

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт
з дисципліни
“Людино-машинна взаємодія”
для студентів
спеціальності 7.05010301
“Програмне забезпечення систем”
та 7.05010302 “Інженерія програмного забезпечення”
денної форми навчання

Частина 1

2014

Методичні вказівки до лабораторних робіт з дисципліни “Людино-машинна взаємодія” для студентів спеціальності 7.05010301 “Програмне забезпечення систем” та 7.05010302 “Інженерія програмного забезпечення” денної форми навчання. Частина 1/Укл.: С.М. Сердюк, Ж.К. Камінська, О.О. Степаненко. – Запоріжжя: ЗНТУ, 2014. – 58с.

Укладачі: С. М. Сердюк, доцент, к.т.н.,
О.О. Степаненко, доцент, к.т.н.
Ж.К. Камінська, асистент

Рецензент: С.К. Корнієнко, доцент, к.т.н.

Відповідальний
за випуск: С.М. Сердюк, доцент, к.т.н.,

Затверджено
на засіданні кафедри
“Програмних засобів”

Протокол №3
від 20 жовтня 2014

ЗМІСТ

ВСТУП.....	4
1 ЛАБОРАТОРНА РОБОТА №1	
Збір, аналіз і класифікація вхідної\вихідної інформації по заданому варіанту інтерфейсу об'єкту, що проектується....	5
1.1 Мета роботи.....	5
1.2 Основні теоретичні відомості	5
1.3 Завдання на лабораторну роботу.....	13
1.4 Зміст звіту.....	13
1.5 Контрольні запитання.....	13
2 ЛАБОРАТОРНА РОБОТА № 2	
Етапи проектування і створення користувацького інтерфейсу. Розробка структури, форм і макета екрану інтерфейсу.....	14
2.1 Мета роботи.....	14
2.2 Основні теоретичні відомості.....	14
2.3 Приклад розробки структури та макета екрану інтерфейсу.....	33
2.4 Завдання до роботи.....	35
2.5 Зміст звіту.....	36
2.6 Контрольні запитання.....	36
3 ЛАБОРАТОРНА РОБОТА № 3	
Етапи проектування користувацького інтерфейсу. Вибір елементів керування. Текст і числа.....	37
3.1 Мета роботи.....	37
3.2 Основні теоретичні відомості.....	37
3.3 Завдання до роботи.....	57
3.4 Зміст звіту.....	58
3.5 Контрольні запитання.....	58
РЕКОМЕНДОВАНА ЛІТЕРАТУРА.....	58
ЧАСТИНА 2	

ВСТУП

Людино-машинний інтерфейс (НМІ – англ. human-machine interface) – є сукупністю засобів і методів, за допомогою яких людина (користувач) взаємодіє з різними, найчастіше складними, такими, що складаються з безлічі елементів, машинами і пристроями.

Інтерфейс це двонаправлений пристрій: отримавши команди від користувача і виконавши їх, він видає інформацію про результати їх виконання, тими засобами, що є у нього (візуальними, звуковими і т. ін.). Приймаючи дану інформацію, користувач вводить нові команди наданими в його розпорядження засобами (кнопки, перемикачі, сенсори, голосом, і т. ін.).

Метою даних лабораторних робіт є побудова людино-машинних інтерфейсів (ЛМІ) для взаємодії з будь-якими об'єктами, у тому числі і віртуальними. Створення подібного інтерфейсу повинне складатися з певних взаємозв'язаних етапів.

У роботі №1 проводиться аналіз необхідної інформації, визначення категорій і вимог користувачів до інтерфейсу. У роботі №2, на основі функцій об'єктів, що були виявлені в роботі №1, виконується другий етап проектування інтерфейсу – розробка його структури, форм і макетів екрану. На наступному етапі, в роботі №3, розглянуті питання реалізації елементів управління з врахуванням основ дизайну форм. У роботі №4 розглянуто останній етап проектування в ході якого вивчається інформаційна графіка, використання миші та клавіатури. У роботі №5 розглянуто метод оцінки якості діяльності людини-оператора системи «людина-техніка-середовище» на основі узагальненого структурного методу А.І.Губінського.

Варіанти завдань для проектування ЛМІ студенти отримують від викладача. Допускається розробка інтерфейсу системи обраної самим студентом, після узгодження предметної області з викладачем.

1 ЛАБОРАТОРНА РОБОТА № 1

Збір, аналіз і класифікація вхідної/вихідної інформації по заданому варіанту інтерфейсу об'єкту, що проектується

1.1 Мета роботи

Вивчити перший етап проектування інтерфейсів, в ході якого необхідно: зібрати, проаналізувати та класифікувати вхідну інформацію щодо об'єкта що проектується.

1.2 Основні теоретичні відомості

Перш ніж приступати до розробки і побудови будь-якої системи слід з'ясувати, які завдання хочуть вирішувати користувачі з її допомогою, і як вони звикли працювати. Необхідно звернути увагу на ті обмеження, які накладають їх комп'ютерні системи на технічне і програмне забезпечення. Пропоноване вами рішення повинне відповідати не лише сьогоденним, але і майбутнім потребам користувачів.

Існує низка ключових запитань які слід поставити на першому етапі проектування – етапі аналізу інформації про користувачів:

- визначення типу користувачів;
- аналіз завдань, що стоять перед ними;
- аналіз робочого середовища користувачів;
- збір вимог, що пред'являються користувачами;
- розробка сценарію дій користувача.

Перший крок: визначення типу користувачів.

Припускає відповідь на питання: «Що представляє собою користувач?», тобто дозволяє скласти уявлення про вік, переваги користувачів, отримати іншу необхідну інформацію.

Другий крок: аналіз завдань, що стоять перед користувачами.

Це визначення потреб користувачів і варіантів того, як вони збираються вирішувати свої завдання.

Незалежно від методу аналізу завдань необхідно отримати відповіді на наступні питання.

1. Які завдання вирішують користувачі?
2. Які завдання є найбільш важливими?
3. Які кроки робляться для вирішення завдань?

4. Які цілі переслідують користувачі при рішенні тих або інших завдань?

5. Яку інформацію необхідно мати для виконання завдань?

6. Який інструментарій (комп'ютери і т. ін.) використовуються для вирішення завдань?

7. Який очікуваний підсумок від рішення задачі?

Аналіз цього кроку необхідно оформити у вигляді компонентно-функціональної структури (див. табл. 1.1, 1.2) та функціонально-об'єктної структури (див. табл. 1.3, 1.4).

Компонентно-функціональна структура має наступні стовпці:

1 – номер мети, для якої був створений програмний продукт;

2 – номер задачі, яку вирішує користувач;

3 – номер процедур, які необхідно виконати для рішення задачі;

4 – назва цілей, задач і процедур.

Функціонально-об'єктна структура відображає склад компонентів, що приймають участь в реалізації процедур (функцій і т. ін.). Вона має наступні стовпці:

1 – номери процедур (беруться з компонентно-функціональної структури);

2 – найменування ергатичних елементів (спеціалістів), що приймають участь у виконанні процедур;

3 – найменування засобів інтерфейсу, за допомогою яких ергатичний елемент здійснює взаємодію з засобами праці;

4 – найменування знарядь праці, за допомогою яких реалізується процедура;

5 – предмети праці, з яких утворюється продукт праці даної процедури;

6 – продукти праці, отримані в результаті виконання даної процедури.

Третій крок: аналіз робочого середовища користувачів.

Припускає відповідь на питання: «Де користувачі вирішують завдання, що стоять перед ними?». На цьому кроці визначається наступна інформація про характеристики середовища, які можуть чинити вплив на виконання користувачами своєї роботи:

- фізичній стороні робочого середовища (освітлення, шум, робочий простір, температура, наявність комп'ютерів і т.д.);

- місце роботи користувача і міри його мобільності (офіс, квартира і т.д.);

- питаннях ергономіки, умовах праці (чи задіються зір, слух, робота ведеться стоячи/сидячи, на клавіатурі і т.д.);

- особливих запитах (рівень підготовки, фізичний стан, інтерес до пізнавального процесу, особливості мови і можливі недоліки);

Четвертий крок: збір вимог, що пред'являються користувачами.

Припускає відповідь на питання: «Яку, з точки зору користувача, користь принесе їм пропонований продукт або інтерфейс?». Ключовими у даному контексті являються наступні питання.

1. Які основні технології потрібні користувачам?
2. Скільки користувачі готові заплатити за продукт?
3. Хто встановлює продукт?
4. Хто супроводжуватиме продукт, коли він буде встановлений?

Існують деякі загальні для усіх користувачів вимоги, відповідно до яких новий програмний продукт повинен:

- скорочувати роботу з паперами;
- зменшувати помилки користувачів;
- автоматизувати існуючі ручні процеси;
- підвищувати швидкість здійснення транзакцій.

П'ятий крок: розробка сценарію дій користувача.

На другому кроці розробки інтерфейсу користувача ми з'ясували, які завдання вирішують користувачі, і які дії необхідно виконати для досягнення поставленої мети. Складно з високим ступенем точності визначити, що являю собою сценарій у порівнянні з завданням, що стоїть перед користувачем. Як правило, сценарій є описом дій, які виконуються користувачем. Можна представити сценарій, як послідовність завдань, що стоять перед користувачами, або подій, спрямованих на досягнення єдиної мети. Представимо сценарій дій користувача у вигляді функціонально-часової структури, наведеної на рис. 1.1. За масштабом функціональних сутностей наведена структура є процедурною мережею, так як вершинами в ній є процедури, виявлені при побудові компонентно-функціональної структури.

Усі ці фактори впливають на розробку продукту.

Приклад виконання першого етапу проектування.

Розглянемо покроковий аналіз вимог, що пред'являються користувачами на прикладі розробки кавового автомата (КА).

Користувач – це особа, яка використовує вже готову (діючу) систему для виконання конкретної мети, тобто користувачем КА являється покупець, мета якого – отримати необхідний напій.

Для того, щоб КА міг виконувати поставлену перед ним мету, необхідно забезпечити його технічне обслуговування наладчиком. Завданнями наладчика також є поповнення запасів КА і вилучення виручки від продажу напоїв.

Таким чином виконані завдання *першого кроку* – визначені типи користувачів КА. Обмежень для користувачів КА практично немає. Єдина вимога – володіння українською мовою (мова інтерфейсу КА).

Обмеженнями для наладчиків є достатній рівень володіння комп'ютером, вміння відновлювати збої і неполадки в автоматі.

Другий крок: завдання, що стоять перед покупцями і наладчиками. Результати аналізу приведені в табл.1.1, табл. 1.2, табл. 1.3, табл. 1.4.

Таблиця 1.1 – Компонентно-функціональна структура для покупців

Мета	Задачі	Процедури	Дії користувачів і системи
1	2	3	4
1			Отримати необхідний напій
	1.1		Вибрати і сплатити напій
		1.1.1	Вибрати напій із запропонованого списку, якщо напій закінчився, то видати відповідне повідомлення
		1.1.2	Вибрати кількість цукру
		1.1.3	Внести потрібну суму грошей упродовж 10 секунд, інакше автомат переходить в початковий стан
		1.1.4	Сплатити вартість напою або повернути внесені гроші. У разі повернення грошей повернутися в початковий стан
		1.1.5	Якщо внесена сума перевищує вартість напою, то видати здачу; якщо менше – то видати відповідне повідомлення "Недостатньо коштів" і повернутися в пункт 1.1.4

Продовження таблиці 1.1

1	2	3	4
		1.1.6	Отримати готовий напій

Таблиця 1.2 – Компонентно-функціональна структура для наладчиків

Мета	Задачі	Проце- дури	Дії наладчика і системи
1	2	3	4
1			Обслуговування КА
	1.1		Отримання інформації про стан автомата
		1.1.1	Інформація про стан балансу
		1.1.2	Інформація про залишок цукру
		1.1.3	Інформація про залишок кави
		1.1.4	Інформація про залишок чаю
		1.1.5	Інформація про залишок склянок
		1.1.6	Інформація про час приготування напою
	1.2		Поповнення ресурсів автомата
		1.2.1	Поповнення запасу чаю
		1.2.2	Поповнення запасу кави
		1.2.3	Поповнення запасу цукру
		1.2.4	Поповнення запасу склянок
	1.3		Зняття балансу
	1.4		Перегляд статистики по напоях
		1.4.1	Статистика з продажу напоїв в грамах
		1.4.2	Статистика з продажу напоїв у відсотках
		1.4.3	Загальна статистика продажу напоїв по датах
	1.5		Ремонтування КА
		1.5.1	Отримання інформації про аварійні ситуації
		1.5.2	Усунення несправностей в КА

Таблиця 1.3 – Функціонально-об’єктна структура для покупців

Номер проце- дури	Ергати- чний елемент	Засоби інтер- фейсу	Знаряд- дя праці	Предмети праці	Продукти праці
1	2	3	4	5	6
1.1.1	Поку- пець	Клавіатура та дисплей	Рука	Інтерфейс покупця	Вибраний напій або інформа- ційне повідом- лення
1.1.2	Те ж	Те ж	Те ж	Те ж	Вибрана кількість цукру
1.1.3	Те ж	Те ж	Те ж	Те ж	Внесена сума
1.1.4	Те ж	Те ж	Те ж	Те ж	Оплата або отримані гроші
1.1.5	Те ж	Те ж	Те ж	Те ж	Здача або інформаційне повідомлення
1.1.6	Те ж	Те ж	Те ж	Те ж	Готовий напій

Таблиця 1.4 – Функціонально-об’єктна структура для наладчиків

Номер проце- дури	Ергати- чний елемент	Засоби інтер- фейсу	Знаряд- дя праці	Предме- ти праці	Продукти праці
1	2	3	4	5	6
1.1.1	Наладчик	Клавіа- тура та дисплей КА	Рука	Системн е меню КА	Інформація про стан балансу
1.1.2	Те ж	Те ж	Те ж	Те ж	Інформація про залишок цукру
1.1.3	Те ж	Те ж	Те ж	Те ж	Інформація про залишок кави
1.1.4	Те ж	Те ж	Те ж	Те ж	Інформація про залишок чаю

Продовження таблиці 1.4

1	2	3	4	5	6
1.1.5	Те ж	Те ж	Те ж	Те ж	Інформація про залишок склянок
1.1.6	Те ж	Те ж	Те ж	Те ж	Інформація про час приготування напою
1.2.1	Те ж	Те ж	Те ж	Те ж	Поповнені запаси чаю
1.2.2	Те ж	Те ж	Те ж	Те ж	Поповнені запаси кави
1.2.3	Те ж	Те ж	Те ж	Те ж	Поповнені запаси цукру
1.2.4	Те ж	Те ж	Те ж	Те ж	Поповнені запаси склянок
1.3	Те ж	Те ж	Те ж	Те ж	Виручка
1.4.1	Те ж	Те ж	Те ж	Те ж	Інформація про продаж напоїв в грамах
1.4.2	Те ж	Те ж	Те ж	Те ж	Інформація про продаж напоїв у відсотках
1.4.3	Те ж	Те ж	Те ж	Те ж	Інформація про продаж напоїв по датах
1.5.1	Те ж	Те ж	Те ж	Те ж	Інформація про аварійні ситуації
1.5.2	Те ж	Те ж	Те ж	Те ж	Новий стан КА (усунення несправностей)

Третій крок: Робоче середовище користувачів.

