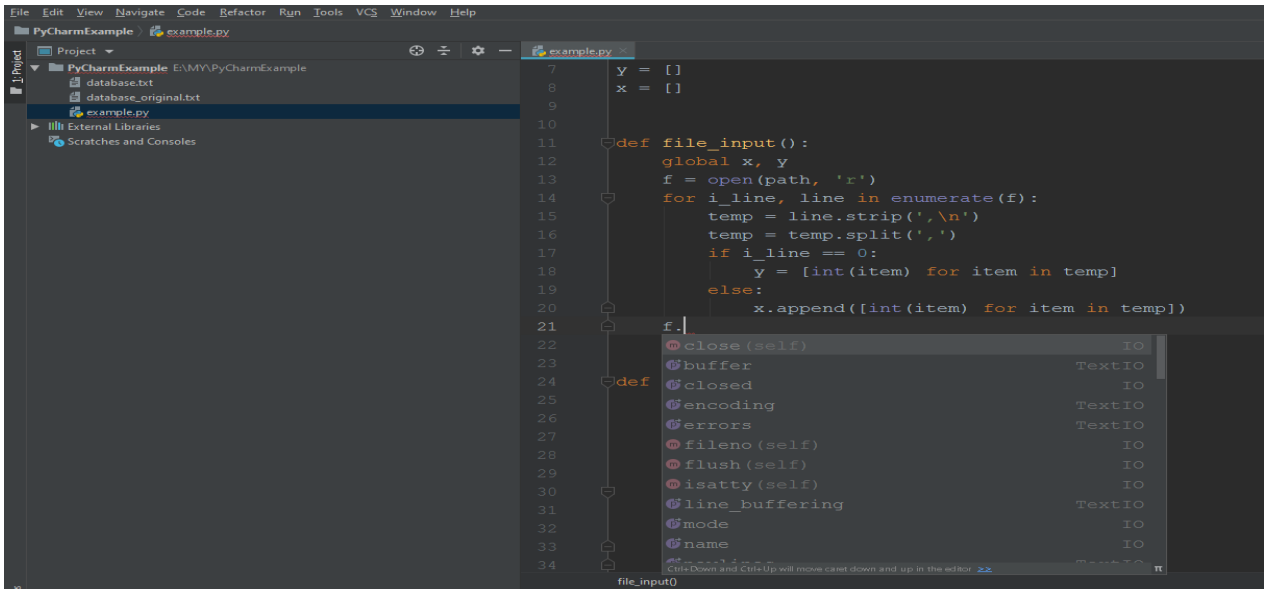
Наступним кроком буде додавання в проект виконуваного файлу, в якому надалі розташовуватиметься наш код (рис. №3).



Дотримуючись подальших підказок, ми вводимо ім'я файлу і отримуємо порожній файл, куди необхідно написати наш код.

Початкова вибірка знаходиться у файлі database.txt.

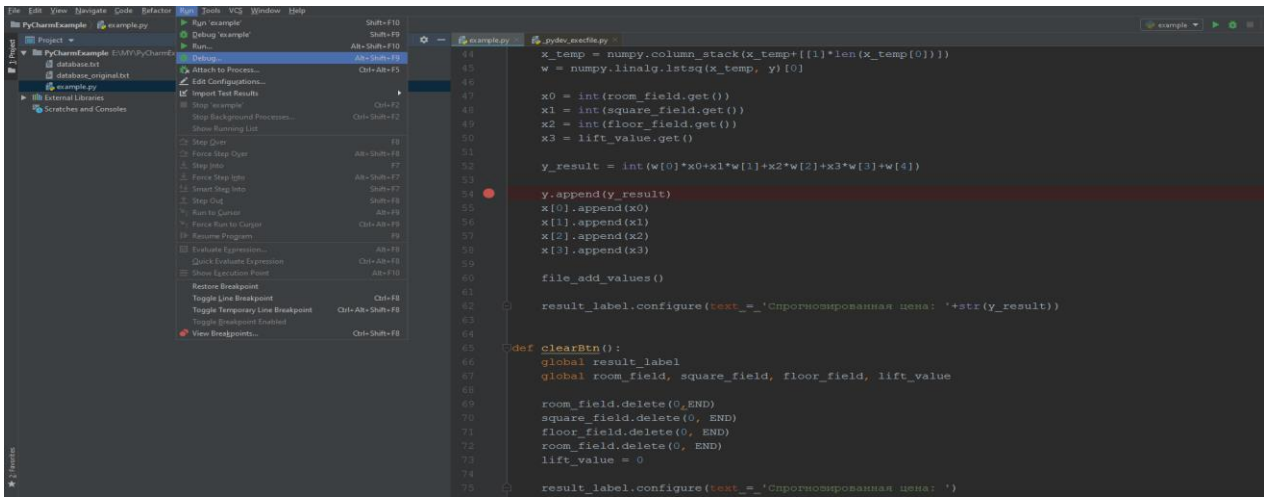
У процесі роботи програма самонавчається і додає у цей файл отримані під час розрахунку результати.У процесі розробки постійно використовувалося автозаповнення (мал.5).



