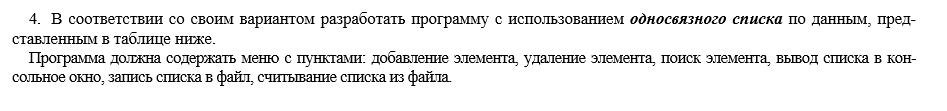
**Варыянт 3**





#include <Windows.h>

#include <iostream>

using namespace std;

// Структура для прадстаўлення вузла звязанага спісу

struct LinkedList {

int value;

LinkedList\* next;

};

// Прататыпы функцый

void insert(LinkedList\*& ptrNext);

void display(LinkedList\* ptr);

void deleteListItem(LinkedList\*& ptr);

void writeToFile(LinkedList\* ptr, const char\* fileName);

void readFromFile(LinkedList\*& ptr, const char\* fileName);

void search(LinkedList\* ptr);

void sum5(LinkedList\* ptr);

int main() {

SetConsoleOutputCP(1251);

LinkedList\* ptr = nullptr; // Указатель на текущий узел

int choice;

do {

// Вывад меню

cout << "\n----- Меню -----\n";

cout << "1. Дадаць элемент\n";

cout << "2. Адлюстраваць спіс\n";

cout << "3. Выдаліць элемент(толькі першае яго уваходжанне)\n";

cout << "4. Запісаць у файл\n";

cout << "5. Прачытаць з файла\n";

cout << "6. Пошук\n";

cout << "7. Знайсці суму дадатных лікаў, кратных 5\n";

cout << "0. Выйсці\n";

cout << "Выберыце дзеянне: ";

// Увод выбару карыстальніка

cin >> choice;

switch (choice) {

case 1:

insert(ptr); // Даданне элемента

break;

case 2:

display(ptr); // Адлюстраванне спісу

break;

case 3:

deleteListItem(ptr); // Выдаленне элемента

display(ptr); // Адлюстраванне абноўленага спісу

break;

case 4:

writeToFile(ptr, "list\_data.txt"); // Запіс у файл

break;

case 5:

readFromFile(ptr, "list\_data.txt"); // Чытанне з файла

break;

case 6:

search(ptr); // Пошук

break;

case 7:

sum5(ptr); // Сума дадатных лікаў, кратных 5

break;

case 0:

cout << "Выхад...";

default:

cout << "Няправільны выбар. Калі ласка, паспрабуйце яшчэ раз.\n";

}

} while (choice!= 0);

// Вызваленне памяці, выдзеленай для вузлоў спісу

while (ptr) {

LinkedList\* temp = ptr;

ptr = ptr->next;

delete temp;

}

return 0;

}

// Функцыя для дадання новага элемента ў пачатак звязанага спісу

void insert(LinkedList\*& ptrNext) {

LinkedList\* ptr;

cout << "Увядзіце лік: ";

int num;

cin >> num;

// Ствараем новы вузел і запаўняем яго дадзенымі

ptr = new LinkedList;

ptr->value = num;

ptr->next = ptrNext; // Усталёўваны паказальнік на наступны элемент(перададзены ў функцыю вузел спісу)

ptrNext = ptr; // Усталёўваны паказальнік на новы элемент у пачатак спісу

}

// Функцыя для адлюстравання элементаў звязанага спісу

void display(LinkedList\* ptr) {

if (ptr == nullptr) {

cout << "Спіс пусты\n";

}

while (ptr) {

cout << ptr->value << " "; // Выводзім значэнне бягучага элемента

ptr = ptr->next; // Пераходзім да наступнага элемента

}

cout << '\n';

}

// Функцыя для выдалення элемента з звязанага спісу

void deleteListItem(LinkedList\*& ptr) {

cout << "Увядзіце лік, чкі трэба выдаліць: ";

int num;

cin >> num;

LinkedList\* current = ptr, \* prev = nullptr, \* tmp;

// Шукаем элемент з зададзеным значэннем

while (current != nullptr && current->value != num) {

prev = current;

current = current->next;