Можливість використання цього КА в навчальних закладах, на виробництві, в торговельних центрах, різних фірмах та ін. Мінімальні вимоги до комп'ютерної системи: процесор не нижче Duron 800 MHz, RAM 128 Мб, Video 32 Мб, HDD 1 Гб. Освітленість повинна бути не менш 50 люксів.

Четвертий крок: Вимоги покупців і наладчиків.*Вимоги покупців:*

- доступність до КА в будь-який час доби;
- зручний і простий інтерфейс;
- на приготування напою вимагається не більше 10 секунд часу;

- забезпечити функцію повернення грошей у разі внесення суми меншої вартості напою;

- отримати здачу у разі внесення суми більшої вартості напою.

Вимоги наладчиків:

- можливість працювати з системним меню, не зупиняючи роботу КА;

- захист системного меню паролем;

- забезпечення поточною інформацією про стан апарату;

- збір статистики з продажу напоїв.

П'ятий крок: Розробка сценарію дій користувача.

На підставі таблиці 1.1 – компонентно-функціональної структури складемо функціонально-часову структуру для користувачів (см. рис.1.1).

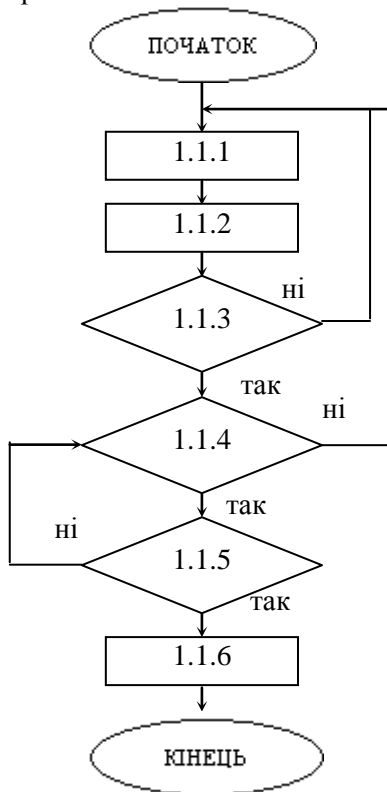


Рисунок 1.1 – Функціонально-часова структура для користувачів

Сценарій дій налагодчика розробляється аналогічно.

1.3 Завдання до роботи

1.3.1 Ознайомитися з пунктом 1.2 і конспектом лекцій.

1.3.2 Обрати та узгодити з викладачем об'єкт для проектування.

1.3.3 Проаналізувати та класифікувати вхідну інформацію об'єкта що проектується. Аналіз виконати у вигляді наведених у п. 1.2 п'яти кроків.

1.3.4 Оформити звіт по роботі.

1.3.5 Відповісти на контрольні питання.

1.4 Зміст звіту

1.4.1 Тема та мета роботи.

1.4.2 Завдання до роботи.

1.4.3 Аналіз об'єкта що проектується. Привести результати виконання п'яти кроків першого етапу проектування. Результати виконання другого кроку представити у вигляді компонентно-функціональної та функціонально-об'єктної структур.

1.4.4 Висновки, за результатами виконання роботи.

1.5 Контрольні запитання

1.5.1 Які кроки необхідно виконати при реалізації першого етапу проектування інтерфейсів і в чому вони полягають?

1.5.2 Для чого потрібне визначення типу користувачів?

1.5.3 Основна ідея аналізу завдань, що стоять перед користувачами.

1.5.4 Що являє собою компонентно-функціональна структура?

1.5.5 Що являє собою функціонально-об'єктна структура?

1.5.6 Які вам відомі загальні для усіх користувачів вимоги до нового програмного продукту?

1.5.7 На що необхідно звернути увагу при аналізі робочого середовища користувачів?

1.5.8 Що являє собою функціонально-часова структура?

ЛАБОРАТОРНА РОБОТА № 2

Етапи проектування і створення користувацького інтерфейсу. Розробка структури, форм і макета екрану інтерфейсу

2.1 Мета роботи

Ознайомитися з основними етапами проектування і створення користувацького інтерфейсу. Розробити структуру, форму і макет екрану інтерфейсу.

2.2 Основні теоретичні відомості

Другий етап розробки складається з певних кроків, які виконуються у заданій послідовності. Перш ніж приступити до програмування, потрібно виконати наступні кроки розробки інтерфейсу:

- визначення цілей та операцій інтерфейсу;
- візуальна ієрархія;
- візуальний потік інтерфейсу;
- групування та вирівнювання елементів інтерфейсу;
- застосування шаблонів для компоновання сторінок.

1 Визначення цілей та операцій інтерфейсу.

Перший крок, напевно є найважливішим у процесі розробки інтерфейсу. На цій стадії розробки необхідно:

- виділити об'єкти, дані і дії зі сценаріїв і завдань, що стоять перед користувачами;
- переглянути та уточнити список об'єктів і дій спільно з користувачами;
- накреслити діаграму взаємодії між об'єктами.

Перш за все, слід визначити початковий набір об'єктів дій користувача на підставі інформації, зібраної на першому етапі, а також сценаріїв, розроблених на другому етапі. Для складання списку об'єктів і даних (іменників), а також дій (дієслів), що застосовуються до них - просто підкресліть всі іменники у ваших сценаріях і задачах, і обведіть кружком всі дієслова.

Вивчіть всі матеріали, що мають відношення до задач та сценаріїв. Записуйте якомога більше об'єктів і дій, не турбуючись про термінологію і багатослівні описи, поки не отримаєте остаточний варіант, без повторень, і в якому, можливо, будуть міститися об'єкти і

дії, не зазначені в сценаріях користувачів. Необхідно також визначити об'єкти пристроїв загального призначення, за допомогою яких користувачі зможуть виконувати вказані вами дії. При цьому не обов'язково, що всі об'єкти складеного списку стануть об'єктами інтерфейсу у фінальній розробці. Деякі з них можуть перетворитися на специфічні види основних об'єктів.

Схема відносин між об'єктами користувацького інтерфейсу відрізняється від традиційних схем програмування. Об'єкти для користувача інтерфейсу необов'язково паралельні об'єктам програмування або бізнес - об'єктам.

Одна з цілей розробки користувацького інтерфейсу полягає в тому, щоб максимально приховати складність внутрішніх бізнес - моделей та моделей програмування. Сховайте від користувачів функціонування «системних» об'єктів (списки клієнтів, заброньованих місць, готелів або баз даних), вони повинні бачити ці об'єкти тільки через систему пошуку або в списках наявних елементів. З точки зору користувачів, мова йде про великі контейнери, що перебувають десь в комп'ютерній системі, і зберігають всі об'єкти, з якими вони працюють. Користувачам не потрібно знати про ці об'єкти більше, ніж потрібно для виправлення, або зберігання інформації.

2 Візуальна ієрархія.

Концепція візуальної ієрархії грає важливу роль у всіх формах графічного дизайну. Її суть полягає в тому, що найважливіше має виділятися на фоні всього, а найменш важливу частину сторінки виділяти не потрібно. Розглянемо цю концепцію на прикладі. На рисунку 2.1 показаний текст, в якому цілком відсутня візуальна ієрархія.

Приглашаем всех горожан и гостей нашего города на празднование Дня Независимости! На празднике Вас будут ждать вокально-инструментальные ансамбли и различные театральные постановки. Также для детей приготовлено специально отведенное место, где будут проводиться конкурсы и выступления клоунов. Вы сможете вкушать блюда национальной кухни, а по завершении всего праздника Вас порадует великолепный фейерверк. Когда: 24 августа в 15:00. Где: бульвар Шевченко. В течение всего дня будет ходить муниципальный транспорт, который доставит Вас из любой точки города прямо на празднование! Приходите, и хорошее настроение Вам обеспечено!

Рисунок 2.1 – Текст з відсутністю візуальної ієрархії

У даному прикладі складно визначити найбільш важливу інформацію. виправимо це за допомогою порожнього простору. На рисунку 2.2 тепер можна побачити різні групи інформації. Однак текст, який впадає в око в першу чергу - «Запрошуємо всіх городян і гостей нашого міста на ». Він знаходиться вгорі, окремо від іншого тексту, і займає лівий верхній кут, куди всі носії мов з написанням зліва направо дивляться, перш за все. Це додасть йому невиправдану значимість.

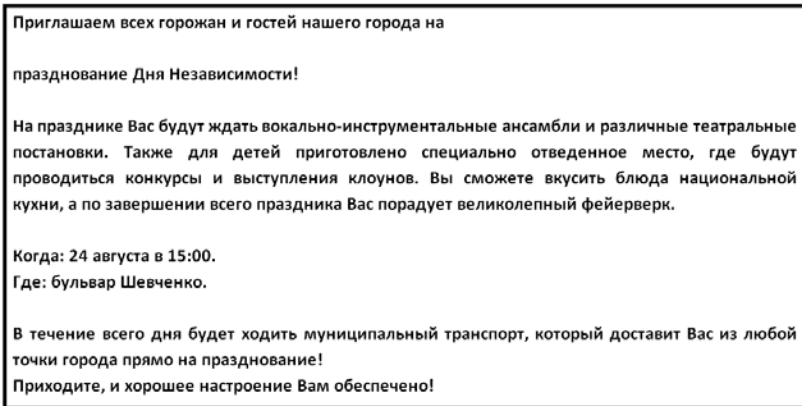


Рисунок 2.2 – Відформатований текст з різними групами інформації

Для вирішення цієї проблеми використовуємо друкарську розмітку і позиціонування (див. рис. 2.3).

Великий, жирний шрифт підкреслює важливість тексту. Найважливіший рядок виділений гігантським шрифтом. Другий за важливістю рядок надрукований великим, але не гігантським шрифтом. Для звичайного тексту використовується шрифт звичайного розміру.

Візуальну ієрархію допоможуть спроектувати наступні механізми:

- перевага верхньому лівому куту;
- порожній простір;
- контрастні шрифти: чим більше і жирніше, тим важливіше інформація;
- контрастні кольори для фону і переднього плану;

- позиціонування, вирівнювання і відступи. Зміщений вправо текст є другорядним по відношенню до того, що знаходиться над ним;
- графіка, наприклад лінії, рамки і кольорові панелі.

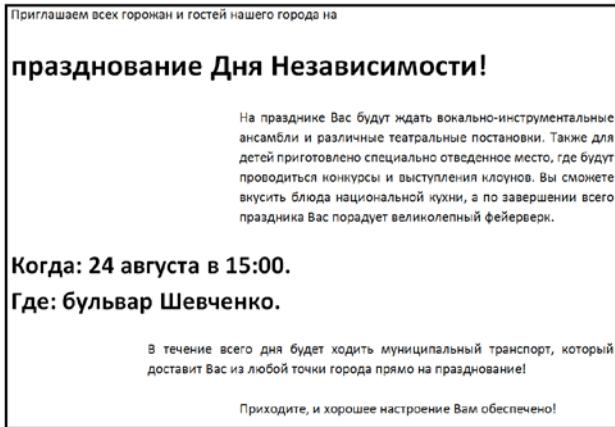


Рисунок 2.3 – Форматований текст за допомогою друкарської розмітки та позиціонування

3 Візуальний потік інтерфейсу.

Візуальний потік визначає траєкторію переміщення, погляду читача під час перегляду сторінки. Він тісно пов'язаний з візуальною ієрархією – добре продумана візуальна ієрархія визначає на сторінці точки фокусування, що привертають увагу до найбільш важливих елементів, а візуальний потік переводить погляд читачів з цих точок на менш важливу інформацію. Необхідно забезпечити контроль над візуальним потоком на сторінці, щоб користувачі проходили по потрібному шляху в правильній послідовності.

