Міністерство освіти і науки України
Кам’янець-Подільський національний університет імені Івана Огієнка
Фізико-математичний факультет
Кафедра комп’ютерних наук

Дипломна робота
бакалавра
з теми: “ANDROID ДОДАТОК ДЛЯ ПОЗНАЧЕННЯ ВІДВІДУВАНОСТІ СТУДЕНТІВ”

Виконав: студент 4 курсу, KN1-B18 групи
спеціальності 122 Комп’ютерні науки
Данилко Дмитро Валентинович

Керівник: Пилипюк Тетяна Михайлівна,
доцент кафедри комп’ютерних наук,
кандидат фізико-математичних наук, доцент

Кам’янець-Подільський – 2022

ЗМІСТ

ВСТУП 3
РОЗДІЛ 1. ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ 5
1.1. Загальна характеристика Visual Studio 5
1.2. Особливості мови програмування C++ 11
РОЗДІЛ 2. ПОЛОЖЕННЯ ПРО ВЕДЕННЯ ЖУРНАЛУ ТА РОЗРОБКА ДОДАТКУ 19
2.1. Загальні положення про ведення журналу 19
2.2. Програмна реалізація додатку 20
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 35
ДОДАТКИ 37

ВСТУП

Актуальність дослідження. У сучасній системі освіти постійно відбуваються зміни, що зумовлені процесами економічного, політичного чи соціального характеру. Змінюються, як пріоритети в системі освіти і у системі управління освітою, так і, зміст та структура самого освітнього процесу. Крім цього, у галузі освіти, досить часто, випробовуються й новітні технології. Зокрема, істотні зміни в інформаційній сфері діяльності людини призводять до суттєвого зниження ефективності застарілих підходів до контролю відвідуваності студентами занять. Саме тому, сьогодні є доцільною розробка спеціальної системи, яка дозволить автоматизувати процес обліку відвідування занять студентами та зробити його швидшим, зручнішим та надійнішим. Прогули, запізнення, відсутність на заняттях – це проблема кожного навчального закладу. Такі речі роблять процес навчання непередбачуваним, що надалі позначається на загальних показниках успішності. Автоматизація процесу контролю відвідуваності, може вирішити ці проблеми надавши гнучкі засоби для моніторингу пропусків занять студентами. Така система, дозволить швидко виявляти проблеми з безпричинними пропусками занять студентами та наддасть певні засоби для запобігання прогулам. Крім того, такий засіб, буде дозволяти студенту стежити за власною відвідуваністю, дозволяючи пересвідчитися в тому, що він повною мірою використовує можливості навчання, які доступні для нього.

Зараз є величезна кількість програм які допомогають викладачам. Основна ціль яких полегшити працю викладачів університетів, шкіл, коледжів й інших навчальних закладів. Але всі ці програми пов’язані безпосередньо з тією чи іншою навчальною дисципліною, викладання якої забезпечує той чи інший викладач. Також є програми, які можуть використовуватися для всіх навчальних дисциплін. Основне призначення запропонованого мобільного додатку – зробити легшим і доступнішим позначення відвідуваності студентів,

тобто прототип друкованого варіанту (журналу), який не увесь час доступний як студентам, так і викладачам.

Мета дослідження: створення мобільного додатку для позначення відвідування студентів.

Для реалізації мети, виконувалися такі **завдання**:

- здійснити вибір та обгрунтувати вибір засобів розробки мобільного додатку;
- розробити мобільний додаток для позначення відвідуваності студентів.

Об’єкт досліджнення: розробка та створення мобільного додатку.

Предмет дослідження: створення мобільного додатку для позначення відвідуваності студентів мовою програмування С++.

Структура роботи. Дипломна робота бакалавра складається зі вступу, двох розділів, списку використаних джерел та додатків.

РОЗДІЛ 1. ВИБІР ТА ОБГРУНТУВАННЯ ЗАСОБІВ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ

1.1. Загальна характеристика Visual Studio

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також вебсайти, вебзастосунки, вебслужби як у рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Visual Studio 97 **(кодове ім'я *Boston*)** — перша випущена версія Visual Studio, в ній вперше були зібрані разом різні засоби розробки ПЗ. Вона була випущена в двох версія Professional і Enterprise і включала Visual Basic 5.0, Visual C++ 5.0, Visual J++ 1.1, Visual FoxPro 5.0, вперше з'явилося середовище розробки ASP — Visual InterDev. Це була перша спроба Microsoft створити єдине середовище для розробки для різних мов програмування: Visual C++, Visual J++, Visual InterDev, і MSDN використовували одне середовище, зване Developer Studio. Visual Basic і Visual FoxPro використовували окремі середовища для розробки.

Visual Studio 6.0 **(кодове ім'я *Aspen*)** — випущена в червні 1998 — остання версія Visual Studio, що працює на платформі Win9x. Як і раніше популярна серед програмістів, які використовували Visual Basic. Ця версія була основним середовищем розробки застосунків під Windows від Microsoft, до появи платформи .NET. Ця версія була базовою для розробників Microsoft'у впродовж наступних чотирьох років. Visual Studio 6.0 була останньою версією яка містила в собі COM версію Visual Basic'a. Також це була остання версія яка містила мову програмування Visual J++. Існували дві версії Visual Studio 6.0: Professional та Enterprise. Enterprise містив у собі додаткові плагіни, які не

були присутні в Professional, включно з Application Performance Explorer, Automation Manager, Microsoft Visual Modeler, RemAuto Connection Manager, Visual Studio Analyzer.

Visual Studio .NET 2002 **(кодове ім'я Rainier; внутрішня версія 7.0)** — випущена в лютому 2002 (містила .NET Framework 1.0). Service Pack 1 для Visual Studio .NET (2002) випущений в березні 2005. Бета — версія була доступною в 2001 році. Найбільшою зміною було впровадження менеджера коду. Застосунки, які були розроблені за допомогою Visual Studio .NET, не компілювались в машинну мову, а перетворювались у формат, який мав назву Microsoft Intermediate Language (MSIL) або Common Intermediate Language (CIL). Коли MSIL-застосунок використовували, він автоматично компілювався в машинну мову для даної платформи, це робило код кросплатформним, що давало змогу виконувати його на різних платформах. Проте такі застосунки могли використовуватись тільки на платформах що підтримували Common Language Infrastructure. Це робило можливим використання застосунків в операційних системах Linux або Mac OS використовуючи спеціальні програми як, Mono та DotGNU. Пакет був випущений одразу в чотирьох версіях: Academic, Professional, Enterprise Developer та Enterprise Architect. Було вперше представлено нову мову програмування C# (сі — шарп), яка була спеціально розроблена для використання в Visual Studio .NET. Також було представлено спадкоємця Visual J++ що мав назву Visual J#. За допомогою Visual Studio .NET можна було створювати звичайні застосунки та вебсайти (використовуючи ASP.NET та Web сервіси). У травні 2005 року було випущено пакет оновлень для Visual Studio .NET.

Visual Studio .NET 2003 (кодове ім'я Everett; внутрішня версія 7.1) — випущена в квітні 2003 (містить .NET Framework 1.1). Це була перша версія, що допускала розробку застосунків для мобільних пристроїв, використовуючи ASP.NET або .NET Compact Framework. Внутрішній номер версії Visual Studio .NET 2003 був 7.1, але версії файлів були 8.0. Visual Studio .NET 2003 також

було випущено в чотирьох варіантах: Academic, Professional, Enterprise Developer, та Enterprise Architect. Версія Enterprise Architect містила спеціальний застосунок Microsoft Visio 2002, що використовувався для побудови UML об'єктів. Пакет оновлень для Visual Studio .NET 2003 було випущено 13 вересня 2006 року.

Visual Studio 2005 (кодове ім'я Whidbey; внутрішня версія 8.0) — випущена в кінці жовтня 2005 (включає .NET Framework 2.0). На початку листопада 2005 також вийшла серія продуктів в редакції Express: Visual C++ 2005 Express, Visual Basic 2005 Express, Visual C# 2005 Express і інше[1]. 19 квітня 2006 редакція Express стала безоплатною. Service Pack 1[2] для VS2005 і всіх Express-редакцій[3] випущений 14 грудня 2006 року. Додаткова латка для SP1, що вирішує проблему сумісності з Windows Vista випущена 3 червня 2007.

