Для введення даних про кожний із товарів на складі в магазині використаємо структуру, яка буде містити наступні елементи:

– код товару,

– назва товару,

– дата надходження товару,

– кількість товару.

Для вводу і обробки списку товарів в магазині використаємо динамічний двоспрямований список з наступними елементами:

– структура – дані кожного товару,

– вказівник на наступний запис товару у списку,

– вказівник на попередній запис товару у списку.

Для організації інтерфейсу користувача будемо використовувати змінну цілого типу – номер обраного пункту меню.

Для роботи з двоспрямованим списком використаємо проміжні змінні-вказівники.

Для збереження списку товарів будемо використовувати бінарний файл «database.txt», який буде по замовчуванню зберігатись на диску «С:\» у попередньо створеному каталозі «С:\1». Для збереження вхідних і вихідних даних буде використовуватися один і той самий файл даних.

2.2 Математична постановка основних задач

Таблиця 2

|  |  |  |
| --- | --- | --- |
| Вхідні данні | Дії | Вихідні дані |
| key – цілочисленого типу (початкова ініціалізація key = -1),  можливі значення для виконання функцій – пункти меню (від 0 до 8). | Якщо key = 1, input\_product(синонім структури елементу);  add\_element\_struct(вказівник на список, вказівник структури елементу); | Новий товар додано у список. |
| Якщоkey = 2, rus\_code("Введите код товара для удаления: ");  зчитування коду елементу;delete\_product(список, введений код елементу);rus\_code("Для продолжения работы нажмите ENTER \n"); | Товар із кодом «код товару» видалено зі списку. |
| Якщо key = 3, rus\_code("Список товаров с начала \n");  show\_list\_head(список); | На екран виведено список товарів з початку списку. |
| Якщоkey = 4, rus\_code("Список товаров с конца\n");  show\_list\_tail(список); | На екран виведено список товарів з кінця списку. |
| Якщо key = 5, search\_unit(список);  rus\_code("Нажмите любую клавишу для продолжения\n"); | Знайдений елемент виведено на екран. |
| Якщо key = 6, save\_struct(список,1);  rus\_code("Список товаров дописан в файл\n") | Елемент дописано в кінець файлу. |
| Якщо key = 7, save\_struct(список,0);  rus\_code("Список товаров сохранен\n"); | Весь список збережено у файл. |
| Якщо key = 8, clear\_struct(вказівник на список);  read\_database(вказівник на список);  rus\_code("Список товаров получен из файла\n") | Список загружено з файлу. |
| Якщо key = 0, clear\_struct(вказівник на список);  rus\_code("Пожалуйста, нажмите любую клавишу для завершения\n"); | Очищено список.  Попередження про завершення роботи. |
| Якщо key не приймає жодного з допустимих значень,  clear\_struct(вказівник на структуру); cout<<"Error\n"; | Повідомлення про помилку. |
| Вказівник на буфер приймання рядка | CharToOem(s,bufer) (для перекодування використали функцію прототип з <windows.h>) | Перекодований рядок |
| &Ps (адреса початку списка), D (вказівник елементу списка-структури) | sklads\*Inc=new sklads;  якщо список не порожній – додаємо наступний елемент:  Inc->P.id=D.id;strcpy(Inc->P.name,D.name);strcpy(Inc->P.date,D.date); Inc->P.num=D.num;  і оновлюємо вказівники:  Inc->next=NULL;Inc->previous=current; current->next=Inc; last=Inc;  current=Inc;  якщо список порожній – додаємо перший елемент:  Inc->P.id=D.id; strcpy(Inc->P.name,D.name); strcpy(Inc->P.date,D.date); Inc->P.num=D.num;  і ініціалізуємо вказівники:  head=Inc;Inc->next=NULL;  Inc->previous=NULL; current=Inc;  last=head; | Sklads:  {Sklad  Previous  Next}-1  …  {Sklad  Previous  Next}-n |
| Ps (список), delete\_id (код товару для видалення) | ініціалізація вказівника на початок списку: sklads\*del = Ps; del = head;  якщо список порожній – видаємо повідомлення і завершуємо функцію;доки список не скінчився виконуємо пошук елементу, код якого дорівнює заданому (delete\_id):  якщо елемент для видалення – перший елемент списку (del==head) – оновлюємо вказівник на перший елемент (наступний елемент тепер перший) і видаляємо елемент: head=head->next;  if(head) head->previous=NULL;  else last=NULL; if(current==del) current=head; deletedel; break;  якщо елемент для видалення останній (del==last)– переміщуємо вказівник останнього елементу на попередній: last=last->previous;  if(last) last->next=NULL; if(current==del) current=last;  delete del; break;  якщо елемент для видалення в середині списку – переміщуємо вказівники (зв’язуємо попередній і наступний елементи):  del->previous->next=del->next;del->next->previous=del->previous; current=del->previous; delete del; break;  якщо елемент не був елементом для видалення, переходимо до наступного елементу: del=del->next; | Sklads:  {Sklad  Previous  Next}-1  …  {Sklad  Previous  Next}-n |