Точки фокусування – це точки, на яких погляд людини зупиняється незалежно від його бажання. Зазвичай погляд проходить від найсильнішої до найслабкішої точки фокусування. Точки фокусування можна створювати безліччю різних способів, наприклад, за допомогою порожнього простору, контрастності, великих шрифтів, заливкою кольором, ліній, що сходяться, різних кордонів, гарнітур, руху. На сторінці повинно бути лише кілька точок, тому що їх велика кількість послаблює значимість один одного.

Правила:

- зверху вниз і зліва направо – це візуальний потік за умовчанням;
- спочатку увагу приваблюють сильні точки фокусування і тільки потім слабкі;
- сенс вмісту сторінки може змінюватися в залежності від того, куди подивиться користувач.

При розробці користувацького інтерфейсу, у якому велике значення має послідовність дій або екранів (наприклад, майстер або діалогове вікно, в якому вибір на ранніх стадіях впливає на можливість подальшого вибору), необхідно продумати візуальний потік інтерфейсу.

Рекомендації: помістіть елементи управління і кнопки вздовж простого візуального шляху. В кінці шляху створіть посилання або кнопку, призначену для завершення завдання (Опублікувати, Назад на головну сторінку або ОК), яка буде переносити користувача на інше місце.

4 Групування та вирівнювання елементів інтерфейсу.

Вирівнювання – ще один, більш тонкий спосіб викликати у глядача асоціації одних елементів з іншими. Наприклад, якщо в діалоговому вікні є два набори кнопок, що знаходяться далеко один від одного, але виконують схожі функції, то можна розташувати одну групу під іншою, і зробити кнопки однакової ширини, щоб підкреслити їх схожість. Або якщо на сторінці дві форми, розділені текстом, то необхідно вирівняти ліві краї форм по одній невидимій лінії, а ліві краї текстових блоків – по іншій.

Властивостями, що характеризують компоновку елементів, які, на думку вчених споконвічно закладені в наші візуальні системи є:

- **Близькість** (див. рис. 2.4 № 1)

Для того щоб користувачі асоціювали елементи один з одним, необхідно помістити їх поряд. Це основа суворого угруповання вмісту та елементів управління в інтерфейсі.

- **Подібність** (див. рис. 2.4 № 2)

Користувачі також асоціюють елементи один з одним, якщо у двох елементів збігається форма, розмір, колір, або напрямок.

- **Безперервність** (див. рис. 2.5)

Користувач віддає перевагу безперервним лініям і кривим, що формуються шляхом вирівнювання невеликих елементів.

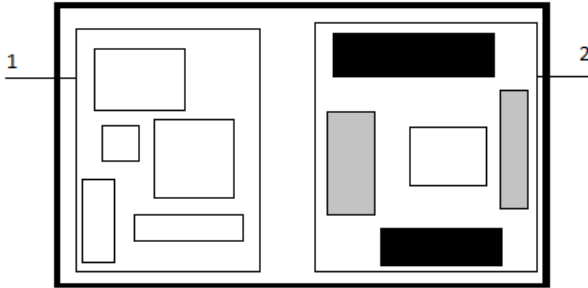


Рисунок 2.4 – Близькість і подібність

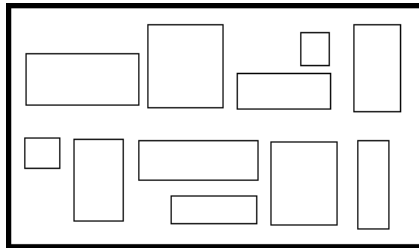


Рисунок 2.5 – Безперервність

- **Замкнутість** (см. рис. 2.6)

Користувач віддає перевагу невеликим замкнутим формам, наприклад прямокутники і плями порожнього простору, навіть тоді, коли вони не були намальовані спеціально. Групи елементів часто виглядають як замкнуті фігури.

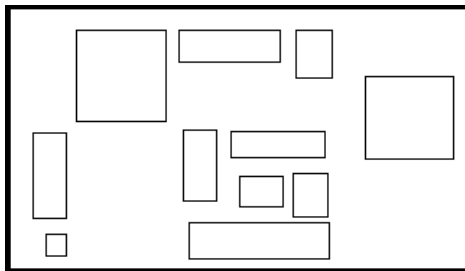


Рисунок 2.6 – Замкнутість

При розробці інтерфейсів необхідно використовувати розглянуті вище властивості в поєднанні один з одним (див. рис. 2.7).

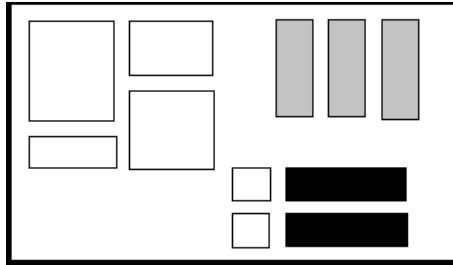


Рисунок 2.7 – Розглянуті вище властивості в поєднанні один з одним

В якості прикладу розглянемо сайт, побудований у відповідності з цими принципами (див. рис. 2.8). Зображення розмито, оскільки його вміст у даний момент не має принципового значення.

Поглянувши на екран, можна зрозуміти безліч речей, про які було розказано вище. По якому шляху був спрямований ваш погляд? Ймовірно, як показано стрілками на рисунку 2.8. Очевидно, що логотип є точкою фокусування – він знаходиться там, де зазвичай розташовуються заголовки веб-сторінки та виділений важким шрифтом. Найменування товарів притягують завдяки своєму шрифту, розміру і поділу за допомогою порожнього простору. Стовпець праворуч, насичений інформацією, виділений синім, так що на нього також звертають увагу.

На сторінці присутні три основні групи, в кожній з яких є великий виділений текст, більш дрібний, і блоки основного тексту. Ці групи формуються за рахунок подібності і близькості; порожній простір відокремлює їх один від одного. Стовпець праворуч протиставляється всьому іншому вмісту за рахунок сильно вираженого краю, в дійсності, навіть двох країв. На рисунку 2.8 дужками позначене угруповання елементів на цій сторінці.

Таке форматування робить свій внесок у створення візуальної ієрархії. Легко розрізнити заголовок, основний вміст і додаткову інформацію. У головному вмісті виділяються три блоки – ймовірно, короткі змісти описів з посиланнями на повний текст. У кожному є свій заголовок, підзаголовки і основний текст. Серед додаткової

інформації праворуч можна також виділити кілька груп, що розділяються заголовками.

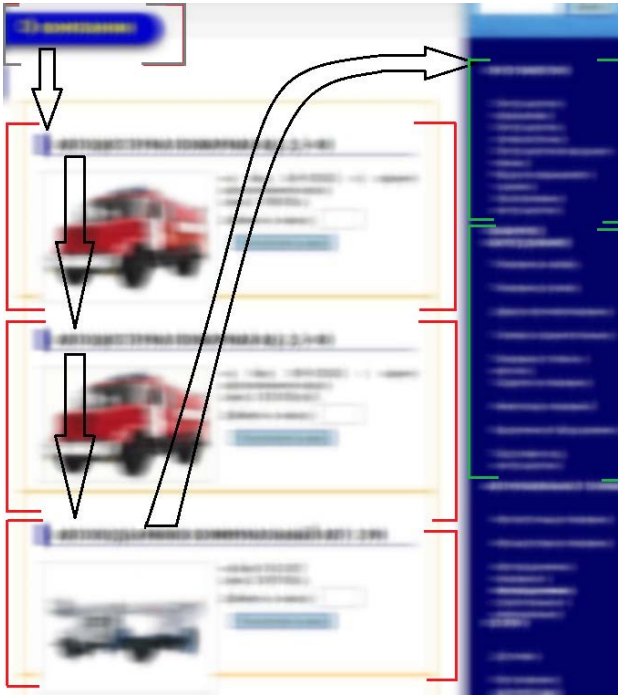


Рисунок 2.8 – Сайт, побудований у відповідності з принципами групування та вирівнювання

5 Застосування шаблонів для компоновання сторінок.

Компоновання сторінки – це мистецтво маніпулювання увагою користувача з метою висловити певний сенс, передати послідовність дій або організувати точки взаємодії. Існує п'ять основних елементів розмітки сторінки: візуальна ієрархія, візуальний потік, групування та вирівнювання, об'єднання цих трьох елементів і використання динамічних дисплеїв.

Дані шаблони пояснюють, як використовувати всі перераховані концепції компоновання сторінок.

Перші два шаблони, Visual Framework (Візуальна схема) [1,c.151] і Center Stage (Центральна сцена) [1,c.155] призначені для визначення візуальної ієрархії цілої сторінки, екрану або вікна, незалежно від типу інформації. Візуальну схему рекомендується визначити на самому початку роботи над проектом, так як від цього залежить, як будуть виглядати всі основні сторінки та вікна інтерфейсу.

• ***Visual Framework (Візуальна схема).***

Для дизайну кожної сторінки необхідно використовувати один і той же базовий макет, набір кольорів і стилістичних компонентів, але при цьому забезпечити достатню гнучкість, щоб поміщати на сторінки різний вміст. Суворі візуальні схеми, що повторюються на кожній сторінці, допомагає вмісту сторінки виділитися на її фоні. Те, що залишається постійним, відходить у свідомості на задній план, а на те, що змінюється, звертається особлива увага.

Даний шаблон використовується при створенні веб-сайтів з безліччю сторінок або користувацького інтерфейсу з безліччю вікон, тобто практично в будь-якому складному програмному забезпеченні. Використання інтерфейсу і навігація повинні бути простими.

Рекомендації. Розробіть загальне уявлення кожної сторінки або вікна в інтерфейсі, що створюється. Компонування елементів на головних сторінках і головних вікнах звичайно може трохи відрізнитися від сторінок другого рівня, але повинне мати певні риси, що пов'язують їх з іншою частиною інтерфейсу. Наприклад:

- колір: фонові кольори, кольори тексту і колірні аспекти;
- шрифти: для заголовків, підзаголовків, звичайного тексту і другорядного тексту;
- стиль написання і графіка: заголовки, імена, вміст, короткі описи і довгі блоки тексту, тобто всі випадки використання мови;
- показники: заголовки, логотипи, навігація у вигляді «хлібних крихт» і індекси для шаблону Card Stack (Пачка карток – буде розглянуто нижче), такі як вкладки або списки посилань;
- навігаційні засоби: набори стандартних посилань, кнопки ОК / Cancel (ОК / Відміна), кнопки повернення і виходу, а також навігаційні шаблони, такі як Global Navigation (Глобальна навігація) [1,c.109], Sequence Map (Карта послідовності) [1,c.123] і

Breadcrumbs (Хлібні крихти) [1,с.125]; техніки, що застосовуються для визначення іменованих заголовків (шаблон Titled Section (Іменовані розділи – буде розглянуто нижче);

- проміжки і вирівнювання елементів: поля сторінок, зазор між рядками, відстань між мітками і пов'язаними з ними елементами управління, а також вирівнювання тексту і міток;

- загальна схема компоновання: розташування елементів на сторінці, в стовпцях і рядках з урахуванням згаданих вище полів сторінок і проміжків між елементами.

При реалізації візуальної схеми необхідно відокремити стилістичні аспекти для користувача інтерфейсу від його вмісту. Наприклад, якщо визначити схему тільки в одному місці, наприклад в таблиці стилів CSS або класі Java, то її можна буде міняти незалежно від вмісту, що спростує внесення поправок в дизайні.

- ***Center Stage (Центральна сцена).***

Помістіть найважливішу частину користувацького інтерфейсу в найбільший підрозділ сторінки або вікна; об'єднайте додаткові інструменти і вміст на невеликих панелях, що оточують сцену (рис. 2.9)

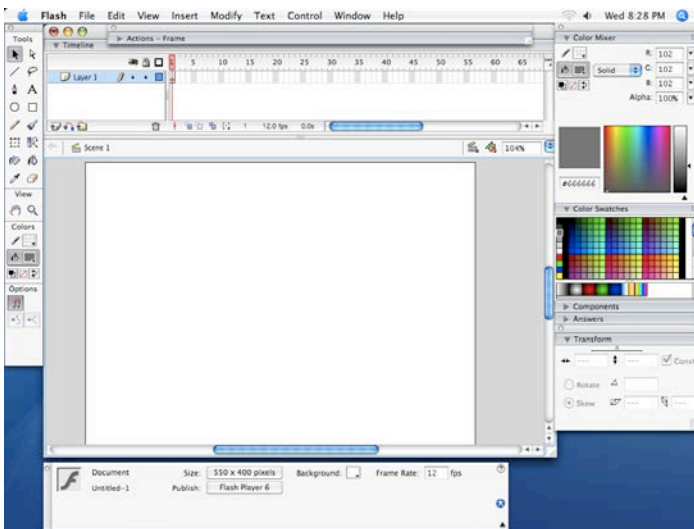


Рисунок 2.9 – Приклад шаблону «центральна сцена»

Даний шаблон використовується для демонстрації користувачеві логічно пов'язаного вмісту, дозволяє йому відредагувати документ або виконати певне завдання. «Центральну сцену» можна використовувати в більшості програм – сюди входять текстові таблиці, форми, веб-сторінки і графічні редактори.