}

// Правяраем, ці знойдзены элемент

if (current == nullptr) {

cout << "Няма такога элемента ў спісе\n";

}

else if (prev == nullptr) { // Калі выдаляем першы элемент

tmp = ptr;

ptr = ptr->next;

delete tmp;

}

else { // Выдаляем элемент з сярэдзіны або канца спісу

prev->next = current->next;

delete current;

}

}

// Функцыя для запісу элементаў звязанага спісу ў файл

void writeToFile(LinkedList\* ptr, const char\* fileName) {

if (ptr == nullptr) {

cout << "Спіс пусты, няма чаго запісаць у файл\n";

return;

}

FILE\* file;

if (fopen\_s(&file, fileName, "wt") != 0) {

cout << "Не атрымалася адкрыць файл для запісу\n";

return;

}

// Запісваем элементы спісу ў файл

while (ptr) {

fprintf\_s(file, "%d ", ptr->value);

ptr = ptr->next;

}

fprintf\_s(file, "\n");

fclose(file);

// Правяраем паспяховае закрыццё файла

fclose(file);

cout << "Дадзеныя запісаны ў файл" << fileName << " паспяхова\n";

}

// Функцыя для чытання элементаў з файла і стварэння звязанага спісу

void readFromFile(LinkedList\*& ptr, const char\* fileName) {

LinkedList\* p = nullptr;

int d;

FILE\* file;

// Спрабуем адкрыць файл для чытання

if (fopen\_s(&file, fileName, "r") != 0) {

cout << "Не атрымалася адкрыць файл для чытання\n";

return;

}

// Счытваем лікі з файла і ствараем новыя вузлы спісу

while (fscanf\_s(file, "%d", &d) == 1) {

p = new LinkedList;

p->value = d;

p->next = ptr;

ptr = p;

}

// Калі былі лічаныя дадзеныя, усталёўваны паказальнік на пачатак спісу

if (p) {

ptr = p;

}

// Правяраем паспяховае закрыццё файла

fclose(file);

cout << "Дадзеныя запісаны ў спіс паспяхова\n";

}

// Функцыя для пошуку элемента ў звязаным спісе

void search(LinkedList\* ptr) {

cout << "Увядзіце лік, якія трэба знайсці: ";

int number, counter = 1;

cin >> number;

// Праходзім па звязаным спісе

while (ptr) {

// Правяраем, ці роўны бягучы элемент шуканаму ліку

if (ptr->value == number) {

// Выводзім інфармацыю аб знойдзеным элеменце

cout << "Лік " << number << " знойдзены ў спісе. Гэта элемент пад нумарам " << counter << endl;

}

ptr = ptr->next; // Пераходзім да наступнага элемента

counter++; // Павялічваем лічыльнік сустрэтых элементаў

}

}

void sum5(LinkedList\* ptr) {

int sum5 = 0; // Пераменная для захоўвання сумы лікаў, кратных 5

// Праходзім па звязаным спісе

while (ptr) {

if (ptr->value % 5 == 0 && ptr->value > 0)

sum5 += ptr->value; // Калі лік кратна 5, дадаем яго да сумы

ptr = ptr->next; // Пераходзім да наступнага вузла

}

// Выводзім вынік

if (sum5 != 0) {

cout << "Сума лікаў, кратных 5: " << sum5;