2.3 Характеристика системи програмування

Для розробки програмного забезпечення було використано середовище розробки MS Visual Studio 2012.

Visual Studio включає в себе редактор початкового програмного коду з підтримкою технології IntelliSenseі можливістю найпростішої [реорганізації (рефакторингу) коду](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3) – тобто процес зміни внутрішньої структури [програм](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)и, без зміни її зовнішньої поведінки. Оскільки кожне перетворення незначне, легше прослідкувати за його правильністю, и в той же час вся послідовність може призвести до істотної перебудови програми та покращенню узгодженості і чіткості. Вбудований відладчик може працювати як відладчик початкового програмного коду, так і як відладчик машинного рівня. Решта інструментів включає в себе редактор форм для спрощення створення графічного інтерфейсу, веб-редактор, дизайнер класів і дизайнер схеми бази даних.Visual Studioдозволяє створювати і підключати сторонні доповнення (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій початкового коду (наприклад,Subversion, Visual Source Safe), додавання нових наборів інструментів (наприклад, для візуального проектування коду на предметно-орієнтовних мовах програмування (як варіант, клієнт Team Explorer для роботи з Team Foundation Server).

Розробляти програму на C++ з допомогоюVisual Studio 2012 можна тільки під Windows7 SP1 та Windows 8, хоча останні оновлення дозволяють використовувати середу розробки і під Windows XP.

У Visual Studio 2012 повністю перероблено інтерфейс з використанням Windows Presentation Foundation(WPF), впроваджено інструменти ASP.NET, присутня підтримка динамічних розширень в мовах програмування C# і Visual Basic, використовуються нові шаблони проектів, інструментарій документування тестових сценаріїв і велика кількість нових бібліотек, що підтримуються у Windows 8, а також новий інструмент для спільної розробки програм.

Щоб створити програму в Visual Studioвиконується наступний порядок дій:

1. MS Visual Studio 2012\MS Visual Studio 2012.exe натиснути Enter.
2. В меню Файл вибираємо пункт «Создать» та натискаємо проект.
3. У списку типів проектів Visual C++ вибираємо «Консольное приложение» та натискаємо «Пустой проект» і задаємо ім'я проекту, натискаємо «ОК».
4. Для додавання в проект файлу з програмним кодом послідовно обираємо пункти«Добавить» і «Создать элемент» та задаємо ім'я файлу.
5. Вводимо програмний код.
6. В меню «Построение» вибираємо «Построить решение», далі у вікні «Выходные данные»буде відображено інформацію про хід виконання та помилки. Також можна використати кнопку F5 для запуску «Отладки».
7. Далі необхідно вибрати меню «Файл» - «Сохранить». При збереженні автоматично виконується компіляція проекту.

2.4 Опис динамічної структури даних, що використовується в КР, та дій, що над нею виконуються

Дані організовано у вигляді двоспрямованого динамічного списку, елементи якого у свою чергу є структурою фіксованої кількості полів для збереження даних про кожний з товарів в магазині.

Для вводу даних про кожний із товарів в магазині створено структуру sklad, яка містить наступні елементи:

id – код товару (ціле число),

name – назва товару (64 символи),

date – дата надходження товару (64 символи),

num – кількість товару (ціле число).

Оголошення структури:

struct sklad

{

int id;

char name[64];

char date[64];

int num;

}.