Погляд користувача необхідно відразу ж привернути до початку найбільш важливої інформації. Чітко виражений центральний елемент приковує увагу.

Рекомендації. Створіть візуальну ієрархію, в якій «центральна сцена» буде домінувати над усім іншим. При розробці інтерфейсу враховуйте наступні чинники:

Розмір. Вміст «центральної сцени» має бути як мінімум в два рази ширше за все, що знаходиться в лівому і правому полях, і в два рази вище того, що займає верхнє і нижнє поля.

Колір. Використовуйте колір, що контрастує з інформацією, що знаходиться на полях.

Заголовки. Великі заголовки є точками фокусування і можуть привертати увагу користувача до верхньої частини «центральної сцени».

Контекст. Що користувач очікує побачити, коли він відкриває сторінку: графічний редактор, довгу текстову статтю, карту або дерево файлової системи? Скористайтеся тим, що знаєте, чого він чекає – помістіть цей об'єкт на центральну сцену і зробіть його легко впізнаваним.

Наступна група шаблонів представляє чотири альтернативні способи розбиття вмісту сторінки або вікна на частини. Їх застосовують, коли на сторінці одночасно необхідно показати дуже багато інформації. Після того як прийнято рішення щодо розбиття вмісту на тематичні розділи, слід вирішити, як представляти їх користувачеві. Чи повинні вони всі бути присутніми на екрані одночасно, або їх можна переглядати незалежно? Дозволити користувачеві будь-яким чином маніпулювати розділами на сторінці? Ці шаблони відносяться до розробки візуальної ієрархії, але оскільки в них задіяна інтерактивність, вони можуть допомогти вам зробити вибір серед специфічних механізмів, що надаються засобами розробки користувацьких інтерфейсів.

• ***Titled Section (Іменовані розділи).***

Розбийте вміст на автономні розділи, присвоївши кожному візуально помітний заголовок, а потім перелічте всі розділи на одній сторінці (див. рис. 2.10).

STANDART PLAN	SILVER PLAN	BUSINESS PLAN	PLATINUM PLAN
<ul style="list-style-type: none"> - 100MB Disc Space - 3GB Transfer - 30 Email Accounts - CGI, PHP, Frontpage - BackUP storage - Account Control Panel 	<ul style="list-style-type: none"> - 100MB Disc Space - 3GB Transfer - 30 Email Accounts - CGI, PHP, Frontpage - BackUP storage - Account Control Panel 	<ul style="list-style-type: none"> - 100MB Disc Space - 3GB Transfer - 30 Email Accounts - CGI, PHP, Frontpage - BackUP storage - Account Control Panel 	<ul style="list-style-type: none"> - 100MB Disc Space - 3GB Transfer - 30 Email Accounts - CGI, PHP, Frontpage - BackUP storage - Account Control Panel
order now \$ 19. ⁹⁵	order now \$ 29. ⁹⁵	order now \$ 59. ⁹⁵	order now \$ 99. ⁹⁵

Рисунок 2.10 – Шаблон «іменовані розділи»

Даний шаблон використовується, коли на сторінці дуже багато вмісту, але ви хочете зробити її максимально зручною для швидкого перегляду та сприйняття.

За допомогою добре визначених і іменованих розділів вміст ділиться на порції, кожна з яких зрозуміла користувачеві з одного погляду. Йому набагато комфортніше вивчати сторінку, акуратно розбиту на подібні розділи.

Рекомендації. Спочатку добре продумайте інформаційну архітектуру – розбийте вміст на зв'язкові фрагменти і надайте їм короткі назви що добре запам'ятовуються. Потім виберіть спосіб подання:

- для заголовків використовуйте шрифт, що виділяється на фоні решти вмісту – більш жирний, більшого розміру, більш насиченого кольору і т.д.

- спробуйте помістити заголовок на смугу контрастного кольору.

- використовуйте порожній простір для відділення розділів один від одного.

• ***Card Stack (Пачка карток).***

Помістіть розділи вмісту на окремі панелі, або «картки», а потім зберіть їх в пачку, щоб завжди було видно тільки одна. Використовуйте вкладки, або інші засоби, щоб надати користувачеві доступ до всіх карток (див. рис. 2.11).

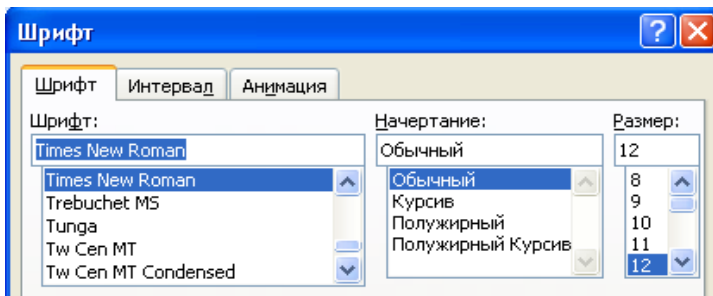


Рисунок 2.11 – Приклад шаблону «пачка карток»

Даний шаблон використовується, коли на сторінці дуже багато матеріалу. Безліч елементів управління або блоків тексту розподілені по всьому інтерфейсу і їх неможливо організувати в жорстку структуру: увага користувача розсіюється.

Структура з іменованими «картками» дозволяє розбити вміст на зрозумілі розділи, уявлення про кожний з яких можна буде скласти з одного погляду.

Рекомендації. Розбийте вміст на взаємопов'язані частини і надайте їм короткі заголовки що добре запам'ятовуються. Потім виберіть спосіб подання:

- вкладки - зазвичай зручні там, де потрібно не більше шести карток. Не укладайте картки так, щоб їх заголовки виводилися в два рядки, тому що цей варіант складно використовувати;
- вертикальні вкладки дозволяють вмістити пачку карток у вузький і довгий простір, куди не поміщаються звичайні вкладки;
- стовпець назв в лівій частині сторінки чудово працює на багатьох веб-сторінках і в діалогових вікнах. У такій стовпець можна помістити безліч карток, що неможливо з вкладками.

• ***Closable Panels (Панелі, що закриваються).***

Помістіть кожний розділ вмісту на окрему панель і дозвольте користувачеві відкривати і закривати їх незалежно один від одного (див. рис.2.12).

Abstract1	(15)
Abstract2	(15)
Abstract3	(15)
Abstract4	(15)
Autos	(31)
Cities	(7)
Clouds	(15)
Computers	(15)
Financial	(11)
<ul style="list-style-type: none"> calculator money1 money2 money3 money4 money5 money6 money7 money8 wallstreet worldbank 	
Flowers	(15)
Golf	(20)
Mountains	(20)

Рисунок 2.12 – Приклад шаблону «Панелі, що закриваються»

Даний шаблон використовується при великому обсязі вмісту, щоб одночасно вивести його на сторінці, але необхідно забезпечити його вивід одним клацанням миші. Вміст ділиться на фрагменти, що зрозуміло іменовані. Користувач може бачити одночасно від двох і більше розділів. Цей шаблон можна використовувати замість шаблону «Пачка карток», проте «Панелі, що закриваються» дають розробнику більше свободи:

- розміри розділів можуть варіюватися в широкому діапазоні;
- користувач може одночасно відкривати кілька розділів;
- якщо створити тільки одну панель, що закривається і помістити її на сторінку або в діалогове вікно великого розміру, то буде створюватися враження, що панель не так важлива, як вміст навколо неї. Якщо ж обидва розділи помістити на дві вкладки, то вони будуть здаватися однаково значимими.

Рекомендації. Розбийте вміст на розділи, надайте їм зрозумілі імена і помістіть їх на панелі, які будуть відкриватися і закриватися по клацанню на кнопки або посиланні. У текст на кнопки або посиланні додайте стрілку, значок «плюс», трикутник - так ви підкажете користувачеві, що тут він може щось відкрити або закрити. У багатьох

програмах трикутник вказує вниз на відкриту панель, коли вона відкрита, і вправо, коли вона закрита.

• **Movable Panels (Панелі, що переміщуються).**

Помістіть різні інструменти, або розділи вмісту на окремі панелі і дозвольте користувачеві переміщати їх по інтерфейсу, створюючи власне компонування (див. рис. 2.13).

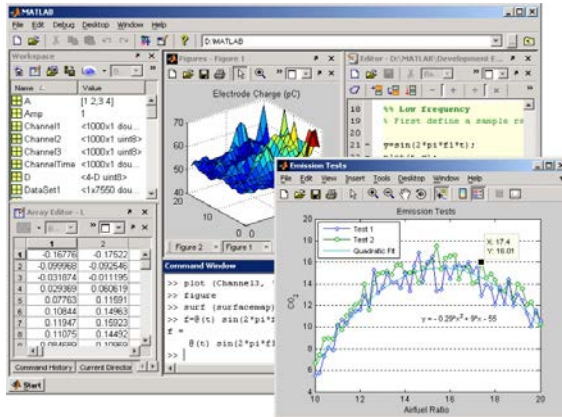


Рисунок 2.13 – Приклад шаблону «Панелі, що переміщуються»

Даний шаблон використовується, коли на сторінці є кілька пов'язаних «фрагментів» інтерфейсу, які не обов'язково повинні знаходитися в одній загальній конфігурації.

Коли людям доводиться довго над чим-небудь працювати в них виникає бажання реорганізувати середовище для оптимальної відповідності їх стилю роботи. Вони можуть помістити більш необхідні інструменти ближче до місця роботи, приховати непотрібні речі.

Рекомендації. Дозвольте користувачеві переміщати фрагменти інтерфейсу на сторінці. У залежності від обраного дизайну можна дати йому свободу поміщати фрагменти в будь-яке місце інтерфейсу, навіть якщо вони будуть перекривати один одного.

Наступні два шаблони ґрунтуються на концепціях візуального потоку, вирівнювання та інших принципах. Вони стосуються просторових взаємовідносин між більш дрібними і статичними елементами на сторінці, такими як текст і елементи керування.

• ***Right / Left Alignment (Вирівнювання по правому / лівому краю).***

При розробці форми або таблиці, що складається з двох стовпців, вирівняйте мітки, які стоять ліворуч по правій стороні, і елементи, що стоять праворуч - по лівій стороні (див. рис. 2.14).

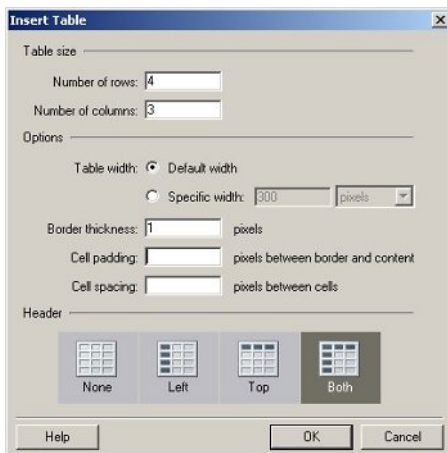


Рисунок 2.14 – Приклад шаблону «Вирівнювання по правому/лівому краю»

Даний шаблон використовується при компоунванні форми або будь-якого іншого набору елементів, перед якими будуть знаходитися текстові мітки. Його також можна застосовувати до внутрішньої структури таблиць.

Коли текст поміщається впритул до елемента, для якого він служить міткою, формується сувора перцепційна пара - набагато більш явна, ніж, якби ці два елементи відділялися один від одного великим простором. Якщо вирівняти мітки різної довжини по лівій стороні, то короткі мітки виявляться недостатньо близько до відповідних елементів управління і поперечне угруповання буде порушене. Елементи управління слід завжди вирівнювати по лівій стороні.

Рекомендації. Замість того, щоб вирівнювати текстові мітки по лівій стороні, вирівняйте їх по правій. Вони повинні знаходитися

впритул до відповідних елементів управління, і між ними повинна залишатися лише кілька пікселів.

• ***Diagonal Balance (Діагональний баланс).***

Організуйте елементи на сторінці асиметрично, але збалансуйте їх, помістивши візуальну вагу у верхній лівий і нижній правий кути (див. рис.2.15).

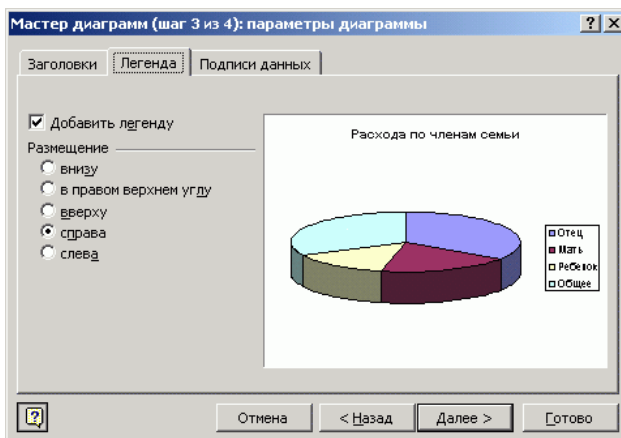


Рисунок 2.15 – Пример шаблона «Діагональний баланс»

Даний шаблон використовується при створенні сторінки, або діалогового вікна, де вгорі знаходиться заголовок, або колонтитул, а внизу – кілька посилань, або кнопок, що включають дії, наприклад ОК або CANCEL. Сторінка досить мала, щоб весь вміст містився на екрані, і прокручувати його немає необхідності.

Елементи сторінки, що візуально виділяються, такі як заголовки, вкладки чи кнопки, повинні сприяти балансуванню композиції на екрані. Діагональний баланс прийнятний для погляду, він допомагає очам користувача комфортно переміщатися зліва направо і зверху вниз.