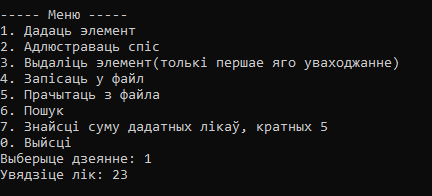
}

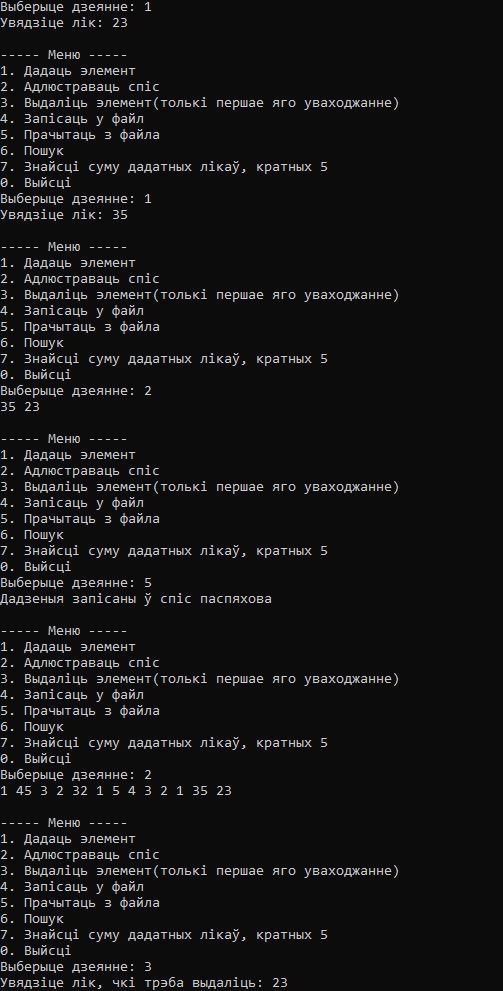
else {

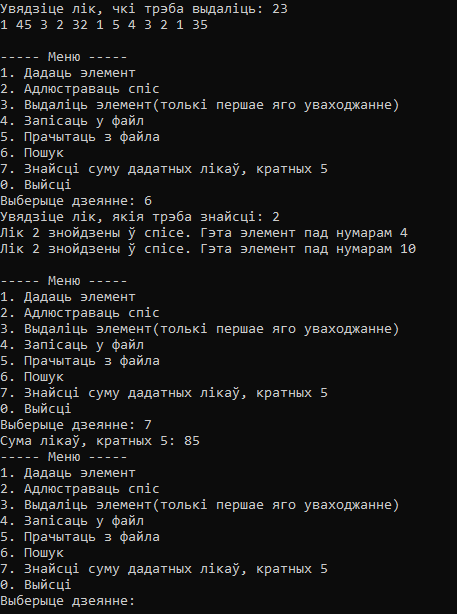
cout << "Лікаў, кратных 5 няма";

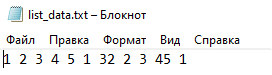
}

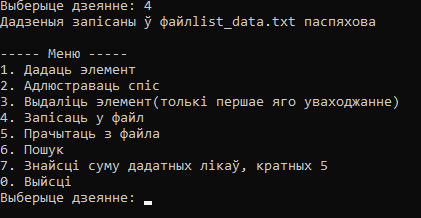
}

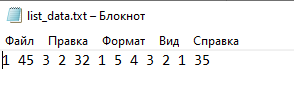












**Варыянт 1**



Адрозненне ад 3 варыянта толькі ў тым, што трэба змяніць адну функцыю

#include <Windows.h>

#include <iostream>

using namespace std;

// Структура для прадстаўлення вузла звязанага спісу

struct LinkedList {

int value;

LinkedList\* next;

};

// Прататыпы функцый

void insert(LinkedList\*& ptrNext);

void display(LinkedList\* ptr);

void deleteListItem(LinkedList\*& ptr);

void writeToFile(LinkedList\* ptr, const char\* fileName);

void readFromFile(LinkedList\*& ptr, const char\* fileName);

void search(LinkedList\* ptr);

void sum(LinkedList\* ptr);

int main() {

SetConsoleOutputCP(1251);

LinkedList\* ptr = nullptr; // Указатель на текущий узел

int choice;

do {

// Вывад меню

cout << "\n----- Меню -----\n";

cout << "1. Дадаць элемент\n";

cout << "2. Адлюстраваць спіс\n";

cout << "3. Выдаліць элемент(толькі першае яго уваходжанне)\n";

cout << "4. Запісаць у файл\n";

cout << "5. Прачытаць з файла\n";

cout << "6. Пошук\n";

cout << "7. Знайсці суму дадатных лікаў\n";

cout << "0. Выйсці\n";

cout << "Выберыце дзеянне: ";