Блок-схема алгоритму нашої програми

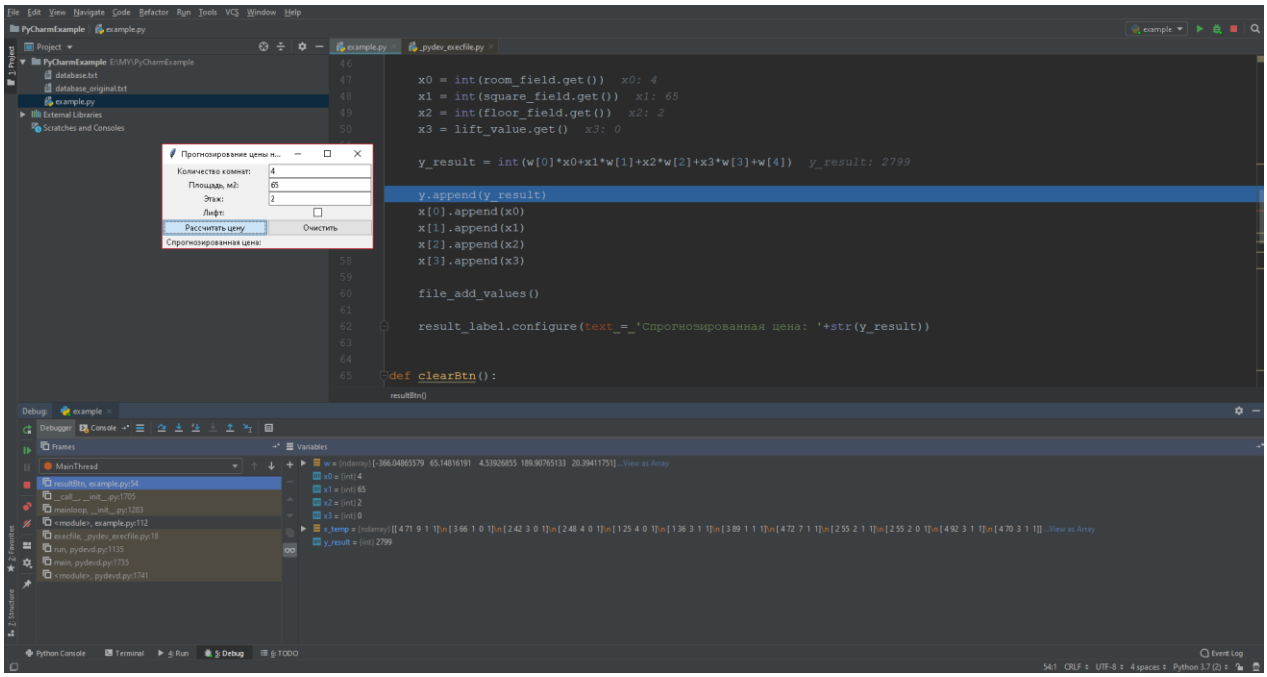
Блок-схема алгоритму програми(мал. №4)

А також, наприклад, вбудовані засоби налагодження.Для цього необхідно біля певного рядка коду встановлювати маркер зупинки програми та запустити програму в режимі Debug (мал.6)