Рекомендації. Помістіть заголовок, вкладки або який-небудь інший сильний елемент у верхній лівий кут сторінки, а кнопки - в нижній правий. Вміст залишається в середині.

Наступні три шаблони визначають динамічні аспекти компоновання вмісту. Responsive Disclosure (Виявлення у відповідь)

[1,с.184] і Responsive Enabling (Включення у відповідь) [1,с.187] представляють собою два способи спрямування дій користувача у відповідності з послідовністю кроків або набором варіантів. Вони вказують, що він може зробити в кожен момент часу, в той же час забороняючи користувачеві переходити в інше місце. Liquid Layout («Гумовий» макет) [1,с.190] - це техніка організації сторінки, що дозволяє міняти розмір і форму сторінки в залежності від бажань користувача.

• ***Responsive Disclosure (Виявлення у відповідь)***

Почніть з самого лаконічного варіанту користувача інтерфейсу, і проведіть його по послідовності кроків, показуючи йому більше елементів інтерфейсу після завершення кожного кроку.

Даний шаблон використовується, коли інтерфейс повинен допомогти користувачеві виконати складне завдання крок за кроком, а ви не хочете, щоб він для виконання кожного кроку переходив на нову сторінку - краще залишити весь інтерфейс на одній сторінці.

У цьому шаблоні інтерфейс як-би «створюється» на очах у користувача. Спочатку користувач бачить тільки елементи, необхідні для завершення першого кроку. Коли він переходить до наступного кроку, відображається черговий набір елементів на додаток до першого, потім ще один і т.д. Оскільки весь призначений для користувача інтерфейс зосереджений на одній сторінці, користувач може з легкістю повернутися назад і змінити що-небудь в елементах, що були раніше налаштовані. Як тільки виправляється щось на одному кроці, користувач бачить ефект, що впливає на наступні кроки.

Рекомендації. Почніть з відображення елементів управління і тексту, необхідних для першого кроку. Коли користувач завершить його, покажіть елементи керування для другого. Залишайте на очах складові попередніх кроків, дозволяючи користувачеві при необхідності повертатися до них. Нехай все необхідне знаходиться на одній сторінці або в одному діалоговому вікні, щоб користувачеві не доводилося несподівано перескакувати в новий простір користувацького інтерфейсу.

• ***Responsive Enabling (Включення у відповідь)***

Почніть з інтерфейсу, де відключено більшість елементів, і проведіть користувача по послідовності кроків, включаючи все більше і більше елементів після завершення кожного кроку.

Даний шаблон використовується, коли інтерфейс веде користувача по складній задачі крок за кроком, тому що користувач не дуже добре знайомий з комп'ютерами або ця задача виконується рідко, а ви не хотіли б, щоб він на кожному кроці переходив на нову сторінку.

Цей шаблон використовує переваги динамічності комп'ютерних дисплеїв і дозволяє користувачеві виконувати задачі в інтерактивному режимі. Таким чином, у користувача формується правильна ментальна модель, яка об'єднує причини і наслідки. Користувальницький інтерфейс підказує можливі наслідки вибору, наприклад: якщо встановити цей прапорець, то доведеться заповнювати ці чотири текстових поля, які щойно стали активними.

Рекомендації. У деяких додатках більшість дій спочатку відключені - доступні тільки ті, які відносяться до першого кроку користувача. У міру того як користувач робить свій вибір і виконує дії, повинні включатися додаткові елементи. Коли можливо, розмішуйте відключені елементи впритул до того, що їх включає. Це допомагає користувачеві знаходити кнопки активації відповідних операцій і розуміти стосунки між двома елементами.

• ***Liquid Layout ("Гумовий" макет)***

Коли користувач змінює розміри вікна, змінюйте розмір вмісту сторінки, щоб вона завжди була «заповнена».

Даний шаблон використовується, коли користувачеві може знадобитися, щоб вміст робочої програми, діалогового вікна або сторінки відображався у більшому (або меншому) просторі. Це зазвичай відбувається, коли сторінка містить багато тексту або містить насичений інформацією елемент керування.

Якщо ви розробляєте не «закритий» користувальницький інтерфейс, неможливо передбачити умови, в яких користувач буде переглядати цей інтерфейс. Надаючи користувачеві певну частку контролю над компонованням сторінки, ви робите для користувача інтерфейс більш гнучким. Користувачеві більше сподобається інтерфейс, який він завжди може налаштувати у відповідності з поточними вимогами і актуальним контекстом.

Рекомендації. Зробіть так, щоб вміст сторінки завжди заповнювало вікно, навіть якщо розмір вікна змінюється. Текст, що складається з декількох рядків потрібно переносити у правого поля.

Весь вміст, що прокручується (списки, таблиці тощо), повинен збільшуватися й, частіше за все, також розширюватися.

Дані повинні розташовуватися так, щоб користувач міг:

- спостерігати за екраном в логічній послідовності;
- виводити потрібну інформацію;
- визначати пов'язані групи інформації;
- розрізняти виняткові ситуації (повідомлення про помилки або попередження);
- визначати, яка дія потрібна з його боку.

Хоча екран не повинен містити надлишкову інформацію, важливо, щоб відображалася вся інформація, що відноситься до завдання, що розв'язується в даний момент. Не слід примушувати користувача запам'ятовувати інформацію на одному етапі, щоб пізніше скористатися нею на іншому етапі.

2.3 Приклад розробки структури та макета екрану інтерфейсу

Після того, як ми з'ясували основні завдання, і вимоги користувачів, складемо структуру і макет інтерфейсу КА.

Так як візуальний потік визначає траєкторію переміщення погляду користувача при перегляді інтерфейсу, введемо вітання для наших покупців у верхній частині головного вікна (див. рис. 2.16 № 1). Далі необхідно, щоб користувач звернув увагу на список напоїв, що пропонується. Тому розмістимо цей список по лівій стороні, так як візуальний потік за замовчуванням визначає такий напрямок погляду – зверху вниз і зліва направо (див. рис. 2.16 № 2). Для відображення всього списку напоїв застосуємо шаблон Titled Section (Іменовані розділи) [1, с.158].

Інформаційне вікно виведення повідомлень системи розмістимо у верхній частині по правій стороні (див. рис. 2.16 № 3).

Для того, щоб користувач міг вибрати потрібну кількість цукру застосуємо елементи управління з функцією зменшення і збільшення і розмістимо їх знизу після інформаційного вікна (див. мал. 2.16 № 4). Елементи управління слід розмістити в нижній частині з вирівнюванням по лівій стороні (див. рис. 2.16 № 5). Для угруповання елементів управління будемо використовувати властивість "близькість".

Для внесення оплати за вибраний напій створимо вікно введення грошових коштів. Розмістимо його біля елемента управління, що володіє функцією внесення грошових коштів в КА (див. рис. 2.16 № 6).

Елемент управління, який здійснює вхід в системне меню розмістимо в нижній частині інтерфейсу з вирівнюванням по правій стороні (див. рис. 2.16 № 7).

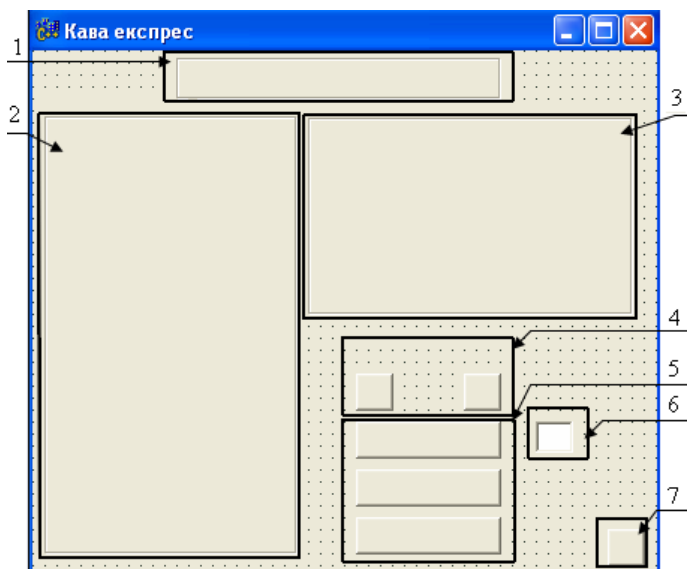


Рисунок 2.16 – Структура та макет інтерфейсу користувача КА

Виведемо повідомлення «Інформація про стан апарату» у верхній частині системного меню з вирівнюванням по центру (див. рис. 2.17 № 1). Розмістимо рисунок з вирівнюванням по лівій стороні (див. рис. 2.17 № 2).

Інформаційне вікно виведення повідомлень системи для наладчика про стан апарату розмістимо у верхній частині системного меню з вирівнюванням по лівій стороні (див. рис. 2.17 № 3). Елементи для керування поповненням ресурсів КА розмістимо в нижній частині інформаційного вікна (див. рис. 2.17 № 4).

Для перегляду статистики з продажу напоїв застосуємо шаблон Card Stack (Пачка карток) і розмістимо його в нижній частині системного меню (див. рис. 2.17 № 5).

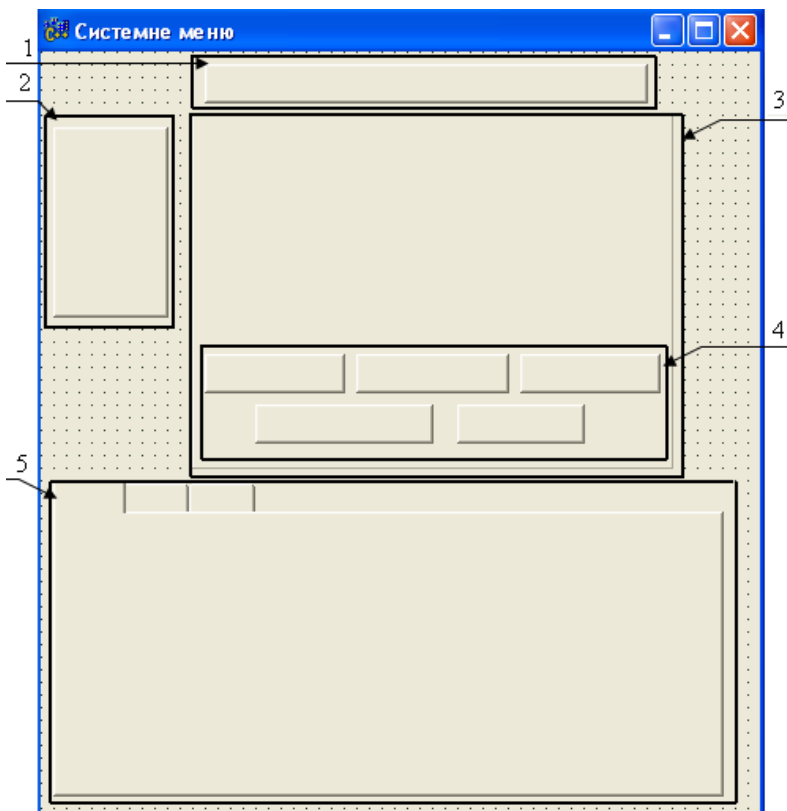


Рисунок 2.17 – Структура та макет системного меню КА

2.4 Завдання до роботи

2.4.1 Ознайомитися з пунктом 2.2 і конспектом лекцій.

2.4.2 Для обраного лабораторній роботі №1 об'єкта продумати структуру, форму і макет екрану інтерфейсу.

2.4.3 Програмно реалізувати інтерфейс.

2.4.4 Оформити звіт по роботі.

2.4.5 Відповісти на контрольні питання.

2.5 Зміст звіту

2.5.1 Тема і мета роботи.

2.5.2 Завдання до роботи.

2.5.3 Текст програми.

2.5.4 Результати роботи програми.

2.5.5 Висновки за результатами виконання роботи.

2.6 Контрольні запитання

2.6.1 З яких кроків складається проектування користувацького інтерфейсу?

2.6.2 Які дії необхідно виконати для визначення цілей та операцій інтерфейсу?

2.6.3 У чому суть концепції візуальної ієрархії?

2.6.4 Що визначає візуальний потік?

2.6.5 Що таке точки фокусування?

2.6.6 Як здійснюється угруповання і вирівнювання елементів інтерфейсу?

2.6.7 Які властивості характеризують компонування елементів?

2.6.8 Що таке компонування сторінки?

2.6.9 Які шаблони для компонування сторінок ви знаєте?

2.6.10 В яких випадках використовується шаблон «Візуальна сцена»?

2.6.11 Які шаблони рекомендовано використовувати, якщо на сторінці присутньо дуже багато інформації?

2.6.12 Який шаблон рекомендовано використовувати для набору елементів, перед якими будуть знаходитися текстові мітки?

3 ЛАБОРАТОРНА РОБОТА № 3

Етапи проектування користувацького інтерфейсу. Вибір елементів керування. Текст і числа.

3.1 Мета роботи

Освоїти третій етап проектування користувацького інтерфейсу, в ході якого необхідно вивчити основи дизайну форм і елементи керування.

3.2 Основні теоретичні відомості

3.2.1 Основи дизайну форм

Взаємодія користувача з програмним інтерфейсом часто припускає використання різних елементів керування (текстових полів, прапорців і т.д.). Для запобігання помилок при побудові графічних інтерфейсів перелічимо і розглянемо принципи, яких необхідно дотримуватися при створенні форм і засобів введення даних користувачем:

Користувач повинен розуміти, які дані в нього запитують і навіщо.