// Увод выбару карыстальніка

cin >> choice;

switch (choice) {

case 1:

insert(ptr); // Даданне элемента

break;

case 2:

display(ptr); // Адлюстраванне спісу

break;

case 3:

deleteListItem(ptr); // Выдаленне элемента

display(ptr); // Адлюстраванне абноўленага спісу

break;

case 4:

writeToFile(ptr, "list\_data.txt"); // Запіс у файл

break;

case 5:

readFromFile(ptr, "list\_data.txt"); // Чытанне з файла

break;

case 6:

search(ptr); // Пошук

break;

case 7:

sum(ptr); // Сума дадатных лікаў

break;

case 0:

cout << "Выхад...";

default:

cout << "Няправільны выбар. Калі ласка, паспрабуйце яшчэ раз.\n";

}

} while (choice!= 0);

// Вызваленне памяці, выдзеленай для вузлоў спісу

while (ptr) {

LinkedList\* temp = ptr;

ptr = ptr->next;

delete temp;

}

return 0;

}

// Функцыя для дадання новага элемента ў пачатак звязанага спісу

void insert(LinkedList\*& ptrNext) {

LinkedList\* ptr;

cout << "Увядзіце лік: ";

int num;

cin >> num;

// Ствараем новы вузел і запаўняем яго дадзенымі

ptr = new LinkedList;

ptr->value = num;

ptr->next = ptrNext; // Усталёўваны паказальнік на наступны элемент(перададзены ў функцыю вузел спісу)

ptrNext = ptr; // Усталёўваны паказальнік на новы элемент у пачатак спісу

}

// Функцыя для адлюстравання элементаў звязанага спісу

void display(LinkedList\* ptr) {

if (ptr == nullptr) {

cout << "Спіс пусты\n";

}

while (ptr) {

cout << ptr->value << " "; // Выводзім значэнне бягучага элемента

ptr = ptr->next; // Пераходзім да наступнага элемента

}

cout << '\n';

}

// Функцыя для выдалення элемента з звязанага спісу

void deleteListItem(LinkedList\*& ptr) {

cout << "Увядзіце лік, чкі трэба выдаліць: ";

int num;

cin >> num;

LinkedList\* current = ptr, \* prev = nullptr, \* tmp;

// Шукаем элемент з зададзеным значэннем

while (current != nullptr && current->value != num) {

prev = current;

current = current->next;

}

// Правяраем, ці знойдзены элемент

if (current == nullptr) {

cout << "Няма такога элемента ў спісе\n";

}

else if (prev == nullptr) { // Калі выдаляем першы элемент

tmp = ptr;

ptr = ptr->next;

delete tmp;

}

else { // Выдаляем элемент з сярэдзіны або канца спісу

prev->next = current->next;

delete current;

}

}

// Функцыя для запісу элементаў звязанага спісу ў файл

void writeToFile(LinkedList\* ptr, const char\* fileName) {

if (ptr == nullptr) {

cout << "Спіс пусты, няма чаго запісаць у файл\n";

return;

}

FILE\* file;

if (fopen\_s(&file, fileName, "wt") != 0) {

cout << "Не атрымалася адкрыць файл для запісу\n";

return;

}

// Запісваем элементы спісу ў файл

while (ptr) {

fprintf\_s(file, "%d ", ptr->value);

ptr = ptr->next;

}

fprintf\_s(file, "\n");

fclose(file);

// Правяраем паспяховае закрыццё файла

fclose(file);

cout << "Дадзеныя запісаны ў файл" << fileName << " паспяхова\n";

}

// Функцыя для чытання элементаў з файла і стварэння звязанага спісу

void readFromFile(LinkedList\*& ptr, const char\* fileName) {

LinkedList\* p = nullptr;

int d;

FILE\* file;

// Спрабуем адкрыць файл для чытання

if (fopen\_s(&file, fileName, "r") != 0) {

cout << "Не атрымалася адкрыць файл для чытання\n";

return;