Visual Studio 2005 підтримує ASP .NET версії 2.0, і дозволяє підтримувати он–лайн сервіси ASP .NET. Також підтримує усі тими SQL Server'ів до 2005 року. Було надано можливості для розробки 64-розрядних застосунків. Можна було компілювати коди ваших застосунків як 32- або 64- розрядні. Visual Studio 2005 містила 64 — бітні версії стандартних бібліотек. Також було випущено ще два продукти, які мали назву, Visual Studio Tools for Applications (VSA) та Visual Basic for Applications (VBA). В них було включено підтримку Microsoft Office 2007. Згодом була додана підтримка таких додатків як, WPF, WCF, WF, LINO та .NET Framework 3.5.

Visual Studio 2008

В листопаді 2007 корпорація Microsoft оголосила[4] про випуск нових продуктів для розробників Visual Studio 2008 (кодове ім'я Orcas) і .NET Framework 3.5.

Visual Studio 2008 сконцентрувала свою увагу на розробці застосунків для Windows Vista, Microsoft Office 2007 та вебзастосунків. Для візуальної розробки було презентовано Windows Presentation Foundation та новий HTML/CSS редактор. Маючи понад 250 нових функціональних можливостей,

Visual Studio 2008 пропонує істотні поліпшення кожної з версій, включаючи Visual Studio Express та Visual Studio Team System.

- *Language Integrated Query (LINQ)* заповнює прогалину між об'єктним програмуванням і даними та дає змогу розробникам зосередитися не на доступі до даних, а на роботі з ними.
- *Visual Studio Team System* підтримує управління збіркою програм, включно з виконанням планових збірок та збірок в результаті процесу безперервної інтеграції. Team Build забезпечує інтегровану підтримку статичного аналізу коду під час виконання збірки і проведення контрольних випробувань збірки.
- Значне спрощення розробки для Web завдяки новій техніці обміну інформацією з вебсервером для вебсайтів, які підтримують AJAX/JSON. Нові елементи управління ASP.NET передбачають покращене управління сторінками і шаблони, а Windows Communication Foundation передбачає вбудовану підтримку RSS і REST.
- *.NET Framework 3.5* також містить декілька нових функціональних можливостей, серед яких можна назвати можливості для Web 2.0, сервіс-орієнтовану архітектуру (Service-Oriented Architecture, SOA) та програми на базі технології ПЗ + Сервіси (Software+Services). Сервіси з підтримкою послідовності операцій надають нові класи моделі програмування, які спрощують створення сервісів з підтримкою послідовності операцій за рахунок використання Windows Communication Foundation і Windows Workflow Foundation. Це дозволяє розробникам на .NET Framework створювати бізнес-логіку сервісу, використовуючи WF, та організовувати обмін повідомленнями з цим сервісом за допомогою WCF.
- Підтримка додаткових протоколів вебсервісів у Windows Communication Foundation, включаючи протоколи Web Services Atomic Transaction (WS-AtomicTransaction) 1.1, WS-ReliableMessaging 1.1, WS-Secure Conversation та Web Services Coordination (WS-Coordination) 1.1.

- Text Template Transformation Toolkit шаблонно орієнтований генератор коду включений як частина середовища.

Visual Studio 2010

Представлений 12 квітня 2010 року. Включає .NET Framework 4.0. З'явилася нова мова F#, Visual C++ підтримує стандарт C++0x.

Інструменти Visual Studio 2010 допоможуть не тільки в створенні звичних програм для мобільних телефонів і персональних комп'ютерів, але в розробці хмарних застосунків. При цьому процес тестування, зневадження і розгортання програм в «хмарі» аналогічний створенню .NET-застосунків. Іншим важливим доповненням в Visual Studio 2010 є інструменти для багатониткової розробки з використанням як некерованого коду, так і .NET Framework.

У Visual Studio 2010 повністю перероблений інтерфейс з використанням Windows Presentation Foundation (WPF), упроваджено наступне покоління інструментів ASP.NET, є підтримка динамічних розширень в мовах програмування C# і Visual Basic, використовуються нові шаблони проектів, інструментарій для документування тестових сценаріїв і велика кількість нових бібліотек, що підтримують Windows 7.

Visual Studio Ultimate 2010, формально це Visual Studio Team System 2010, кодове ім'я Rosario, це новий інструмент для спільної розробки застосунків.

Visual Studio 2012

Представлена 2 серпня 2012 року. Включає .NET Framework 4.5. Головні нововведення це підтримка Windows Runtime, C++/CX(Component Extensions), бібліотека C++ AMP для GPGPU програмування, компілятор Visual C++ майже підтримує стандарт C++11. З'явився новий тип проектів, котрі дозволяють писати рідні застосунки (у стилі Windows Metro) для операційної системи Windows 8. Інструмент швидкої розробки та розгортання настільних біснес-застосунків LightSwitch був також включений в пакет Visual Studio. Значно оновився

зовнішній інтерфейс програми, були поліпшені додатки оглядів проектів (англ. *solution explorer*) та тестів.

Загалом вийшло 5 оновлень цієї версії, останнє датується 24 серпня 2015 року.

Visual Studio 2013

Побачивши світ у вигляді попередньої версії 26 червня 2013 року, нова версія з типовим кодовим ім'ям *Dev12* остаточно вийшла у продаж 17 жовтня 2013 року.

Помітними змінами були покращення інтерфейсу користувача для командної розробки, підтримка платформи десктопної та мобільної розробки Windows 8.1, а також покращення для Web-розробників. Були перероблені та вдосконалені засоби діагностики та відлагожування, представлені нові засоби аналізу витоків пам'яті тощо.

Усього вийшло 5 оновлень даної версії, останнє відбулося 20 липня 2015 року.

Visual Studio 2015

Наступна версія Visual Studio, під кодовим ім'ям *Dev14* була представлена 20 червня 2015 року. Суттєвою зміною стала підтримка багатьох цільових платформ: окрім базової Windows з'явилась можливість будувати проекти для IOS та Android. Для розробників комп'ютерних ігор була додана підтримка програмного каркасу Unity. Був оновлений механізм автентифікації: користувач під час запуску Visual Studio синхронізується з єдиним аккаунтом Microsoft.

Версія має в собі .NET Framework 4.6 та підтримку універсальної платформи Windows 10. Розробників на мові C++ потішили новим функціоналом стандарту C++14, та навіть деякими поліпшеннями з C++17.

Останнім оновленням на цей момент є Update 2 від 30 березня 2016 року, у якому багато уваги приділено стабільності, та продовжено роботу у напрямку підтримки нових стандартів мови C++.

Visual Studio 2016

Перший нестабільний випуск наступної версії програми під умовною назвою «15» побачив світ 30 березня 2016 року. Основними змінами стали інтерфейс інсталятора та численні незначні покращення у різних компонентах середовища розробки. Очікується підтримка мови програмування Solidity.

Версія остаточно побачила світ під назвою Visual Studio 2017 - 7 березня 2017 року.

Visual Studio 2019

Перша версія нової Visual Studio (Preview) була випущена у грудні 2018 року. Вона мала 5 етапів. У березні був вже визначений RC (Release Candidate). Тому через місяць, а саме 2 квітня була випущена стабільна версія Visual Studio 2019 на Windows та Visual Studio on MAC. Як і у минулих версіях є три типи (Community, Professional, Enterprise).

1.2 Особливості мови програмування C++

C++ (Сі-плюс-плюс) — мова програмування загального призначення з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої, процедурної та ін. Б'ярн Страуструп (англ. *Bjarne Stroustrup*) почав створювати C++ в AT&T Bell Laboratories (Мюррей-Хілл[en], Нью-Джерсі) у 1979 року. На етапі зародження мова мала назву «Сі з класами». Згодом Страуструп перейменував мову на *C++* у 1984р. Бере витоки з мови програмування C. Вперше описана міжнародним стандартом ISO/IEC 14882:1998 (C++98), найбільш актуальним же є стандарт ISO/IEC 14882:2020 (C++20).

У 1990-х роках C++ стала однією з найуживаніших мов програмування загального призначення. Мову використовують для системного програмування, розробки прикладного програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм, наприклад, відеоігор. C++ суттєво вплинула на інші популярні сьогодні мови програмування: C# та Java.