Рекомендації:

- у випадку великих (довгих) форм відведіть деяку частину простору на пояснення користувачеві, навіщо потрібна інформація, яка запитується у формі;
- якщо на панелі інструментів розташовується елемент керування, призначення якого неочевидно, додайте для нього описову спливаючу підказку;
- при написанні міток використовуйте слова, які повинні бути зрозумілими для цільової аудиторії – проста мова для новачків і спеціалізовані терміни для експертів.

Кількість питань повинна бути мінімальною.

Задаючи користувачеві питання, особливо якщо це відбувається в процесі виконання якої-небудь іншої задачі, програма чине на нього тиск, змушує перервати потік роздумів і звернути увагу на те, чого він зовсім не очікував.

Рекомендації:

- якщо можливо, заздалегідь заповніть елементи керування введенням відомою чи передбачуваною інформацією (значенням за замовчуванням).

Згідно з концепцією Д. Нормана – знання «із зовнішнього світу» найчастіше точніше, ніж знання «в голові» (якими людина користується автоматично).

Рекомендації:

- користувачу важко запам'ятовувати різні списки і перерахування. Тому, при необхідності вибору з обумовленого набору елементів, цей набір або список йому необхідно надати.

Не слід буквально відтворювати програмну модель, яка складає основу програми.

Дизайн, що буквально відображає структуру даних (наприклад, для редагування записів бази даних) може вийти складним або нецікавим.

Рекомендації:

- зробити форму менш обтяжливою, намагаючись врахувати залежності між елементами структури, знайомі графічні конструкції, не сформульовані припущення або відомості про користувача, отримані при попередньому взаємодії з ним;

- спростіть завдання так, щоб його можна було вирішити прямою маніпуляцією, наприклад, перетягуванням об'єктів по екрану.

Вибір елементів керування впливає на те, які питання буде очікувати користувач.

Наприклад, перемикач передбачає вибір одного елемента з групи, а текстове поле з одного рядка – коротку, але досить вільну за формою відповідь.

Рекомендації:

- реакції користувачів при інтерактивній взаємодії з інтерфейсом (коли розмірковують про те, що у них запитують) формуються на основі фізичної форми елемента керування – його типу, розміру і т. д.

3.2.2 Вибір елементів керування

Далі описані основні елементи керування та шаблони для запити у користувачів різних типів інформації. При виборі елементів

керування для кожного типу інформації треба ґрунтуватися на наступних факторах:

Доступний простір.

Одні елементи керування займають багато екранного простору, інші можуть бути менші за розміром, але складніші у використанні, ніж більш великі.

Рекомендації:

- у коротких формах можна використовувати екранний простір на перемикачі або ілюстровані списки;
- у складних застосуваннях бажано упакувати як можна більше вмісту в якомога менший простір.

Досвід користувача в предметній області.

Текстове поле представляє собою хороший вибір елемента керування, якщо всі користувачі знають, що припустимо, в даному випадку можливі тільки значення від 1 до 10 і від 20 до 30.

Очікування, сформовані іншими програмами.

Виділення тексту жирним шрифтом, підкресленням шрифтом чи курсивом, як правило, позначено на кнопках з відповідними піктограмами, а не, наприклад, перемикачами.

3.2.2.1 Списки елементів

Вибір елемента керування залежить від того, скільки елементів або варіантів дозволяється обирати користувачеві (один або багато), а також від числа елементів у списку (два, декілька або багато).

Елементи керування для вибору одного з двох варіантів (бінарний вибір)

• ***Прапорець***

Переваги: простий, займає мало місця.

Недоліки: недвозначно визначає тільки один варіант, тому його протилежність залишається неявною і не сформульованою.

• ***Два перемикача***

Переваги: обидва варіанти сформульовані і видні на екрані.

Недоліки: займають більше місця.

• ***Список, що розкривається з двома варіантами вибору***

Переваги: обидва варіанти сформульовані; невеликий і передбачуваний об'єм займаного простору; просте розширення при необхідності додати більш ніж двох варіантів.

Недоліки: одночасно на екрані видно тільки один варіант.

- **Кнопка-перемикач с «залипанням»**

Переваги: ті ж, що і у прапорця.

Недоліки: ті ж, що у прапорця. Крім того, вона не сприймається в якості стандартного елемента керування для вибору текстових значень.

- **Меню з двома елементами, що представляють собою перемикачі**

Переваги: займає дуже мало місця, тому що знаходиться в смузі меню або спливаючому меню.

Недоліки: спливаючі меню іноді важко виявити; вимагає великої вправності.

Елементи керування для вибору одного з N елементів, коли число N невелике

- **N перемикачів**

Переваги: всі варіанти вибору завжди видно на екрані.

Недоліки: займає багато місця.

- **Список, що розкривається з N елементів**

Переваги: займає мало місця

Недоліки: одночасно видно тільки один варіант вибору, за винятком ситуації, коли список відкритий.

- **Набір з N взаємовиключних піктографічних кнопок-перемикачів**

Переваги: займає мало місця; видно всі варіанти вибору.

Недоліки: значки можуть бути складними для розуміння і вимагати спливаючих підказок, користувачі можуть не усвідомлювати, що варіанти вибору взаємовиключні.

- **Меню з N елементів, що представляють собою перемикачі**

Переваги: займає мало місця в головному інтерфейсі; видно всі варіанти вибору.

Недоліки: спливаючі меню буває важко виявити; вимагає великої вправності.

- **Список або таблиця, що допускають єдиний вибір**

Переваги: видно кілька варіантів вибору; рамку можна зробити невеликою, залишивши на увазі тільки три елементи.

Недоліки: займає більше місця, ніж список, що розкривається або лічильник.

- **Лічильник**

Переваги: займає мало місця.

Недоліки: одночасно видно тільки один варіант вибору; вимагає великої вправності; незнайомий непідготовленим користувачам. Звичайно краще використовувати список, що розкривається.

Елементи керування для вибору одного з N елементів, коли N велике

- ***Список, що розкривається з N елементів, при необхідності прокручування***

Переваги: займає мало місця.

Недоліки: одночасно видно тільки один варіант вибору, за винятком ситуації, коли список розкрито; вимагає великої вправності для прокручування елементів у списку, що розкривається.

- ***Список або таблиця, що допускають єдиний вибір***

Переваги: видно кілька варіантів вибору, при необхідності можна зробити зовсім невеликим.

Недоліки: займає більше місця, ніж список, що розкривається.

- ***Допускає єдиний вибір дерево або каскадний список з елементами, розбитими на категорії***

Переваги: видно кілька варіантів вибору; в певних випадках організація даних полегшує пошук елементів.

Недоліки: можуть бути незнайомі непідготовленим користувачам; займають багато місця; вимагають великої вправності.

Елементи керування для вибору декількох з N елементів у будь-якому порядку

- ***Масив з N прапорців***

Переваги: всі варіанти сформульовані і видні на екрані.

Недоліки: займає багато місця.

- ***Масив з N кнопок, що «залипають»***

Переваги: займає мало місця; всі варіанти вибору видно на екрані.

Недоліки: значки можуть бути складними для розуміння і вимагати спливаючих підказок; кнопки можуть здатися взаємовиключними.

- ***Меню з N елементів, що представляють собою прапорці***

Переваги: займає мало місця в головному інтерфейсі; видно всі варіанти вибору.

Недоліки: спливаючі меню буває важко виявити; вимагає великої вправності.

- ***Список або таблиця, що допускають множинний вибір***

Переваги: на екрані видно кілька варіантів вибору, а при необхідності можна зробити цей елемент дуже невеликим.

Недоліки: не всі варіанти вибору видно без прокрутки; займає багато місця, користувачі можуть не розуміти, що список допускає множинний вибір.

- ***Список елементів, що представляють собою прапорці***

Переваги: на екрані видно кілька варіантів вибору, а при необхідності цей елемент керування можна зробити зовсім невеликим; засіб вибору (прапорець) представлено очевидним чином.

Недоліки: не всі варіанти вибору видно без прокрутки; займає багато місця.

- ***Дерево, що допускає множинний вибір або каскадний список, елементи якого розбиті на категорії***

Переваги: на екрані видно кілька варіантів вибору; в певних випадках організація даних полегшує пошук елементів.

Недоліки: можуть бути незнайомі непідготовленим користувачам; вимагають великої вправності; виглядають так само, як дерево або список з єдиним вибором.

- ***Користувальницький варіант браузера, наприклад, для пошуку файлів, кольорів або шрифтів***

Переваги: добре підходить для перегляду доступних варіантів.

Недоліки: може бути незнайомий деяким користувачам; складний у розробці дизайну; найчастіше представляє собою окреме вікно, тому його швидкодія звичайно нижча, ніж у елементів керування, що знаходяться прямо на сторінці.

3.2.2.2 Текст

Елементи керування зазвичай вибираються залежно від числа рядків в тексті, що вводиться, від того, чи визначаються рядки заздалегідь, чи може користувач вводити довільний текст, і чи містить текст форматування.

Елементи керування для введення одного рядка тексту

- ***Текстове поле з одного рядка***

Елементи керування для введення одного рядка тексту або вибору одного з N варіантів

• **Комбіноване поле**

Переваги: реагує на дії користувача швидше, ніж окреме діалогове вікно; відомо всім користувачам.

Недоліки: щоб зберегти розумну довжину списку, що розкривається, доводиться обмежувати число елементів.

• **Текстове поле з кнопкою Додатково (також можна використовувати її з комбінованим полем, а не текстовим)**

Переваги: дозволяє відкривати спеціалізоване діалогове вікно для вибору елемента, наприклад утиліту пошуку файлів.

Недоліки: менш знайоме деяким користувачам, ніж комбіноване поле; повільніше у використанні.

Елементи керування для введення декількох рядків неформатованого тексту

• **Текстова область шириною в декілька рядків**

Елементи керування для введення декількох рядків форматованого тексту

• **Текстова область з вбудованими тегами**

Переваги: досвідчені користувачі можуть не вдаватися за допомогою до панелі інструментів, безпосередньо вводячи теги.

Недоліки: не відноситься до класу редакторів WYSIWYG (what you see is what you get - що бачиш, те й отримувеш).

• **Вдосконалений текстовий редактор**

Переваги: миттєва реакція, тому що текст, що редагується грає роль вмісту для попереднього перегляду.

Недоліки: необхідно використовувати панель інструментів, тому робота тільки за допомогою клавіатури неможлива.

3.2.2.3 Числа

Елементи керування для введення чисел будь-якого типу або формату

• **Текстове поле з використанням шаблону Forgiving Format (Великодушний формат).**

617 555 -1212

Переваги: візуально елегантне рішення; допускає безліч форматів і типів даних.

Недоліки: по вигляду елемента керування не можна зробити однозначний висновок про очікуваний формат; вимагає уважної перевірки введених даних у програмі.

• **Текстове поле з використанням шаблону *Structured Format***

(Структурований формат).

6125 - 3185

Переваги: бажаний формат однозначно визначається, виходячи з форми елемента керування.

Недоліки: може займати більше місця; велика візуальна складність; не дозволяє відхилятися від певного формату.

• **Лічильник (оптимальний для цілих чисел або дискретних значень)**

Переваги: користувач може відкрити потрібне значення, користуючись покажчиком миші без використання клавіатури (при необхідності можливий ввід з клавіатури).

Недоліки: незнайомий деяким користувачам; для того щоб відкрити потрібне значення, може знадобитися досить довго утримувати кнопку.

Елементи керування для введення чисел з обмеженого діапазону

• **Повзунок**

Переваги: візуально демонструється позиція значення в діапазоні; користувач не може ввести число за межами діапазону.

Недоліки: займає багато місця, неочевидний спосіб доступу з клавіатури; позначки значень можуть захащувати представлення.

• **Лічильник**

Переваги: коли використовуються кнопки, значення обмежуються певним діапазоном; займає мало місця; можливе введення як з клавіатури, так і за допомогою миші.

Недоліки: знайомий не всім користувачам; необхідна перевірка; неможливо візуально визначити положення значення в діапазоні.

• **Текстове поле з перевіркою помилок за фактом**

Переваги: знайоме всім користувачам; займає мало місця; можливий доступ тільки з клавіатури.

Недоліки: необхідна перевірка; ніяких обмежень на значення, що вводиться, не накладається.

• **Повзунок з текстовим полем**

Переваги: допускає візуальне і числове введення.

Недоліки: складний; коли обидва елементи знаходяться на сторінці, займає багато місця; вимагає перевірки текстового поля, якщо користувач вводить значення з клавіатури.

Елементи керування для введення частини більш великого діапазону

• **Подвійний повзунок**

Переваги: займає менше місця, ніж два повзунки.

Недоліки: незнайомий більшості користувачів; відсутній доступ з клавіатури, якщо тільки не використовуються текстові поля.

• **Два повзунки**

Переваги: виглядають витонченішими, ніж подвійний повзунок.

Недоліки: займають багато місця; відсутній доступ з клавіатури, якщо тільки не використовуються текстові поля.

• **Два лічильники**

Переваги: коли використовуються кнопки, діапазони значень обмежені; займають мало місця; підтримують введення як з клавіатури, так і за допомогою миші.

Недоліки: незнайомий деяким користувачам; необхідна перевірка; відсутнє візуальне представлення позиції значення в діапазоні.

• **Два текстових поля з перевіркою помилок**

Переваги: знайомі всім користувачам; займають набагато менше місця, ніж повзунки.