}

// Счытваем лікі з файла і ствараем новыя вузлы спісу

while (fscanf\_s(file, "%d", &d) == 1) {

p = new LinkedList;

p->value = d;

p->next = ptr;

ptr = p;

}

// Калі былі лічаныя дадзеныя, усталёўваны паказальнік на пачатак спісу

if (p) {

ptr = p;

}

// Правяраем паспяховае закрыццё файла

fclose(file);

cout << "Дадзеныя запісаны ў спіс паспяхова\n";

}

// Функцыя для пошуку элемента ў звязаным спісе

void search(LinkedList\* ptr) {

cout << "Увядзіце лік, якія трэба знайсці: ";

int number, counter = 1;

cin >> number;

// Праходзім па звязаным спісе

while (ptr) {

// Правяраем, ці роўны бягучы элемент шуканаму ліку

if (ptr->value == number) {

// Выводзім інфармацыю аб знойдзеным элеменце

cout << "Лік " << number << " знойдзены ў спісе. Гэта элемент пад нумарам " << counter << endl;

}

ptr = ptr->next; // Пераходзім да наступнага элемента

counter++; // Павялічваем лічыльнік сустрэтых элементаў

}

}

void sum(LinkedList\* ptr) {

int sum = 0; // Пераменная для захоўвання сумы лікаў

// Праходзім па звязаным спісе

while (ptr) {

if (ptr->value > 0)

sum += ptr->value; // Калі лік дадатны, дадаем яго да сумы

ptr = ptr->next; // Пераходзім да наступнага вузла

}

// Выводзім вынік

if (sum!= 0) {

cout << "Сума датных лікаў: " << sum;

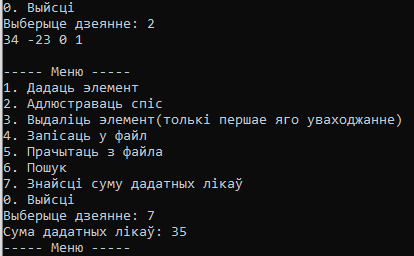
}

else {

cout << "Дадатных лікаў няма";

}

}



**Варыянт 2**



#include <Windows.h>

#include <iostream>

using namespace std;

// Структура для прадстаўлення вузла звязанага спісу

struct LinkedList {

double value;

LinkedList\* next;

};

// Прататыпы функцый

void insert(LinkedList\*& ptrNext);

void display(LinkedList\* ptr);

void deleteListItem(LinkedList\*& ptr);

void writeToFile(LinkedList\* ptr, const char\* fileName);

void readFromFile(LinkedList\*& ptr, const char\* fileName);

void search(LinkedList\* ptr);

void sum(LinkedList\* ptr);

int main() {

SetConsoleOutputCP(1251);

LinkedList\* ptr = nullptr; // Указатель на текущий узел

int choice;

do {

// Вывад меню

cout << "\n----- Меню -----\n";

cout << "1. Дадаць элемент\n";

cout << "2. Адлюстраваць спіс\n";

cout << "3. Выдаліць элемент(толькі першае яго уваходжанне)\n";

cout << "4. Запісаць у файл\n";

cout << "5. Прачытаць з файла\n";

cout << "6. Пошук\n";

cout << "7. Знайсці сярэдняе дадатных лікаў\n";

cout << "0. Выйсці\n";

cout << "Выберыце дзеянне: ";

// Увод выбару карыстальніка

cin >> choice;

switch (choice) {

case 1:

insert(ptr); // Даданне элемента

break;

case 2:

display(ptr); // Адлюстраванне спісу

break;

case 3:

deleteListItem(ptr); // Выдаленне элемента

display(ptr); // Адлюстраванне абноўленага спісу

break;

case 4:

writeToFile(ptr, "list\_data.txt"); // Запіс у файл

break;

case 5:

readFromFile(ptr, "list\_data.txt"); // Чытанне з файла

break;

case 6:

search(ptr); // Пошук

break;

case 7:

sum(ptr); // Сума дадатных лікаў

break;

case 0:

cout << "Выхад...";

default:

cout << "Няправільны выбар. Калі ласка, паспрабуйце яшчэ раз.\n";