В 1998 році мова C++ була стандартизована Міжнародною організацією стандартизації під номером 14882:1998 — Мова Програмування C++. Поточний стандарт — C++11, він був прийнятий у 2011 році робочою групою МОС після десятирічної підготовки.

Стандарт C++ на 1998 рік складається з двох основних частин: ядра мови і стандартної бібліотеки. Стандартна бібліотека C++ увібрала в себе бібліотеку шаблонів STL, що розроблялася одночасно із стандартом. Зараз назва STL офіційно не вживається, проте в колах програмістів на C++ ця назва використовується для позначення частини стандартної бібліотеки, що містить визначення шаблонів контейнерів, ітераторів, алгоритмів і функторів.

Стандарт C++ містить нормативне посилання на стандарт Сі від 1990 року і не визначає самостійно ті функції стандартної бібліотеки, які запозичуються із стандартної бібліотеки C.

Поза тим, існує величезна кількість бібліотек C++, котрі не входять в стандарт. У програмах на C++ можна використовувати багато бібліотек C.

Стандартизація визначила мову програмування C++, проте за цією назвою можуть ховатися також неповні, обмежені достандартні варіанти мови. Спочатку мова розвивалася поза формальними рамками, спонтанно, у міру завдань, що ставилися перед ним. Розвиток мови супроводив розвиток кросс-компілятора Cfront. Нововведення в мові відбивалися в зміні номера версії кросс-компілятора. Ці номери версій кросс-компілятора розповсюджувалися і на саму мову, але стосовно теперішнього часу мову про версії мови C++ не ведуть.

Нові можливості в порівнянні з C

Мова Сі++ багато в чому є надмножиною C. Нові можливості C++ включають оголошення у вигляді виразів, перетворення типів у вигляді функцій, оператори **new** і **delete**, тип **bool**, посилання, розширене поняття константності та змінності, функції, що підставляються, аргументи за умовчанням, перевизначення, простори імен, класи (включаючи і всі пов'язані з класами можливості, такі як успадкування, функції-члени (методи),

віртуальні функції, абстрактні класи і конструктори), перевизначення операторів, шаблони, оператор **::**, обробку винятків, динамічну ідентифікацію і багато що інше. Сі++ є також мовою строгого типування і накладає більше вимагань щодо дотримання типів, порівняно з Сі.

У С++ з'явилися коментарі у вигляді подвійної косої риски («//»), які були в попереднику С — мові BCPL.

Деякі особливості С++ пізніше були перенесені в С, наприклад ключові слова ***const*** і ***inline***, оголошення в циклах ***for*** і коментарі в стилі С++ («//»). У пізніших реалізаціях С також були представлені можливості, яких немає в С++, наприклад макроси ***vararg*** і покращена робота з масивами-параметрами.

В цьому розділі описуються можливості, безпосередньо не пов'язані з об'єктно-орієнтованим програмуванням (ООП). Багато які з них, проте, особливо важливі у поєднанні з ООП.

- Ключове слово ***inline*** означає, що функція є хорошим кандидатом на оптимізацію, при якій в місцях звернення до функції компілятор вставить тіло цієї функції, а не код виклику. Приклад: *inline double Sqr(double x) {return x*x;}*
- Замість функцій ***malloc*** і ***free***, введені нові оператори **new** і **delete**. Якщо **T** — довільний тип.
- **new T** виділяє пам'ять, достатню для розміщення одного об'єкта типу *T*; після завершення виклику оператора, компілятор здійснює ініціалізацію об'єкта (викликаючи його конструктор, якщо такий був визначний) і повертає вказівник типу *T**.
- **new T[n]** виділяє пам'ять, достатню для розміщення *n* об'єктів типу *T*; після завершення виклику оператора, компілятор здійснює ініціалізацію кожного з *n* об'єктів і повертає вказівник типу *T**.
- **delete p** — звільняє пам'ять, на яку посилається вказівник *p*, виділену для нього раніше операцією *new T*. Деінціалізація об'єкта (викликаючи деструктора) забезпечується компілятором ще до виклику оператора *delete*.

- **delete [] p** — звільняє область пам'яті, виділену для цього масиву раніше операцією *new T[n]*. Деінціалізація кожного елементу масиву забезпечується компілятором ще до виклику оператора.

Як видно, однією з принципових відмінностей операторів *new* та *delete* від своїх попередників, *malloc* і *free*, є обов'язковість ініціалізації об'єктів, пам'ять під які було призначено. Іншою відмінністю є те, що загальна реалізація (тобто визначена за умовчанням) оператора *new* не повертає нульове значення вказівника у випадку помилки призначення пам'яті (наприклад з причини її браку). Натомість, *new* кидає виняток (наприклад, *std::bad_alloc* в ситуації браку пам'яті). Так само як і для *free*, якщо значенням аргументу оператора *delete* є 0, ані звільнення пам'яті, ані деінціалізація не відбувається (при цьому, подібна ситуація не вважається помилковою).

- Функції можуть приймати аргументи за посиланням. Наприклад, функція void f(int& x) {x=3;} присвоює своєму аргументу значення 3. Функції також можуть повертати результат за посиланням, і посилання можуть бути поза всяким зв'язком з функціями. Наприклад, *{double& b=a[3]; b=sin(b);}* еквівалентно *a[3]=sin(a[3]);*. Посилання певною мірою схожі з вказівниками, з такими особливостями: при описі посилання ініціалізувалися вказівником на існуюче значення даного типу; посилання довічно указує на одну і ту ж адресу; при зверненні до посилання операція читання пам'яті за адресою посилання проводиться автоматично. На відміну від вказівників, посилання не може бути константним саме по собі, однак може посилатися на константний об'єкт. Наприклад, *int const & const ref = a[3];* — є некоректним, з точки зору C++, виразом, на відміну від *int const * const ref = &a[3];* в свою чергу, і *int const & ref = a[3];*, і *int const * ref = &a[3];* — є цілком прийнятними.
- Можуть бути декілька функцій з одним і тим же ім'ям, але різними типами або кількістю аргументів (перевантаження функцій; при цьому тип

значення, що повертається, на перевантаження не впливає). Наприклад, цілком можна писати:

```
void Print(int x);
void Print(double x);
void Print(int x, int y);
```

- Один або декілька останніх аргументів функції можуть задаватися за умовчанням. Наприклад, якщо функція описана як *void f(int x, int y=5, int z=10)*, виклики *f(1)*, *f(1,5)* і *f(1,5,10)* еквівалентні.
- При описі функцій відсутність аргументів в дужках означає, на відміну від Сі, що аргументів немає, а не те, що вони невідомі. Якщо аргументи невідомі, треба користуватися багатокрапкою, наприклад *int printf(const char* fmt ...)*. Тип першого аргументу повинен бути заданий.
- Можна описувати оператори над новими типами. Наприклад, так:

```
struct Date {int day, month, year;};
void operator ++(struct Date& date);
```

Оператори нічим не відрізняються від (інших) функцій. Не можна описувати оператори над зумовленими типами (скажімо, перевизначати множення чисел); не можна вигадувати нові операції, яких немає в C++ (скажімо **); арність (кількість параметрів) і пріоритет операцій зберігається (скажімо, у виразі a+b*c спочатку виконуватиметься множення, а потім складання, до яких би типів не належали *a*, *b* і *c*.) Можна перевизначити оператор [] (з одним параметром) і () (з будь-яким числом параметрів).

- Додані простори імен **namespace**. Наприклад, якщо написати

```
namespace Foo {
  const int x=5;
  typedef int** T;
  void f(y) {return y*x};
  double g(T);
  ...
```

}

то поза фігурними дужками ми повинні звертатися до *T, x, f, g* як *Foo::T, Foo::x, Foo::f, Foo::g*. Якщо ми в якійсь одиниці трансляції (файл основного коду, наприклад myFile.cpp, та всі заголовкові файли що він включає) хочемо звертатися до них безпосередньо, ми можемо написати

using namespace Foo;

Або ж

using Foo::T;

Також можна створити синонім на вже існуючий простір імен (наприклад, аби уникнути постійно повторювати довгу назву простору)

```
namespace MyVeryOwnNameSpace {
  typedef std::vector< std::string > StringTable;
}
namespace My = MyVeryOwnNameSpace;
```

Простори імен потрібні, щоб не виникало колізій між пакетами, що мають однакові імена глобальних змінних, функцій і типів. Спеціальним випадком є безіменний простір імен

```
namespace {
  ...
}
```

Всі імена, описані в ньому, доступні в поточній одиниці трансляції і більше ніде, неначебто ми до кожного опису приписали *static*.