Недоліки: необхідна перевірка; ніяких обмежень на значення, що вводяться, не накладається.

3.2.2.4 Дата і час

Варіанти введення дати і час наступні:

• **Текстове поле з «великодушним форматом» (див. п.3.2.2.3)**

Переваги: візуально просте; допускає великий діапазон форматів і типів даних; доступ з клавіатури.

Недоліки: за зовнішнім виглядом елемента керування не можна точно визначити очікуваний формат даних; вимагає перевірки за фактом введення.

• **Текстове поле з структурованим форматом (див. п.3.2.2.3)**

Переваги: очікуваний формат легко визначити, виходячи з зовнішнього вигляду елемента керування.

Недоліки: може займати більше місця; велике візуальне навантаження.

• **Елемент керування у формі календаря або годин**

Переваги: введення обмежується тільки допустимими значеннями.

Недоліки: займає багато місця, може не підтримувати доступ з клавіатури.

• **Селектор, що розкривається, з елементом керування у формі календаря або годин**

Переваги: поєднує переваги текстового поля і календаря; займає мало місця.

Недоліки: складна взаємодія; вимагає вправності для вибору значень у розкривної частини.

3.2.3 Приклад розробки інтерфейсу користувача з елементами керування

Після розробки структури нашої форми та макета екрану інтерфейсу кавового автомата, необхідно визначити перелік елементів керування, необхідних для успішного виконання поставлених завдань.

Головна форма. Створимо привітання для наших користувачів (покупців). З стандартної панелі інструментів (Standard) вибираємо мітку (Label) у властивостях (properties) у рядку заголовків (caption) записуємо привітання «Ласкаво просимо!» (див. рис. 3.1 № 1). Для всіх інших повідомлень також використовуємо мітки (Label). Для асортименту напоїв застосуємо елемент радіокнопки (RadioGroup) зі стандартної панелі інструментів. Радіокнопки утворюють групу взаємопов'язаних індикаторів, з яких може бути вибраний тільки один варіант. У властивостях в рядку Items типу (Tstrings) перераховуємо весь перелік напоїв (див. рис. 3.1 № 2).

Після вибору напою виводимо повідомлення «внесіть гроші в продовж ...» і включаємо таймер на 10 секунд для внесення оплати за напій:

```
void __fastcall TForm1::RadioGroup1Click(TObject *Sender)
{
    if (RadioGroup1->ItemIndex >= 0) //якщо індекс дорівнює 0 або
        більше (тобто якщо обраний будь-який напій із запропонованого
        списку)
```

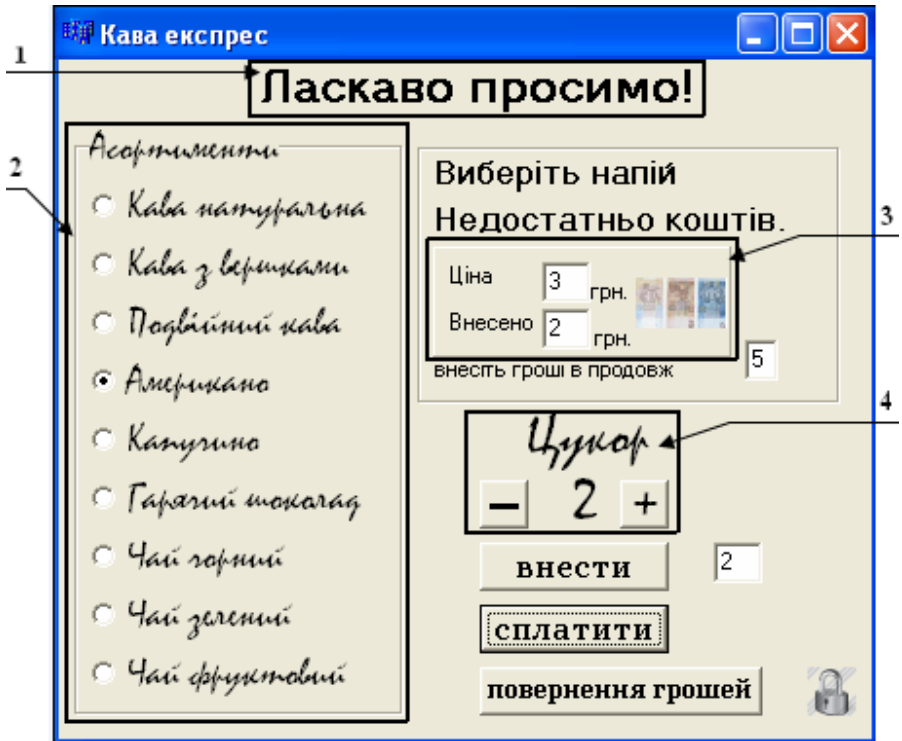


Рисунок 3.1 – Виведення повідомлення "Недостатньо коштів" при внесенні суми меншої за необхідну

```

{
    Timer2->Interval = 10000; //включаємо таймер на 10 секунд
    Label12->Caption = "внесіть гроші в продовж"; //в мітку Label2
    вивести дане повідомлення
    if (RadioGroup1->ItemIndex == 2) //обраний другий напій
    {
        Edit3->Text = "4"; //в елемент Edit3 вивести 4 (тобто ціна
        дорівнює 4 грн.)
    }
    else
    {

```

```

        if (RadioGroup1->ItemIndex == 4)    //обраний четвертий
напій
    {
        Edit3->Text = "4";
    }
    else
    {
        if (RadioGroup1->ItemIndex == 6)    //обраний шостий
напій
    {
        Edit3->Text = "4";
    }
    else
    {
        Edit3->Text = "3";    //в елемент Edit3 вивести 3 (тобто
ціна всіх інших напоїв дорівнює 3 грн.)
    }
    }
    }
    int tim;    // змінна цілого типу (лічильник)
    for (tim=10; tim>0; tim--)    //лічильник початкове значення
якого дорівнює 10, зменшується на 1, поки не буде дорівнює 0
    {
        Timer3->Interval = 1000;    //інтервал таймера 1 секунда
    }
    } }

```

Якщо протягом 10 секунд оплата не проведена, автомат переходить в початковий стан і виводить повідомлення «Скасування замовлення»:

```

void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
    Timer2->Interval = 0;    //коли таймер дорівнює 0
    Label2->Caption = "";    //робимо мітку Label2 порожньою (тобто
прибираємо повідомлення «Виберіть напій»)
    Label8->Caption = "Скасування замовлення";    //в мітку
Label8 записуємо це повідомлення (тобто замість «Внесіть гроші»,
з'являється «Скасування замовлення»)
    Timer4->Interval = 4000;    //включаємо таймер на 4 секунди

```



```

Timer2->Interval = 0;    //після 4 секунд переходимо в
початковий стан
    }

```

Створимо панель (Panel) у груповому вікні GroupBox для відображення вартості напою і внесених користувачем грошей (див. рис. 3.1 № 3).

Для визначення кількості цукру застосуємо дві керуючих кнопки (button): одна на зменшення, інша на збільшення (див. рис. 3.1 № 4):

```

void __fastcall TForm1::Button3Click(TObject *Sender)
{
    Sah--; //зменшення значення цукру на одиницю
    Label10->Caption = Sah; //в мітку Label10 записуємо значення
}
void __fastcall TForm1::Button4Click(TObject *Sender)
{
    Sah++; //збільшення значення цукру на одиницю
    Label10->Caption = Sah;
}

```

Для введення грошових коштів використовуємо поле Edit зі стандартної панелі інструментів. Для внесення грошових коштів в автомат застосуємо кнопку (button), в заголовку напишемо: «внести»:

```

void __fastcall TForm1::Button2Click(TObject *Sender)
{
    Edit2->Visible = false;    //значення елемента Edit2 рівно false
(тобто даний елемент не виводимо на головну форму)
    Label12->Visible = false;    //даний елемент не виводимо на
головну форму
    if (!(Edit1->Text == "")) //якщо елемент Edit1 не пусте значення
    {
        Edit4->Text = (Edit1->Text);    //в елемент Edit4 занести те
значення, яке міститься в Edit1 (тобто заносимо гроші в автомат)
        Timer2->Interval = 0;    //таймер вимкнений
        Timer3->Interval = 0;    //таймер вимкнений
    }
}

```

Для повернення внесених грошей використовуємо керуючу кнопку (button), в заголовку напишемо: «Повернення грошей»:

```
void __fastcall TForm1::Button5Click(TObject *Sender)
{
    Edit4->Text = "0"; //елемент Edit4 приймає значення 0 (тобто
    внесені в автомат гроші повертаються користувачеві)
}
```

Для оплати внесених грошей також використовуємо керуючу кнопку (button) стандартної панелі інструментів. При спробі оплатити напій сумою меншою за вартість напою, видати повідомлення "Недостатньо коштів» (див. рис. 3.1). Якщо внесена сума перевищує вартість напою, то видати здачу і видати повідомлення «Оплата зроблена. Приготування напою» (див. рис. 3.2):

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    Edit2->Visible = true; //значення елементу Edit2 равно true
    (тобто даний елемент, що відображає час, що залишився для внесення
    коштів, виводимо на головну форму)
    Label12->Visible = true; //даний елемент, що містить
    повідомлення про внесення грошей, виводимо на головну форму
    if (RadioGroup1->ItemIndex < 0) //якщо індекс менше 0 (тобто
    якщо не вибрано ні один напій)
    {
        Label6->Caption = "Ви не зробили вибір"; //вивести в Label6
        дане повідомлення
    }
    else
    {
        if ((Edit1->Text) < (Edit3->Text)) //якщо значення у Edit1
        менше, ніж у Edit3 (тобто якщо внесена сума менше вартості напою)
        Label8->Caption = "Недостатньо коштів."; //то вивести в
        Label8 дане повідомлення
        else
        {
            Panel1->Visible=false; //Panel1 приймає значення false
            (тобто панель, де відображається ціна і внесені гроші прибираємо з
            головної форми)
```

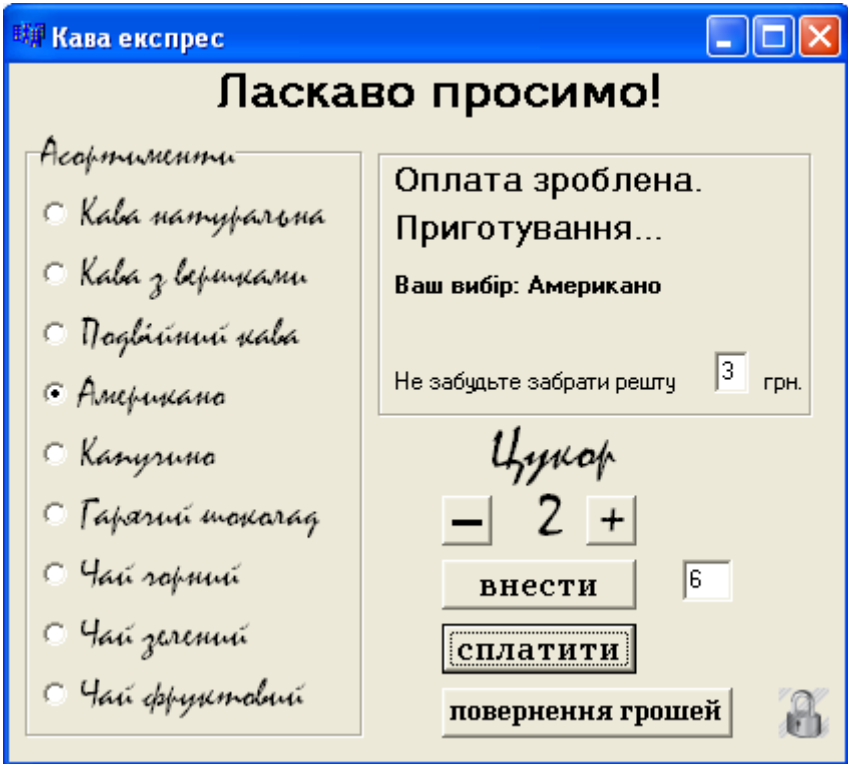


Рисунок 3.2 – Виведення повідомлення "Оплата зроблена. Приготування ..." при внесенні необхідної суми

Label8->Caption = "Приготування..."; //Замість «Внесіть гроші» виводимо "Приготування..."

Label2->Caption = "Оплата зроблена."; //замість «Виберіть напій» виводимо "Оплата зроблена"

Label6->Caption = "Ваш вибір: " + RadioGroup1->Items->Strings[RadioGroup1->ItemIndex] ; //виводимо в мітці Label6 повідомлення «Ваш вибір» і напис напою (тобто за індексом визначаємо вибраний напій і виводимо його)

Label12->Caption = "Не забудьте забрати решту"; //в мітку Label12 виводимо дане повідомлення