Для вводу і обробки списку товарів в магазині створено динамічний двоспрямований список sklads з наступними елементами:

sklad (для зручності використаємо синонім «Р») – запис про товар у магазині (структура),

\*next – вказівник на наступний товар у списку,

\*previous – вказівник на попередній товар у списку.

Оголошення структури:

struct sklads

{

sklad P;

struct sklads \*next;

struct sklads \*previous;

}.

Структура sklad є елементом двоспрямованого динамічного списку sklads.

Над даними виконуються наступні дії:

* введення даних про товар (заповнення структури sklad),
* додавання елементу списку (розширення структури sklads – вказівники кожного елементу вказують на попередній і наступний елементи),
* вивід товару (консольне відображення даних, які зберігаються в полях структури sklad),
* вивід списку товарів з початку (консольне відображення даних, які зберігаються в полях кожної структури sklad, включеної до списку, відображення даних починається з першого елементу),
* вивід списку товарів з кінця (консольне відображення даних, які зберігаються в полях кожної структури sklad, включеної до списку, відображення даних починається з останнього елементу),
* видалення елементу списку(звуження структури sklads – вказівники кожного елементу вказують на попередній і наступний елементи),
* пошук елементу у списку (починаючи з початку списку виконується пошук структури даних про товар, код якого дорівнює вказаному),
* збереження списку товарів в файл (запис усіх елементів списку у бінарний файл),
* отримання списку товарів з файлу (зчитування усіх елементів списку з ініціалізацією усіх вказівників),
* очищення списку товарів (знищення динамічного списку і усіх його елементів).

2.5 Опис програмного інтерфейсу з користувачем

Інтерфейс користувача організовано за допомогою меню. При запуску програми на екран виводиться:

Меню

1. Добавление записи о товаре

2. Удаление товара со склада

3. Показ списка товаров с начала

4. Показ списка товаров с конца

5. Поиск товара

6. Добавление записи в файл

7. Перезапись файла

8. Вычитка списка из файла

Выход

Ваш выбор: \_\_.

Вибір здійснюється вводом з клавіатури номеру обраного пункту меню, без крапки.

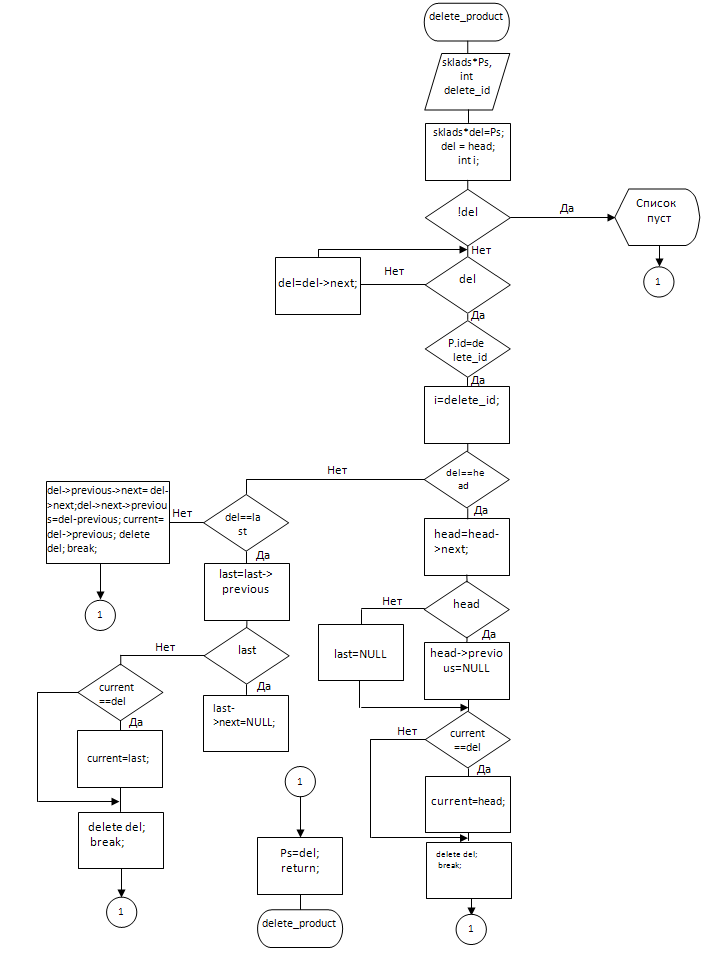
Якщо введено номер одного з пунктів меню, виконується обрана дія, якщо введено довільний (не припустимий) символ (набір символів), на екрані відображається повідомлення: «Error», та при натисканні будь-якої клавіші програма завершує роботу.