- Доданий новий тип **bool**, що має значення **true** і **false**. Операції порівняння повертають тип **bool**. Вирази в дужках після *if, while* приводяться до типу **bool**.
- // означає, що вся частина рядка, що залишилася, є коментарем.
- Додані **шаблони** (**template**). Наприклад, *template<class T> T Min(T x, T y) {return x<y?x:y;}* визначає функцію *Min* для будь-яких типів. Шаблони можуть задавати не тільки функції, але і типи. Наприклад, *template<class*

T> struct Array{int len; T val;};* визначає масив значень будь-якого типу, після чого ми можемо писати *Array<float> x;*

- Введена стандартна бібліотека шаблонів (STL, англ. *Standard Template Library*), що визначає шаблони і функції для векторів (одновимірних масивів довільної довжини), множин, асоціативних масивів (map), списків, символьних рядків, потоків введення-виводу та інші шаблони і функції.
- Якщо описана структура, клас (про класи див. нижче), об'єднання (union) або перерахування (enum), її ім'я є ім'ям типу, наприклад:

struct Time{int hh,mm,ss;};

Time t1, t2;

- Усередині структури або класу можна описувати нові типи, як через *typedef*, так і через опис інших структур або класів. Для доступу до таких типів поза структурою або класом, до імені типу додається ім'я структури і дві двокрапки:

struct S {typedef int** T; T x;}; S::T y;

Об'єктно-орієнтовані особливості мови

C++ додає до C об'єктно-орієнтовані можливості. Він вводить класи, які забезпечують три найважливіші властивості ООП: інкапсуляцію, успадкування і поліморфізм.

Проблеми старого підходу

В мові C основним способом організації даних були структури. Структура складається з набору полів, які ніяк не захищені. Якщо елементи структури мають змінну довжину, їх представляють у вигляді вказівників. Виділення і звільнення пам'яті під ці вказівники робляться вручну. Так, наприклад, одновимірний масив змінної довжини в мові C з перевіркою меж може представлятися таким чином:

```
struct Array {
   double* val;
   int len;
};
```

void FreeArray (**const struct Array***);

void AllocArray (**const struct Array***, int len);

double Elem (**const struct Array***, int i);

void ChangeElem (**const struct Array***, int i, double x);

Така реалізація небезпечна і неефективна з багатьох причин:

- Необхідно викликати FreeArray і AllocArray. Програміст може забути викликати одну з цих функцій, або викликати її дуже рано/запізно, або двічі, або з вказівником на неправильний масив. Все це приводить до помилок, що важко виявити.
- Функції Elem і ChangeElem повільні.
- Немає ніякого способу перешкодити програмістам створювати і інші функції для роботи із структурою Array. Ці функції можуть робити з полями len і val будь-що.
- Немає ніякого способу перешкодити програмістам безпосередньо міняти поля len і val.
- Присвоєння об'єктів типу struct Array приведе до того, що їхні поля val указуватимуть на одну і ту ж область пам'яті. Немає ніякого способу ні заборонити присвоєння, ні змінити таку поведінку.

Мова С++, використовуючи ООП, усуває всі ці проблеми.

Отже, вибір мови програмування С++ для створення мобільного додатку позначення відвідуваності студентів є обгрунтованим.

РОЗДІЛ 2. ПОЛОЖЕННЯ ПРО ВЕДЕННЯ ЖУРНАЛУ ТА РОЗРОБКА ДОДАТКУ

2.1. Загальні положення про ведення журналу

Журнал навчальної групи є одним із основних документів в освітньому процесі. В ньому враховуються проведені з навчальною групою заняття, навчальне навантаження викладачів, успішність та відвідування занять студентами, відпрацювання ними академічних заборгованостей і пропусків занять та незадовільних оцінок. Викладачі повинні виконувати наступні вимоги при веденні журналу:

Всі записи в журналі робляться викладачами чорною або синьою пастою, чітко та акуратно, без виправлень.

У вертикальному (ліва сторона журналу) і в горизонтальному (права сторона журналу) рядку повинна бути проставлена дата проведення заняття та його тривалість у навчальних годинах – (2, 4, 6);

У вертикальному рядку виставляються оцінки та відмітки про відсутність студентів на заняттях. Перевірка наявності студентів здійснюється на початку кожного заняття. Про причини відсутності на заняттях викладачу доповідає староста групи. Викладач особисто робить відмітки в журналі, використовуючи такі скорочення (рис. 2.1):

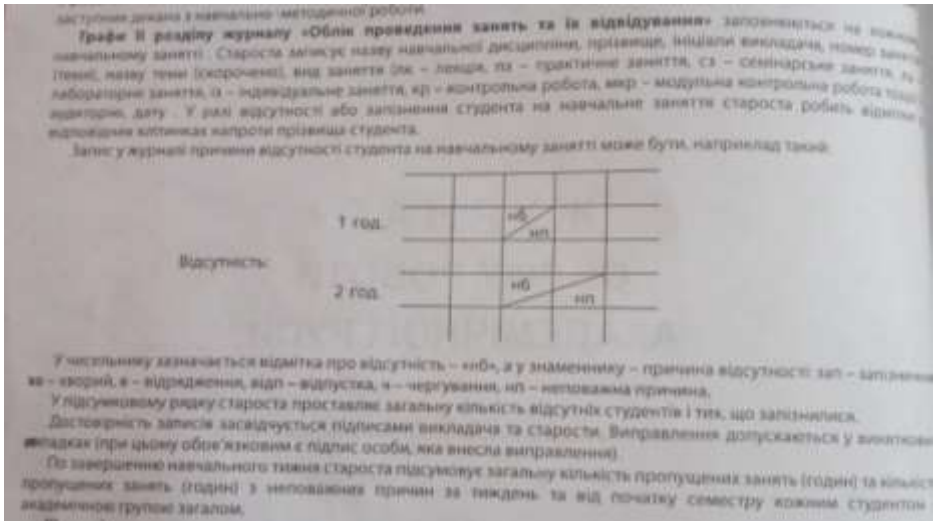


Рис. 2.1. Облік проведення занять та їх відвідування (з журналу обліку роботи академічної групи)

Викладач несе відповідальність тільки за кількість відсутніх, відмічених у журналі, а не за достовірність вказаних причин відсутності.

В горизонтальному рядку у відповідних стрічках повинні бути вказані види заняття (лекція, семінар, тощо), номер та скорочена назва теми, прізвище викладача, який проводив заняття. Відпрацювання заборгованостей, пропущених занять та незадовільних оцінок «0» відмічається виставленням оцінки через дріб. Допускається відмітка про відпрацювання пропущеної лекції та інших не контрольних видів занять округленням відмітки про відсутність студента.

В журналі також відмічається проведення модульних, підсумкових контролів (екзаменів та заліків) з виставленням оцінок за національною, стобальною та ECTS шкалами. Викладач зобов'язаний вести паралельно свій робочий журнал та постійно контролювати, щоб всі записи в журналі робилися тільки ним. Про всі випадки появи в журналі сторонніх записів (оцінок, відміток про відпрацювання, та ін.) терміново доповісти через завідувача кафедри.

Методисти, фахівці факультетів організовують зберігання, видачу журналів на заняття та відпрацювання заборгованостей. Журнали кожного дня повинні здаватися на

2.2. Програмна реалізація додатку

Додаток являє собою програму написану мовою c++ .У додатку є адмін який вже зареєстрований спочатку входить адмін і далі через його кабінет можна зробити різні операції , наприклад зареєструвати студента який присутній на занятті. Надалі студент зможе заходити у свій кабінет. Променю студента і адміна напишу докладніше далі. Зареєстрований студент буде означати що він присутній на занятті. Після заняття переносимо всі данні у

друкований журнал і через меню адміна видаляємо студентів , щоб програма була готова до наступної пари.