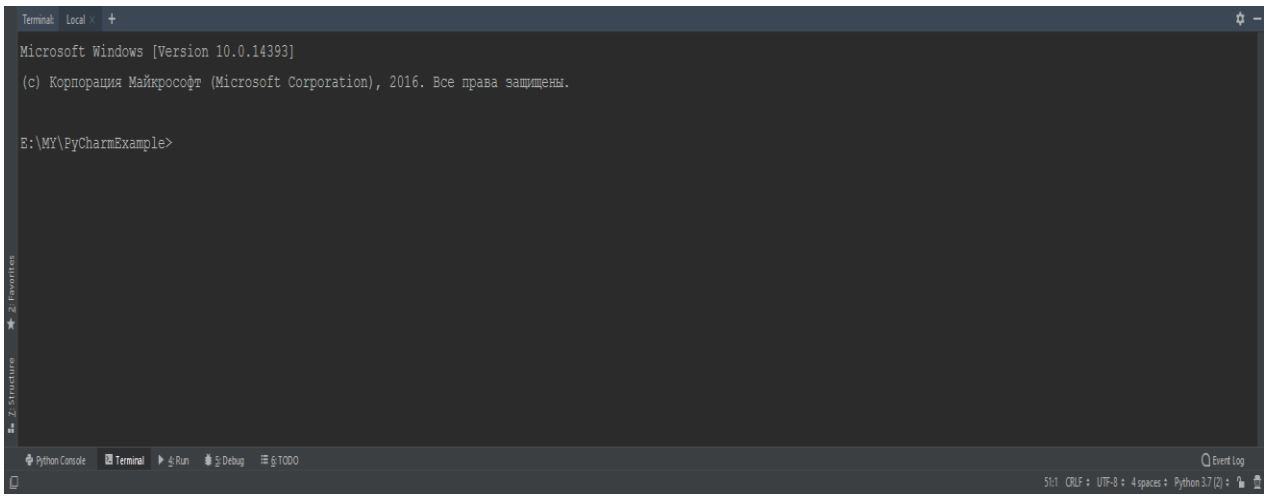
В результаті при виконанні, як тільки програма заходить у місце, де був поставлений маркер, виконання програми зупиняється і ми можемо бачити налагоджувальну інформацію, зміст змінних і т.д.

А також можна кроково виконувати програму далі і дивитися, як в результаті будуть змінюватися змінні і стан нашого ПЗ (мал. №7).

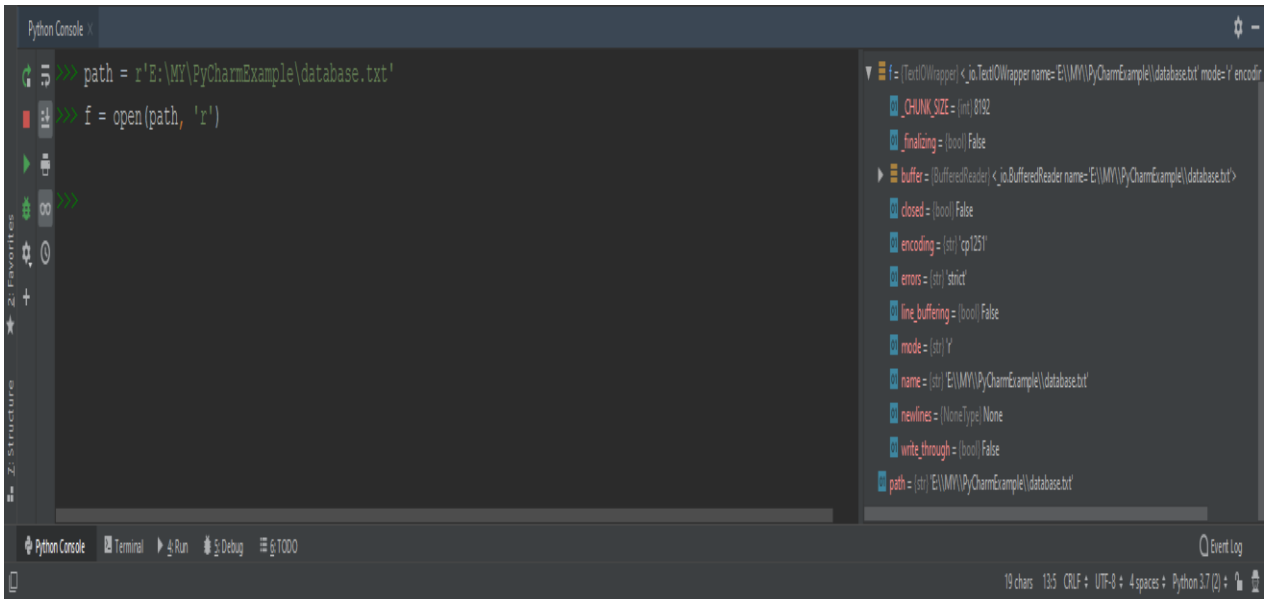


Як можна побачити на мал.7 нижня частина вікна заповнена налагоджувальною інформацією та можливими варіантами дій, що були описані раніше.