2.6 Проектування схем алгоритмів згідно вимог

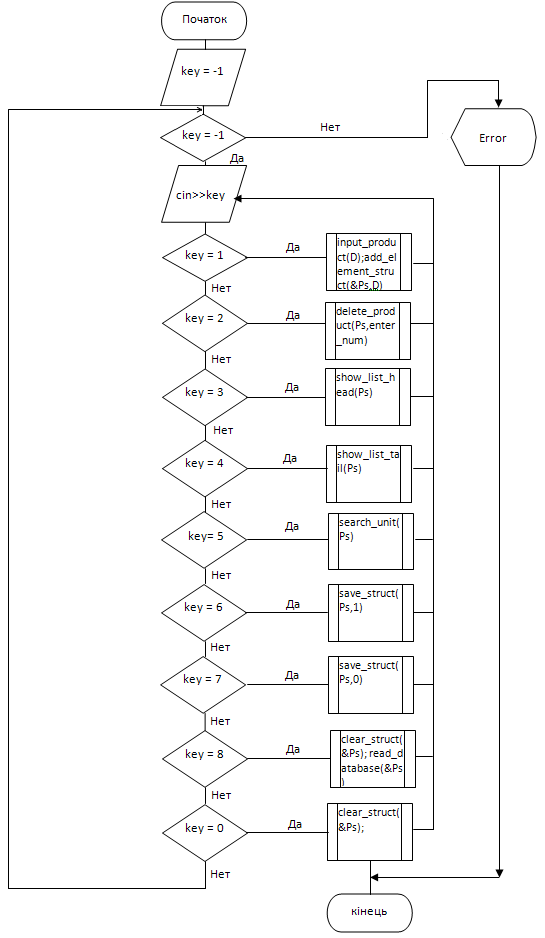
Таблиця 3

|  |  |
| --- | --- |
| АМ функції перекодування символів (російською) | АМ додавання товару (елементу) у список |
| АМ функції вводу товару |  |

АМ функції видалення заданого (за кодом товару) товару (елементу) із списку



АМ основної програми



2.7 Опис головної програми та опис програмних модулів

У головній програмі організовано підключення вбудованих бібліотек і заголовних файлів, оголошення та, за необхідності, ініціалізацію змінних, структур даних, функцій. Основний процес організації інтерфейсу користувача та звернення до усіх функцій програми також організовано у головній програмі. Головна програма складається з наступних блоків:

* блок підключення стандартних бібліотек прототипів функцій С++;
* блок оголошення структури даних і динамічного списку;
* блок оголошення і ініціалізації змінних-вказівників;
* блок оголошення функцій;
* блок основної програми:

оголошення змінних,

відображення інтерфейсу користувача (меню дій),

обробка результату вибору – виконання обраної дії, за допомогою виклику функції.

Організоване, за допомогою оператору множинного вибору (case), меню користувача, дозволяє обирати, яку дію потрібно виконати. Кожну дію в меню оформлено у вигляді окремої функції, описаної у блоці функцій:

Таблиця 4

|  |  |  |
| --- | --- | --- |
| Назва і формат виклику функції | Функціональне призначення | Стислий опис функціоналу |
| char bufer [255];  char\*rus\_code (char\*s);  rus\_code(“Строка виводу”) | Функція перекодування (російською) символів, що знаходяться у буфері виводу.  Використовується для організації інтерфейсу користувача (виводу на екран меню, повідомлень). | Використовуючи функцію прототип <CharToOem> з бібліотеки <windows.h>, дозволяє виводити на екран російські літери. Виконує перекодування символів з ANSI коду ОС Windows в OEM код консолі Windows і повертає в буфер перекодований рядок. |
| void add\_element\_struct (sklads \*\*Ps, sklad&D);  add\_element\_struct (вказівник на список, адреса елементу, який додається) | Функція додавання елемента списку. Використовується для додавання елементу (товару), введеного користувачем, або вичитаного з файлу даних, до динамічного списку. | Використовуючи роботу з вказівниками додає передану у функцію структуру з параметрами товару, заповнену у функції введення товару, до динамічного двоспрямованого списку, вказівник на який передано вхідним параметром. |
| void input\_product (sklad&D);  input\_product (адреса структури елементу (товар)) | Функція введення товару. Використовується для запиту, зчитування і заповнення структури даних про товар, який потрібно додати у список. | Використовуючи класові об’єкти, створювані за допомогою заголовного файлу бібліотеки <iostream>, – потоки вводу виводу, здійснює запит, отримання даних про товар і заповнення структури даних товару (елементу списку). |
| void print\_product (sklad); print\_product (структура елементу (товар)) | Функція виводу товару. Використовується для виводу елементу (товару) із списку,введеного користувачем, або вичитаного з файлу даних. | Використовуючи класові об’єкти, створювані за допомогою заголовного файлу бібліотеки <iostream>, – потоки вводу виводу, здійснює вивід на екран даних з полів структури елементу списку (товару), який передано вхідним параметром. |
| void show\_list\_head (sklads\*Ps);show\_list\_head (список) | Функція виводу списку товарів з початку. Використовується для виводу на екран усіх структур даних про товар, які є елементами динамічного списку, починаючи з першого елементу списку. | Використовуючи роботу з вказівниками та функцію виводу елементу, здійснює послідовний вивід на екран даних з структури кожного елементу списку (товару), починаючи з першого елементу списку. Список передано вхідним параметром. |
| void show\_list\_tail (sklads\*Ps);  show\_list\_tail (список) | Функція виводу списку товарів з кінця. Використовується для виводу на екран усіх структур даних про товар, які є елементами динамічного списку, починаючи з останнього елементу списку. | Використовуючи роботу з вказівниками та функцію виводу елементу, здійснює послідовний вивід на екран даних з структури кожного елементу списку (товару), починаючи з останнього елементу списку. Список передано вхідним параметром. |
| void delete\_product (sklads\*, int);  delete\_product (список, код товару) | Функція видалення елементу (товару) із списку. Використовується для видалення із списку елементу (структури товару), код якого дорівнює заданому (список і код передаються вхідними параметрами). | Використовуючи роботу з вказівниками, потоки вводу виводу, здійснює пошук у списку елементу (товару), код якого дорівнює заданому та видалення знайденого товару (елементу) зі списку, методом переміщення вказівників, та очищення пам’яті, яку займав елемент. Якщо список пустий, або видалення виконано успішно, відображаються відповідні повідомлення. |
| void search\_unit (sklads\*Ps);  search\_unit (список) | Функція пошуку елементу у списку. Використовується для пошуку у списку елементу (структури товару), код товару якого дорівнює введеному користувачем. | Використовуючи класові об’єкти, створювані за допомогою заголовного файлу бібліотеки<iostream>, – потоки вводу виводу, здійснює запит і зчитування коду товару, який потрібно знайти. Використовуючи роботу з вказівниками, потоки вводу виводу та функцію виводу товару, здійснює пошук у списку елементу (товару), код якого дорівнює заданому та вивід знайденого товару на екран. Якщо список пустий, або товар по коду не знайдено, відображаються відповідні повідомлення. |
| void save\_struct (sklads\*,int);  save\_struct(список,1-додавання в кінець файлу/0 – перезапис файлу); | Функція збереження списку товарів в файл. Використовується для збереження списку в файл даних у двох режимах: додавання списку в кінець файлу/перезапис файлу, в залежності від заданого режиму (1або 0). | Використовуючи об’єкти, створювані за допомогою заголовного файлу бібліотеки <fstream>, організацію файлового потоку вводу, класові об’єкти, створювані за допомогою заголовного файлу бібліотеки <iostream>, – потоки вводу виводу, і роботу з вказівниками, здійснює збереження списку в файл даних у двох режимах: додавання списку в кінець файлу/перезапис файлу, в залежності від заданого режиму (1або 0). У разі виникнення помилок при роботі з файлом користувачу надається повідомлення. |
| void read\_database (sklads\*\*);  read\_database (вказівник на список) | Функція отримання списку товарів з файлу.  Використовується для вичитки списку із файлу даних в структуру двоспрямованого списку. | Використовуючи об’єкти, створювані за допомогою заголовного файлу бібліотеки <fstream>, організацію файлового потоку вводу, класові об’єкти, створювані за допомогою заголовного файлу бібліотеки <iostream>, – потоки вводу виводу, і роботу з вказівниками, здійснює зчитування елементів списку з файлу даних, додавання елементів в список з ініціалізацією вказівників списку, шляхом виклику функції додавання елементу. У разі виникнення помилок при роботі з файлом користувачу надається повідомлення. |
| void clear\_struct (sklads\*\*);  clear\_struct (вказівник на список) | Функція очищення списку товарів.  Використовується для видалення всього списку, очищення пам’яті у разі завершення роботи програми, або зчитування списку із файлу. | Використовуючи роботу з вказівниками та класові об’єкти, створювані за допомогою заголовного файлу бібліотеки <iostream>, – потоки вводу виводу, виконується видалення кожного елементу та знищення вказівників списку, якщо список пустий, користувачеві надається повідомлення. |