Перш за все, ми включимо весь файл заголовка

#include

#include #include

#include

Використання простору імен std:

Використання оператора простору імен просто означає, що всі речі в просторі імен std доступні в області, в якій він присутній, без необхідності префіксувати їх з std:.

Далі створюємо всі ці функції, які згадані нижче:

int adminView(); - меню адміна

int studentView(); - меню студента

int studentLogin(); - вхід студента

int checkCredentials(рядок userName, пароль рядка); - перевірка данних

int getAllStudentsbyRollNo(); - номери студентів присутніх у журналі

int deleteAllStudents(); - видалити всіх студентів

int deleteStudentbyRollno(); - видалити студента за номером

int checkListOfStudentsRegistered(); - перевірка кількості зареєстрованих студентів

int checkPresenseCountbyRollno(); - пошук студента за номером у журналі

int getListOfStudentsWithTheirPresenseCount(); - отримаьт список студентів із кількістю їх присутності

int registerStudent(); - реєстрація студента

int adminLogin(); - вхід адміна

int markMyAttendance(ім'я користувача рядка); - кількість присутніх студентів але для кабінету студента

int countMyAttendance(ім'я користувача рядка); - чи присутній зареєстрований студент

затримка int();

Потім, ми запишемо функцію main()

Це дуже важлива функція кожної програми на C++. І в цьому проекті ця основна функція показує головне меню, яке включає такі функції:

1. Student login
2. Admin login
3. Exit

```
int main(int argc, char** argv) {
while(1)
{
   system("cls");
   cout<<"\n Attendance Management System \n";
   cout<<"-------------------------------------\n\n";
   cout<<"1. Student Login\n";
   cout<<"2. Admin Login\n";
   cout<<"0. Exit\n";
   int choice;
   cout<<"\n Enter you choice: ";
   cin>>choice;
   switch(choice)
   {
   case 1: studentLogin(); break;
   case 2: adminLogin(); break;
   case 0:
         while(1)
```

```
            {
            system("cls");
            cout<<"\n Are you sure, you want to exit? y | n \n";
            char ex;
            cin>>ex;
            if(ex == 'y' || ex == 'Y')
              exit(0);
            else if(ex == 'n' || ex == 'N')
             {
                 break;
             }
             else{
                cout<<"\n Invalid choice !!!";
                getchar();
             }
            }
            break;
    default: cout<<"\n Invalid choice. Enter again ";
    getchar();
    }
}
return 0;
}
```

Наступним кроком пишемо код функції adminLogin()

Тут ми вводимо ім'я користувача та пароль адміністратора.

Call adminView() ,delay() functions.

```
int adminLogin()
{
system("cls");
cout<<"\n --------- Admin Login --------";
```

```
string username;
string password;
cout<<"\n Enter username : ";
cin>>username;
cout<<"\n Enter password : ";
cin>>password;
if(username=="ADMIN" && password=="ADMIN@123")
{
   adminView();
   getchar();
   delay();
}
else
{
cout<<"\n Error ! Invalid Credintials..";
cout<<"\n Press any key for main menu ";
getchar();getchar();
}
return 0;
}
```

Admin Username: "ADMIN" & Admin Password: "ADMIN@123"

Далі функція adminView(). Після введення імені користувача і пароля в adminView() ми розглянемо цей тип меню, які включають в себе деякі функції, такі як зареєструвати студента, видалити всі імена студента зареєстрованого, видалити студента і так далі.

```
int adminView()
{
int goBack = 0;
while(1)
```

```
{
system("cls");
cout<<"\n 1 Register a Student";
cout<<"\n 2 Видалити всі імена студентів, зареєстровані";
cout<<"\n 3 Видалити студента за номером ";
cout<<"\n 4 Контрольний список студента, зареєстрованого userame";
cout<<"\n 5 Перевірити кількість присутніх будь-якого студента roll no";
cout<<"\n 6 Отримати список студентів з кількістю відвідуваності";
cout<<"\n 0. Повернутися <- \n";
вибір int;
cout<<"\n Введіть свій вибір: ";
cin>>choice;
switch(choice)
{
   case 1: registerStudent();break;
   case 2: deleteAllStudents(); break;
   case 3: deleteStudentbyRollno(); break;
   case 4: checkListOfStudentsRegistered(); break;
   case 5: checkPresenseCountbyRollno(); break;
   case 6: getListOfStudentsWithTheirPresenseCount(); break;
   case 0: goBack = 1;break;
   default: cout<<"\n Invalid choice. Enter again ";
   getchar();
}
if(goBack == 1)
{
break;
}
}
```

```
return 0;
}
}
```

Код функції registerStudent()

Ця функція приймає всі деталі студента та створює ім’я користувача та пароль студента, а також перевіряє, чи студент вже існує чи ні.

```
int registerStudent()
{
   cout<<"\n ----- Form to Register Student ---- \n";
   string name, username, password, rollno, address, father, mother;
   cout<<"\n Enter Name : ";     cin>>name;
   cout<<"\n Enter Username : ";     cin>>username;
   cout<<"\n Enter password : ";     cin>>password;
   cout<<"\n Enter rollno : ";     cin>>rollno;
   getchar();
   char add[100];
   cout<<"\n Enter address : ";     cin.getline(add, 100);
   cout<<"\n Enter father : ";     cin>>father;
   cout<<"\n Enter mother : ";     cin>>mother;
   ifstream read;
   read.open("db.dat");
   if(read)
   {   int recordFound =0;
     string line;
      while(getline(read, line)) {
      if(line == username+".dat" )
      {
         recordFound = 1 ;
         break;
```

```
            }
            }
            if(recordFound == 1)
            {
            cout<<"\n Username already Register. Please choose another username
";
            getchar(); getchar();
            delay();
            read.close();
            return 0;
            }
        }
        read.close();
        ofstream out;
        out.open("db.dat", ios::app);
        out<     out.close();
        ofstream out1;
        string temp = username+".dat";
        out1.open(temp.c_str());
        out1<     out1<     out1<     out1.close();
        cout<<"\n Student Registered Successfully !!";
        cout<<"\n Please any key to continue..";
        getchar(); getchar();
        return 0;
    }
```

Код функції deleteAllStudents()

Він видаляє дані всіх студентів.

```
int deleteAllStudents()
{
```

```
cout<<"\n In delete all students !!";
cout<<"\n Please any key to continue..";
//todo: implement this functionality
getchar(); getchar();
return 0;
}
```

Код функції deleteStudentbyRollno()

Ця функція видалить певні відомості про студента, використовуючи його номер в списку.

```
int deleteStudentbyRollno()
{
cout<<"\n Delete any Student by their Roll No \n";
cout<<"\n Please any key to continue..";
getchar();getchar();
return 0;
}
```

Код функції checkListOfStudentsRegistered()

Він містить список усіх зареєстрованих студентів.

```
int checkListOfStudentRegistered()
{
cout<<"\n List of All registered registered !!";
cout<<"\n Please any key to continue..";
getchar();
getchar();
return 0;
}
```

Код функції checkPresenseCountbyRollno()

Ця функція перевірить кількість присутніх студентів за номером у списку.

```
int checkPresenseCountbyRollno()
```

```
{
cout<<"\n Check presense count of any Student by Roll No \n";
cout<<"\n Please any key to continue..."    ;
getchar();getchar();
return 0;
}
```

Код функції getListOfStudentsWithTheirPresenseCount()

Він надає всім студентам кількість відвіданих занять.

```
int getListOfStudentsWithTheirPresenseCount()
{
cout<<"\n All Students with their Presense count !!";
cout<<"\n Please any key to continue..";
getchar();
getchar();
return 0;
```

Код функції checkStudentCredentials (рядок ім’я користувача, рядок пароль)

Щоб прочитати або відкрити файл a.dat, спочатку потрібно визначити його тип, а потім перетворити його, щоб його було легше відкривати. Це перевірка чи студент вже існує чи ні.

```
int checkStudentCredentials(string username, string password)
{
ifstream read;
read.open("db.dat");
if (read) {
int recordFound = 0;
string line;
string temp = username + password + ".dat";
cout<<"\n file name is : "< while(getline(read, line)) {
   if(line == temp)
```

```
    {
        recordFound = 1;
        break;
    }
}
if(recordFound == 0)
    return 0;
else
   return 1;
}
else
{
   return 0;
}
}
```