Також PyCharm надає можливості, не виходячи з нього, використовувати термінал або консоль операційної системи, що є дуже зручним (рис.8).



Або, наприклад, використовувати консоль інтерпретатора Python.Ця можливість дуже зручна, тому що можна швидко перевірити певні ділянки коду та ідеї, і вже потім без жодних помилок внести це до коду проекту.Приклад такого випадку наведено на мал.9



На мал.9 можна побачити, що при перевірці якихось команд у консолі Python, у правій частині цього вікна ми бачимо, що схоже на налагоджувальну інформацію.

Наприкінці для розробки програми було обрано тему прогнозування ціни на квартиру.Програма має графічний інтерфейс, працює із файлами.Програма має цікаву особливість, вона вміє самонавчати.Програма щоразу запам'ятовує дані потокових розрахунків і щоразу враховує їх разом із вихідною навчальною вибіркою.У процесі опрацьованої роботи наведено приклад використання системи програмування PyCharm та деяких її інструментів, необхідних для комфортної розробки.

ВИСНОВОК

У цій роботі на тему «Системи програмування» було зроблено наступне:

- проаналізовано літературу з обраної теми;
- Розглянуто існуючі мови програмування;
- Розглянуто основні існуючі системи програмування;
- наведено приклад реалізації проекту за допомогою JetBrains PyCharm та мови програмування Python;

Як приклад реалізації проекту було прийнято рішення розробити програму прогнозування ціни на квартиру згідно з навчальною вибіркою.Було розроблено алгоритм, який представлений на блок-схемі і власне сам код програми.

У процесі розробки такого програмного забезпечення було розглянуто використання наступних інструментів та процесів системи програмування JetBrains PyCharm:

- Створення проекту;
- додавання файлів у проект;
- Інструменти автозаповнення;
- режим налагодження;
- Вбудовані можливості терміналу (консолі) операційної системи;
- вбудована консоль інтерпретатора Python;

Крім використаних інструментів було виявлено і багато інших, як, наприклад, інтеграція з системою контролю версій Git або подібними.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1) Марк Саммерфілд.Python на практиці.- Переклад з англійської.- М.: ДМК Прес, 2014. - 338 с.

2) Сузі Р. А. Створення додатків із графічним інтерфейсом користувача // Мова програмування Python: Навчальний посібник.- М.: Інтуїт, Біном.Лабораторія знань, 2006. - 328 с.

3) Інформатика: Базовий курс/Симонович С.В.та інших. – СПб.: Пітер, 2001. – 640 з.

4) Угрінович Н. Інформатика та інформаційні технології.Підручник для 10-11 класів.4-те вид.- М.: Біном.Лабораторія знань, 2007. - 511с.

5) Черпаков І.В.Основи програмування.Підручник та практикум.ISBN: 978-5-9916-9983-9 Москва: Видавництво Юрайт, 2018. - 219 с.Джерела іноземними мовами

6) John E. Grayson.Python і Tkinter Programming.- Manning Publications, 1999. - 658 p.

7) Ivan Van Laningham.Teach Yourself Python in 24 Hours.Sams, 2000Електронні ресурси

8) Короткий огляд мови Python [Електронний ресурс].URL: http://www.helloworld.ru/texts/comp/lang/python/python2/index.htm

9) Порівняння мов програмування щодо роботи на фріланс-біржі [Електронний ресурс].URL: https://www.pvsm.ru/java/118107

10.) Актуальність вивчення сучасних мов програмування [Електронний ресурс].URL: https://scienceproblems.ru/aktualnost-izuchenija-sovremennyh-jazykov.html

11) Системний підхід у технології програмування.Системи програмування. [Електронний ресурс].URL:http://bourabai.kz/alg/system5.htm#5.1.2.2

12) PyCharm.Інструкція з початку роботи.[Електронний ресурс].URL: https://py-charm.blogspot.com/2017/09/blog-post.html

13) Документація з Visual Studio.[Електронний ресурс].URL: https://docs.microsoft.com/ru-ru/visualstudio/ide/?view=vs-2019

14) Середовище програмування Delphi.[Електронний ресурс].URL: http://citforum.ru/programming/32less/les11.shtml#12