2.8 Опис та обґрунтування тестових прикладів виконання програми

При створені програми використовувався метод покрокової розробки, оскільки цей метод дозволяє весь час контролювати управляючу та інформаційну структуру алгоритмів. Враховуючи модульність організації програми, для налагодження програми створювались функціональні тести для перевірки базових функцій програми та перевірки коректності виконання програмних переходів. Наведені у таблиці тести включають опис вхідних даних і опис очікуваних результатів для кожного тесту.

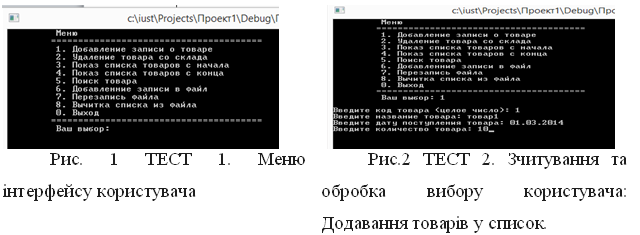
Таблиця 5

|  |  |
| --- | --- |
| Вхідні дані | Очікуваний результат |
| ТЕСТ 1. Меню інтерфейсу користувача | |
| Виведення на екран пунктів меню та запит вибору користувача. | На екрані відображено меню та рядок вводу обраного пункту:   1. Добавление записи о товаре 2. Удаление товара со склада   3. Показ списка товаров с начала  4. Показ списка товаров с конца  5. Поиск товара  6. Добавление записи в файл  7. Перезапись файла  8. Вычитка списка из файла  0. Выход  Ваш выбор: (cin>>key) |
| ТЕСТ 2. Зчитування та обробка вибору користувача | |
| key = -1 – початкове значення,  cin>>key | key = 1– Додавання запису про товар:  input\_product(D); add\_element\_struct(&Ps,D);  key = 2– Видалення товару зі складу:  delete\_product(Ps,enter\_num); вивід повідомлення:  "Для продолжения работы нажмите ENTER";  key = 3 – відображення списку товарів з початку: вивід повідомлення: "Список товаров с начала"; show\_list\_head(Ps);  key =4 – відображення списку товарів з кінця: вивід повідомлення:"Список товаров с конца"; show\_list\_tail(Ps);  key =5 – пошук товару за кодом: search\_unit(Ps);  вивід повідомлення: "Нажмите любую клавишу для продолжения";  key = 6 – додавання списку в файл: save\_struct(Ps,1); вивід повідомлення: "Список товаров дописан в файл";  key =7 – оновлення вмісту файлу:save\_struct(Ps,0); вивід повідомлення: "Список товаров сохранен ");  key =8 – вичитка списку з файлу: clear\_struct(&Ps); read\_database(&Ps);  key =0 – вивільнення пам’яті і завершення програми: clear\_struct(&Ps); повідомлення: "Пожалуйста, нажмите любую клавишу для завершения"; при натискані кнопки – завершення програми;  keyне входить в область допустимих значень–вивільнення пам’яті і переривання програми:clear\_struct(&Ps); повідомлення: "Error". |

## 2.9 Аналіз отриманих результатів

Функціональність програми відповідає поставленій меті і вимогам. Помилок в роботі програми не виявлено.

Результати проведених тестів:



Тестування можна вважати успішно завершеним.

на Allbest.ru