Код функції getAllStudentsbyRollNo()

Ця функція показує всіх студентів за їхнім списком.

```
int getAllStudentsbyRollNo()
{
cout<<"\n List of All Students by their Roll No \n";
cout<<"\n Please any key to continue..";
getchar();getchar();
return 0;
}
```

Код функції studentLogin()

Вхід студента викликає дві функції

1. studentView()
2. delay()

```
int studentLogin()
```

```
{
system("cls");
cout<<"\n -------- Student Login ---------";
studentView();
delay();
return 0;
}
```

Код функції studentView ()

Під час входу для студентів, студенти введуть своє ім’я користувача та пароль і позначатимуть свою відвідуваність, а також можуть перевірити кількість відвідувань.

```
int studentView()
{
cout<<"\n ------- Student Login-------- \n";
string username, password;
cout<<"\n Enter username : ";
cin>>username;
cout<<"\n Enter password : ";
cin>>password;
int res = checkStudentCredentials(username, password);
if(res  == 0)
{
  cout<<"\n Invalid Credentials !!";
  cout<<"\n Press any key for Main Menu..";
  getchar(); getchar();
  return 0;
}
int goBack = 0;
while(1)
{
```

```
system("cls");
cout<<"\n 1 Mark Attendance fo Today ";
cout<<"\n 2 Count my Attendance";
cout<<"\n 0. Go Back <- \n";
int choice;
cout<<"\n Enter you choice: ";
cin>>choice;
switch(choice)
{
   case 1: markMyAttendance(username); break;
   case 2: countMyAttendance(username); break;
   case 0: goBack = 1;break;
   default: cout<<"\n Invalid choice. Enter again ";
   getchar();
}
if(goBack == 1)
{
break; //break the loop
}
}
```

Код функції delay()

Це зберігає запис.

```
int delay()
{
for(int i = 0; i<3; i ++)
{
   cout<<"\n Saving Records ...";
   for(int ii = 0; ii<20000; ii ++)
   {
      for(int iii = 0; iii<20000; iii ++)
```

```
        { }
    }
}
cout<<"\n Exiting Now ...";
for(int i = 0; i<3; i ++){
   for(int ii = 0; ii<20000; ii ++) {
     for(int iii = 0; iii<20000; iii ++){
     }
    }
```

Код функції markMyAttendance(рядок імені користувача)

Ця функція позначає відвідування студента на сьогодні.

```
int markMyAttendance(string username)
{
cout<<"\n Mark Attendance for today !!";
cout<<"\n Please any key to continue..";
getchar();
getchar();
return 0;
}
```

Код функції countMyAttendance(string username);

Підрахувати відвідуваність студента.

```
int countMyAttendance(string username)
{
cout<<"\n Count my attendance for today !!";
cout<<"\n Please any key to continue..";
getchar();getchar();
return 0;
```

ВИСНОВКИ

Під час розробки програми мною були засвоєні основні функції мови програмування С++. Створений додаток для позначення відвідуваності студентів, дає можливість користувачу виконувати основні дії такі як: позначати чи присутній студент на занятях чи ні, підраховувати кількість пропущених занять тощо. Дана програма також дозволяє викладачам позначати присутніх/відсутніх студентів за допомогою телефона. Використовувати її можна у будь яких навчальних закладах.

Робота над дипломною роботою бакалавра додала мені практичного досвіду використання мови програмування С++ для створення програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Григорович В.Г. Дрогобич : Редакційно-видавничий відділ Дрогобицького державного педагогічного університету імені Івана Франка, 2016. 1400 с.

2. Григорович В.Г. Методичні вказівки до виконання лабораторних і практичних завдань з курсу «Алгоритмізація та програмування». Частина 1 : навч. посібник. 2012. 237с.

3. Ковалюк Т.В. Алгоритмізація та програмування: підручник з грифом МОН України. 2021. 400с.

4. Кравець П.О. Об'єктно-орієнтоване програмування : навч. посібник. 2012. 624с.

5. Ковалюк Т. В. «Алгоритмізація та програмування». Львів : Магнолія-2006, 2013. 400 с.

6. URL: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio

7. Прат С. Лекції та вправи / С. Прат. -Харків: 2003. - 476с.

8. Шрайнер Д., Ву М., Нейдер Дж., Девис Т. OpenGL. Руководство по программированию / Пер. с англ. - 4-е изд. - СПб.: Питер, 2006. - 623 с.

9. Bjarne Stroustrup, "Programming -- Principles and Practice Using C++", 2nd edition. 2012. 400.

10. Бабаків Д., Бабаків ДО «Програмування в середовищі C++ Builder». – Мінськ, 2004 р. 326с.

11. Герберт Шилдт «Мистецтво програмування на C++». – Спб.: «БХВ», 2005.

12. Культин Н. «C/C++ у завданнях і прикладах». – Спб.: «БХВ», 2005.

13. Мозковий М.В. «C++ Майстра-клас. 85 нетривіальних проектів, рішень і завдань». – М .: «Наука й техніка», 2007. 345с.

14. Мэтью Уилсон «C++: практичний підхід до рішення проблем програмування». – М.: «Кудиц-образ», 2006. 600с.

15. .Павловская Т.А. «С/С++. Програмування мовою високого рівня». – Спб.: «Питер», 2003. 346с.

16. Павловская Т.А., Щупак Ю. А. «С++. Объектно-ориентированное програмування: Практикум.». Спб.: «Питер», 2006. 300с.

17. Рейсдорф К., Хендерсон К. «Borland C++ Builder. Освой самостійно». М.: «Видавництво БІНОМ», 2003. 230с.

18. Штерн В. «Основи С++: Методи програмної інженерії». М.: «Лорі», 2003. 280с.

19. Якушев Д. «Філософія програмування мовою С++». – М.: «Новий видавничий будинок», 2004.300с.

20. Зубенко В.В. Програмування: навчальний посібник (гриф МОН України) / В.В. Зубенко, Л.Л. Омельчук. - К.: ВПЦ "Київський університет", 2011. 623 с.

21. Нікітченко М.С. Теоретичні основи програмування : навчальний посібник / М.С Нікітченко Ніжин : Видавництво НДУ імені Миколи Гоголя, 2010, - 121с.

22. Л.Л. Омельчук, А.Б. Ставровський. Методичні вказівки з підготовки та оформлення кваліфікаційних та курсових робіт для студентів факультету комп’ютерних наук та кібернетики. К.: Київський національний університет імені Тараса Шевченка, 2017. 300с.

23. URL: https://acode.com.ua/uroki-po-cpp/

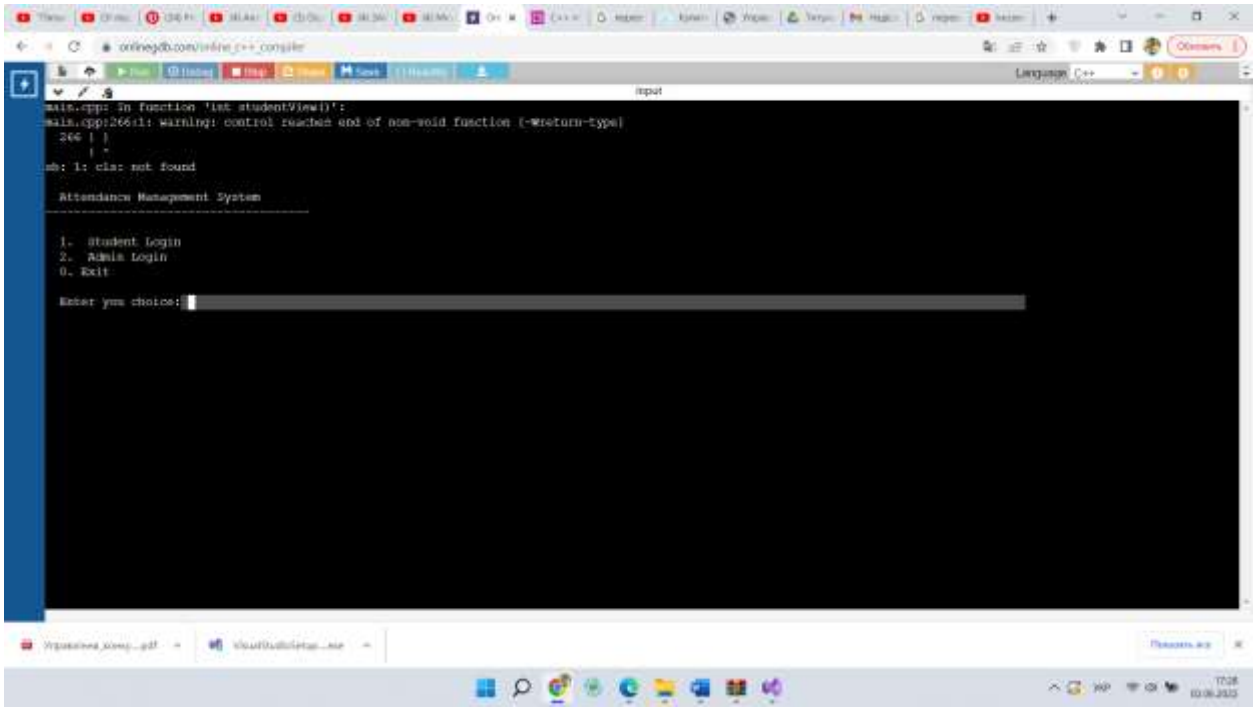
24. URL: http://www.znannya.org/?view=Cpp_basics