```

Label13->Visible = true;    //значення мітки Label13
дорівнює true (тобто виводимо запис «грн.»)
Edit2->Text = ((Edit4->Text)-(Edit3->Text));    //в елемент
Edit2 виводимо значення здачі
Timer1->Interval = 5000;    //включаємо таймер на 5
секунд (тобто час приготування напою 5 секунд)
    }
} }

```

Після приготування напою вивести повідомлення «Напій готовий. Будь ласка, заберіть » (див. рис. 3.3), після чого автомат переходить в початковий стан:

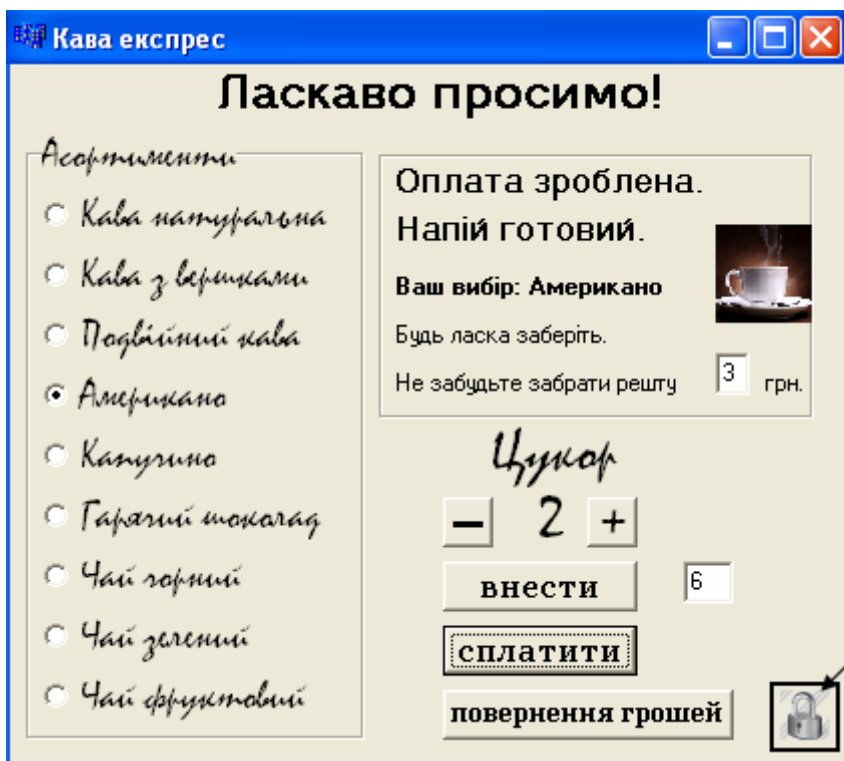


Рисунок 3.3 – Успішне виконання програми. Виведення повідомлення "Напій готов. Будь ласка заберіть"

```

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    Label8->Caption = "Напій готовий."; //замість "Приготування..."
    виводимо "Напій готовий."
    Label5->Caption = "Будь ласка заберіть."; //виводимо в Label5
дане повідомлення
    PlaySound("coins.wav", 0, SND_ASYNC); //включаємо звуковий
сигнал при видачі здачі
    Image1->Visible=true; //виводимо зображення
    Timer1->Interval = 0; //вимикаємо таймер про час приготування
напою
    Timer4->Interval = 3000; //включаємо таймер на 3 секунди, після
чого автомат переходить в початковий стан
}

```

Для того щоб наладчик міг зайти в системне меню створимо на головній формі керуючу кнопку. З панелі інструментів Additional вибираємо елемент BitBtn, щоб можна було відобразити зображення на поверхні кнопки (див. рис.3.3 № 1):

```

void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
    Form2 = new TForm2(this); //відкриваємо другу (допоміжну)
форму
    if(Form2->ShowModal() == mrCancel) //якщо друга форма
(модальна) закрита, то
        return; //повертаємося в головну форму
}

```

Так як системне меню захищено паролем, створимо **форму для введення даного пароля**. За допомогою мітки (Label) введемо повідомлення «Введіть пароль». Застосуємо елемент Edit для введення пароля (див. рис. 3.4 № 1). Створимо кнопку (button) при натисканні, на яку відкривається системне меню, у випадку правильного введення пароля. Правильним паролем є слово «password», яке було введено в нижньому регістрі. Якщо пароль невірний - вивести повідомлення «Пароль введено невірно. Спробуйте ще раз » (див. рис. 3.4). При вході у системне меню створимо вікно повідомлень про аварійні ситуації (див. рис. 3.5), яке буде видаватися випадковим чином:

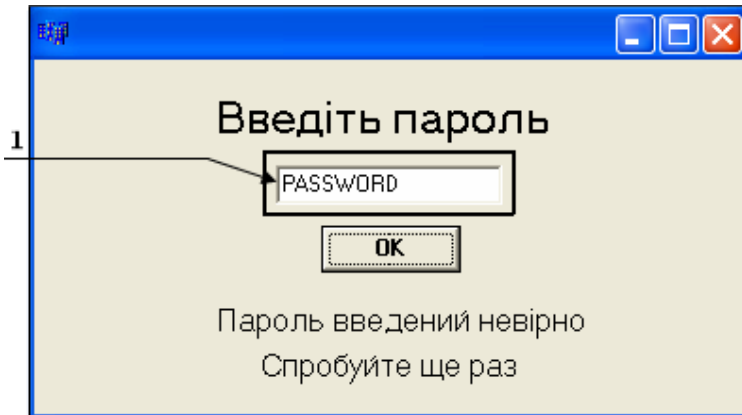


Рисунок 3.4 – Невірне введення пароля

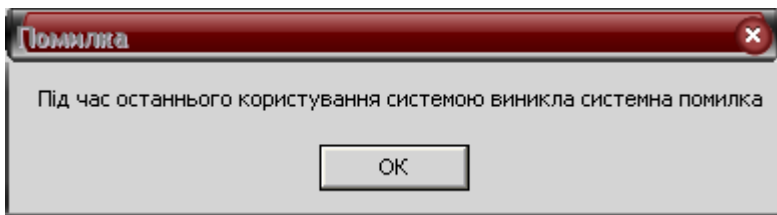


Рисунок 3.5 – Вікно повідомлень про аварійні ситуації

```
void __fastcall TForm2::Button1Click(TObject *Sender)
{
    int OShibka= rand(); // змінна OShibka набуває випадкового
    значення
    if (Edit1->Text == "password") //якщо в елемент Edit1 записано
    слово password
    {   Form3 = new TForm3(this); //відкриваємо третю форму
        if(OShibka %2==0) //якщо випадкове значення парне число, то
        видається повідомлення про помилку
        {
            MessageBox(NULL, "Під час останнього користування
            системою виникла системна помилка", "Помилка", MB_OK);
        }
    }
}
```

```

    }
    if(Form3->ShowModal() == mrCancel) //якщо третю форму
(модальну) закриваємо, то
        return; //повертаємося в головну форму
    }
    else //інакше (якщо пароль введений невірно)
        Label2->Caption = "Пароль введений невірно"; //виводимо в
мітку Label2 дане повідомлення
        Label3->Caption = "Спробуйте ще раз"; //виводимо в мітку
Label3 дане повідомлення
    }

```

Форма системного меню. За допомогою мітки (Label) виведемо повідомлення «Інформація про стан апарату». Створимо панель (Panel) на якій розмістимо всю інформацію про стан апарату та керуючі кнопки (button) для поповнення і зняття грошей з кавового автомата (див. рис.3.6). Інформацію про стан апарату заповнимо випадковими значеннями:

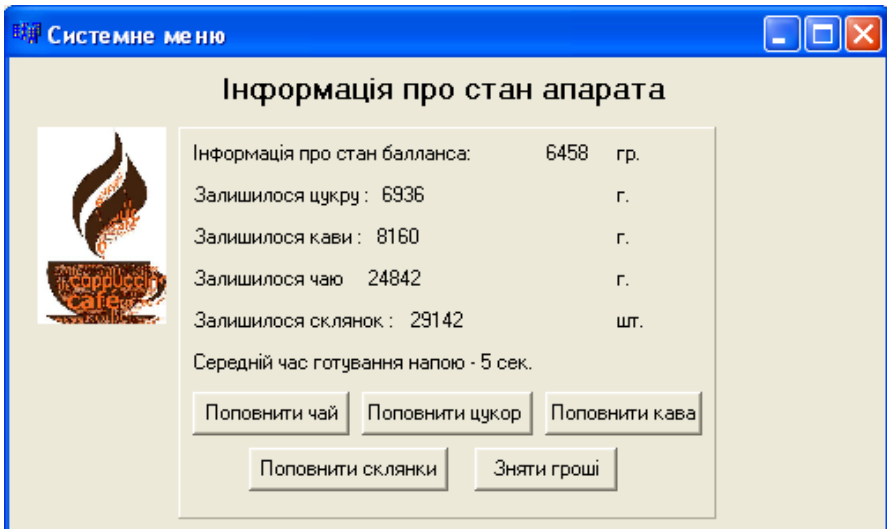


Рисунок 3.6 – Системне меню наладчика

```

fastcall TForm3::TForm3(TComponent* Owner): TForm(Owner)
{
    x = rand(); //змінна x набуває випадкового значення
    Label3->Caption = x; //в мітку Label3 записуємо випадкове
значення
    OstSah = rand();
    Label6->Caption = OstSah; //в мітку Label6 записуємо
випадкове значення (тобто залишок цукру дорівнює випадковому
значенню OstSah)
    OstKof = rand();
    Label10->Caption = OstKof; //залишок кави
    OstCh = rand();
    Label12->Caption = OstCh; // залишок чаю
    OstStkn = rand();
    Label18->Caption = OstStkn; // залишок склянок
}

```

Поповнення автомата буде здійснюватися збільшенням кожного найменування на 100 одиниць:

```

void __fastcall TForm3::Button1Click(TObject *Sender)
{
    x=0; // значення x = 0 (тобто після натискання на кнопку «Зняти
гроші» значення балансу дорівнює 0)
    Label3->Caption = x; // виводимо в мітку Label3 значення x
}
void __fastcall TForm3::Button2Click(TObject *Sender)
{
    OstSah+=100; // до змінної OstSah додаємо 100 одиниць
    Label6->Caption = OstSah; // виводимо в мітку Label6 нове
значення залишку цукру
}
void __fastcall TForm3::Button3Click(TObject *Sender)
{
    OstKof+=100
    Label10->Caption = OstKof; // нове значення залишку кави
}
void __fastcall TForm3::Button4Click(TObject *Sender)
{
    OstCh+=100;
}

```



```

Label12->Caption = OstCh; // нове значення залишку чаю
}
void __fastcall TForm3::Button5Click(TObject *Sender)
{
    OstStkn+=100;
    Label18->Caption = OstStkn; //нове значення залишку склянок
}

```

Форма «Про програму». Для цього застосуємо панель (Panel) на якій помістимо кілька міток для виведення інформації, а також додамо кнопку (button) для закриття даної форми (див. рис. 3.7).

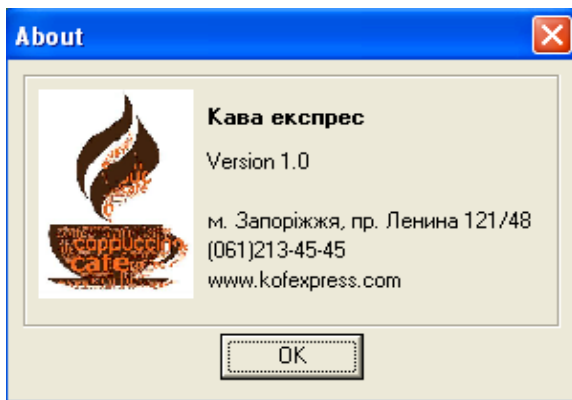


Рисунок 3.7 – Форма «Про програму»

3.3 Завдання до роботи

3.3.1 Ознайомитися з пунктом 3.2 і конспектом лекцій.

3.3.2 Для вибраного в першій лабораторній роботі об'єкту продумати такий дизайн форм, щоб користувач міг легко працювати з інтерфейсом.

3.3.3 Програмно реалізувати інтерфейс що містить наступні елементи керування: елементи для вибору, числові, текстові, дати і час.

3.3.4 Оформити звіт по роботі.

3.3.5 Відповісти на контрольні питання.

3.4 Зміст звіту

3.4.1 Тема і мета роботи

3.4.2 Завдання до роботи.

3.4.3 Текст програми.

3.4.4 Результати роботи програми.

3.4.5 Висновки за результатами виконання роботи.

3.5 Контрольні запитання

3.5.1 Які принципи необхідно враховувати при створенні інтерфейсу користувача?

3.5.2 Які чинники дають основу для вибору того або іншого елементу керування?

3.5.3 Які ви знаєте елементи керування для вибору? Переваги і недоліки названих елементів.

3.5.4 Які ви знаєте текстові елементи керування? Переваги і недоліки названих елементів.

3.5.5 Які ви знаєте числові елементи керування? Переваги і недоліки названих елементів.

3.5.6 Які ви знаєте варіанти введення дати і часу? У чому їх переваги і недоліки?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Тидвелл Дж. Разработка пользовательских интерфейсов [Текст]. – СПб.: Питер, 2008 – 416 с.

2. Солсо Р. Когнитивная психология [Текст]. – СПб.: Питер, 2006 – 589 с.

3. Сергеев С.Ф. Инженерная психология и эргономика [Текст]. – М.: НИИ школьных технологий, 2008 – 176 с.

4. Мунипов В.М. Эргономика: человекоориентированное проектирование техники, программных средств и среды : Учебник / В.М. Мунипов, В.П. Зинченко [Текст]. -М.: Логос, 2001 - 356 с.

5. Венда В.Ф. Инженерная психология и синтез систем отображения информации [Текст]. - М.: МЭСИ, 1975- 396 с.

6. Архангельський А.Я. С++ Builder 6. Справочное пособие. Книга 1. Язык С++ [Текст]. - М: "Бином-Пресс", 2004 - 544 с.