25. URL: http://apeps.kpi.ua/dlia-choho-vykorystovuiut-cpp

ДОДАТКИ

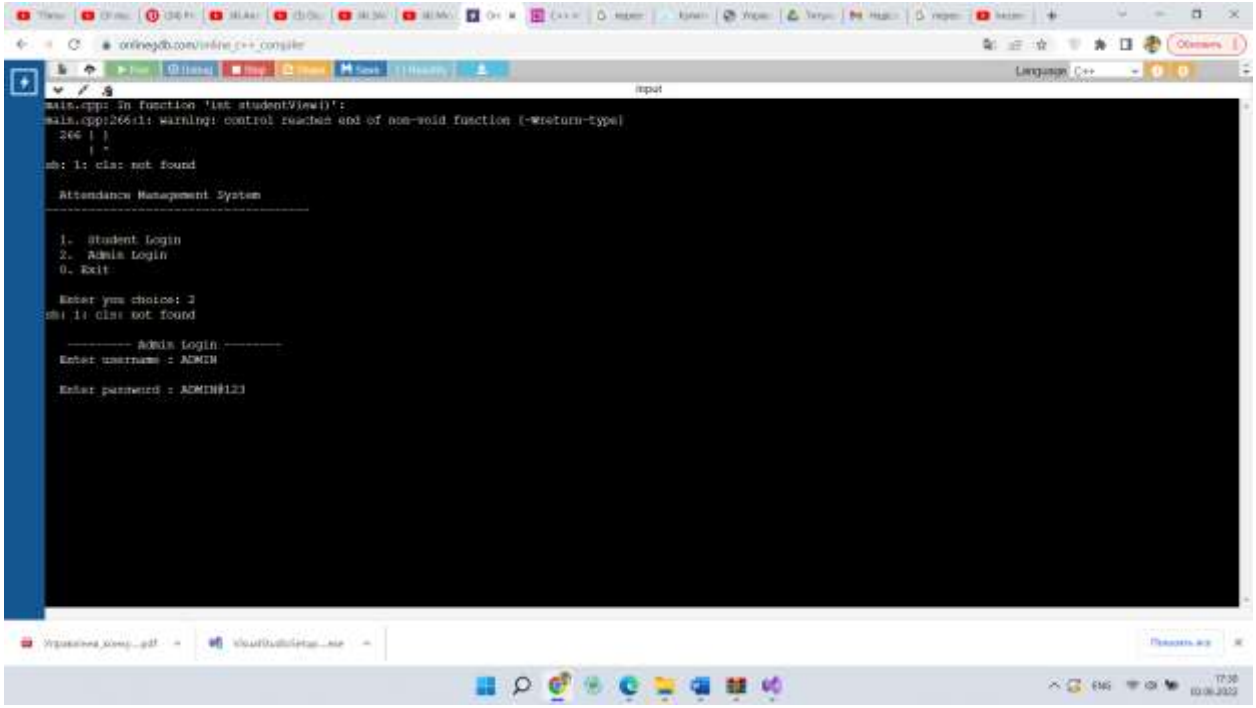
ДОДАТОК А

Головний екран виводу



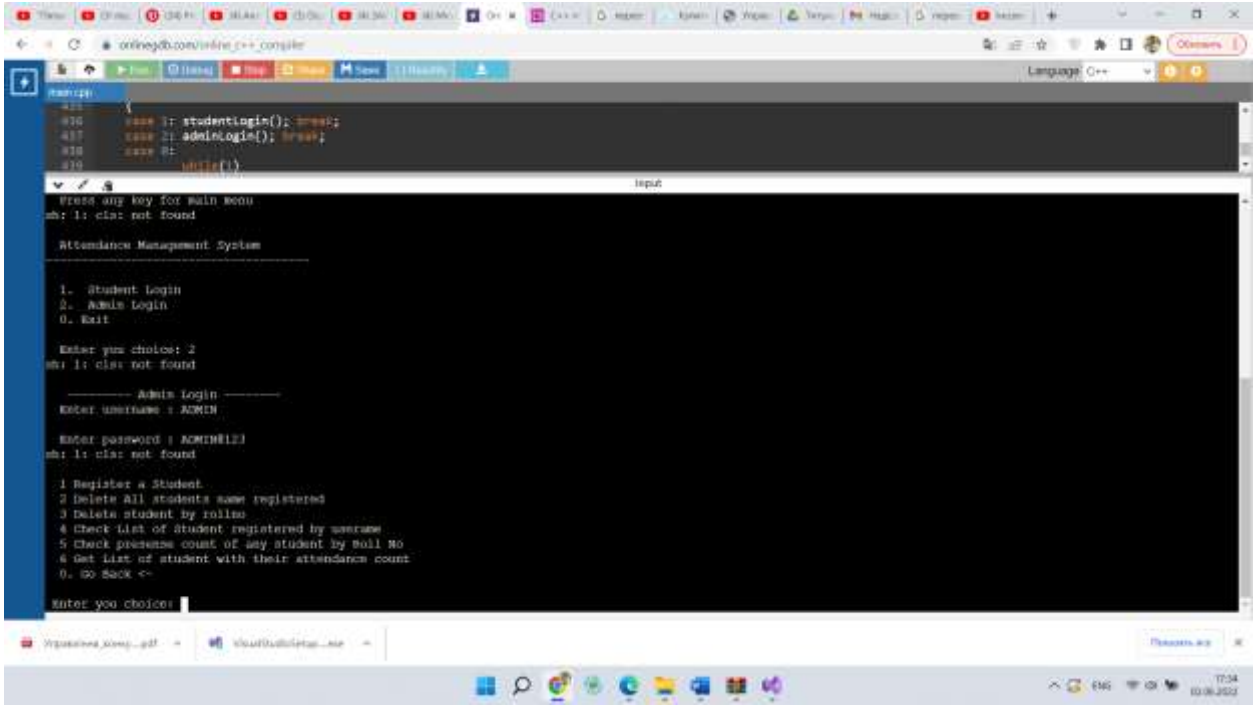
ДОДАТОК Б

Екран входу адміністратора



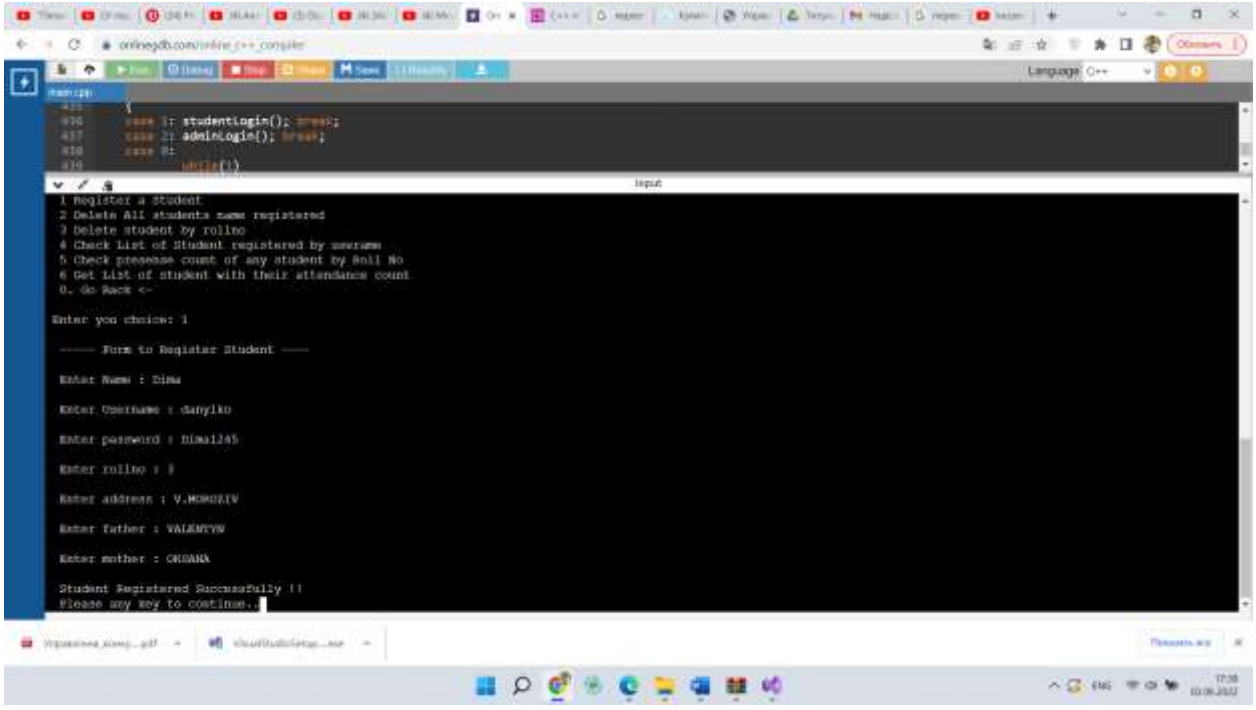
ДОДАТОК В

Екран подання адміністратора



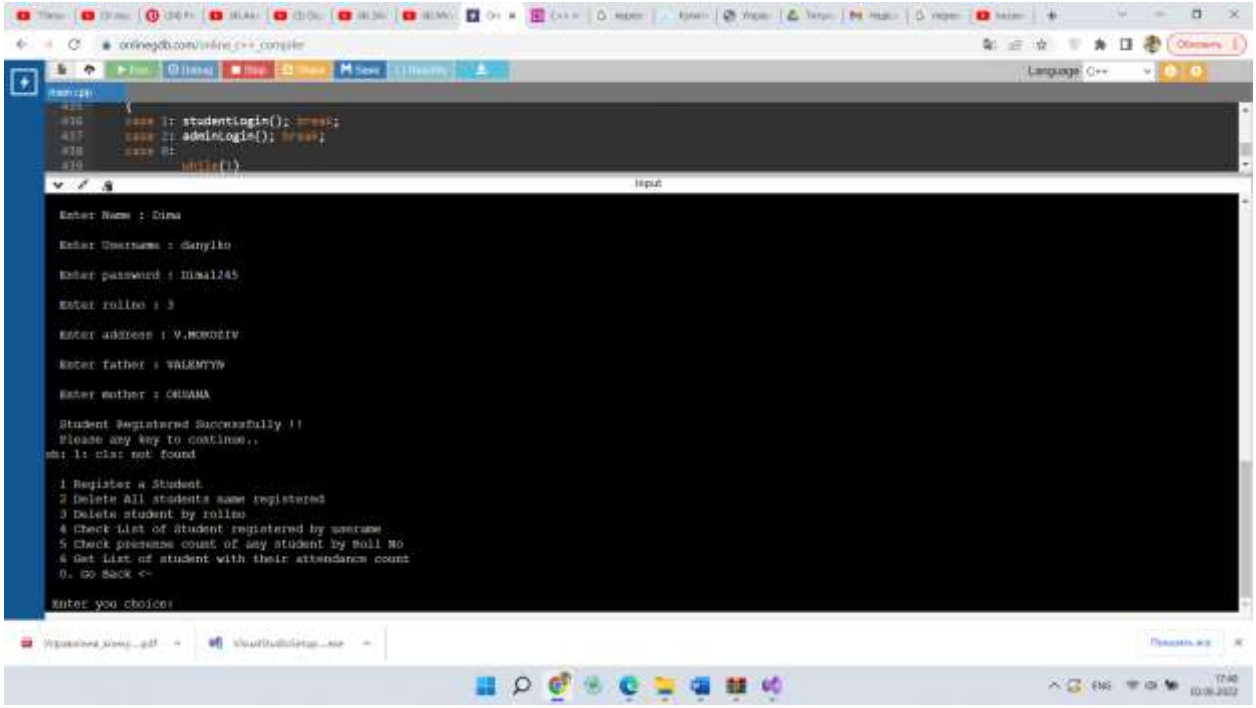
ДОДАТОК Г

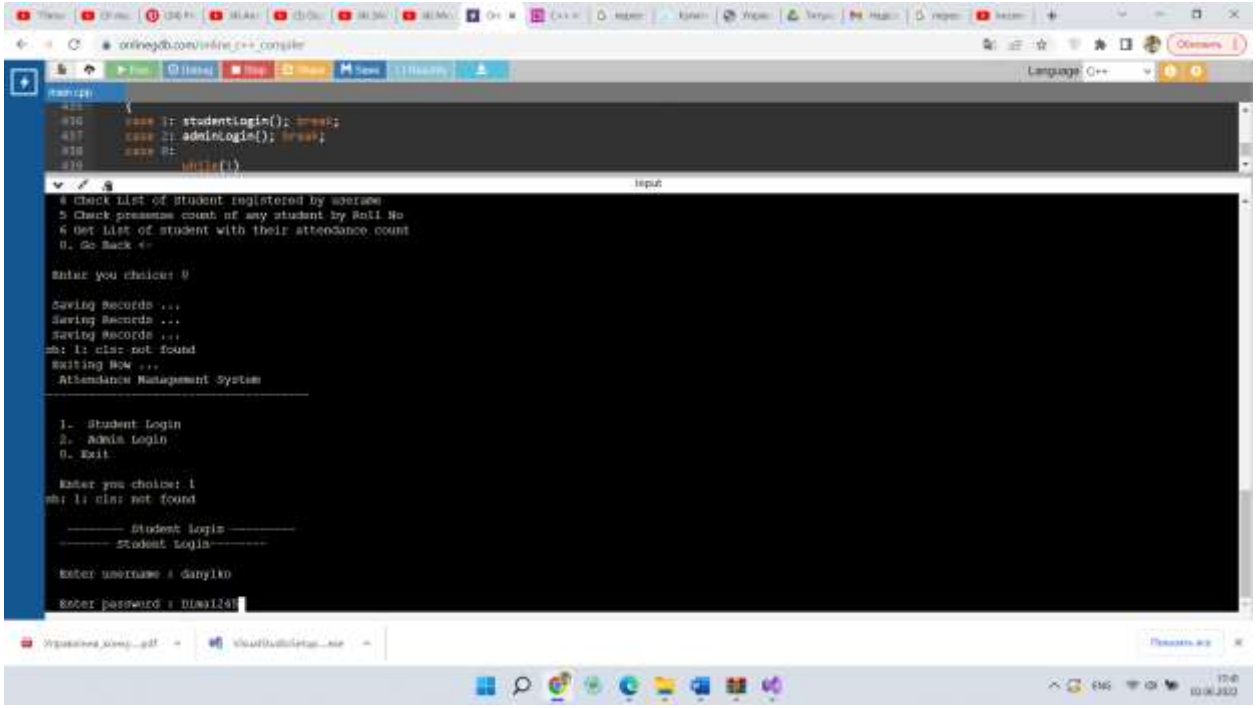
Екран реєстрації студентів



ДОДАТОК Д

Екран зареєстрованого студента



ДОДАТОК Е

Екран входу студента