}

} while (choice != 0);

// Вызваленне памяці, выдзеленай для вузлоў спісу

while (ptr) {

LinkedList\* temp = ptr;

ptr = ptr->next;

delete temp;

}

return 0;

}

// Функцыя для дадання новага элемента ў пачатак звязанага спісу

void insert(LinkedList\*& ptrNext) {

LinkedList\* ptr;

cout << "Увядзіце лік: ";

double num;

cin >> num;

// Ствараем новы вузел і запаўняем яго дадзенымі

ptr = new LinkedList;

ptr->value = num;

ptr->next = ptrNext; // Усталёўваны паказальнік на наступны элемент(перададзены ў функцыю вузел спісу)

ptrNext = ptr; // Усталёўваны паказальнік на новы элемент у пачатак спісу

}

// Функцыя для адлюстравання элементаў звязанага спісу

void display(LinkedList\* ptr) {

if (ptr == nullptr) {

cout << "Спіс пусты\n";

}

while (ptr) {

cout << ptr->value << " "; // Выводзім значэнне бягучага элемента

ptr = ptr->next; // Пераходзім да наступнага элемента

}

cout << '\n';

}

// Функцыя для выдалення элемента з звязанага спісу

void deleteListItem(LinkedList\*& ptr) {

cout << "Увядзіце лік, чкі трэба выдаліць: ";

double num;

cin >> num;

LinkedList\* current = ptr, \* prev = nullptr, \* tmp;

// Шукаем элемент з зададзеным значэннем

while (current != nullptr && current->value != num) {

prev = current;

current = current->next;

}

// Правяраем, ці знойдзены элемент

if (current == nullptr) {

cout << "Няма такога элемента ў спісе\n";

}

else if (prev == nullptr) { // Калі выдаляем першы элемент

tmp = ptr;

ptr = ptr->next;

delete tmp;

}

else { // Выдаляем элемент з сярэдзіны або канца спісу

prev->next = current->next;

delete current;

}

}

// Функцыя для запісу элементаў звязанага спісу ў файл

void writeToFile(LinkedList\* ptr, const char\* fileName) {

if (ptr == nullptr) {

cout << "Спіс пусты, няма чаго запісаць у файл\n";

return;

}

FILE\* file;

if (fopen\_s(&file, fileName, "wt") != 0) {

cout << "Не атрымалася адкрыць файл для запісу\n";

return;

}

// Запісваем элементы спісу ў файл

while (ptr) {

fprintf\_s(file, "%lf ", ptr->value);

ptr = ptr->next;

}

fprintf\_s(file, "\n");

fclose(file);

// Правяраем паспяховае закрыццё файла

fclose(file);

cout << "Дадзеныя запісаны ў файл" << fileName << " паспяхова\n";

}

// Функцыя для чытання элементаў з файла і стварэння звязанага спісу

void readFromFile(LinkedList\*& ptr, const char\* fileName) {

LinkedList\* p = nullptr;

double d;

FILE\* file;

// Спрабуем адкрыць файл для чытання

if (fopen\_s(&file, fileName, "r") != 0) {

cout << "Не атрымалася адкрыць файл для чытання\n";

return;

}

// Счытваем лікі з файла і ствараем новыя вузлы спісу

while (fscanf\_s(file, "%lf", &d) == 1) {

p = new LinkedList;

p->value = d;

p->next = ptr;

ptr = p;

}

// Калі былі лічаныя дадзеныя, усталёўваны паказальнік на пачатак спісу

if (p) {

ptr = p;

}

// Правяраем паспяховае закрыццё файла

fclose(file);

cout << "Дадзеныя запісаны ў спіс паспяхова\n";

}

// Функцыя для пошуку элемента ў звязаным спісе

void search(LinkedList\* ptr) {

cout << "Увядзіце лік, якія трэба знайсці: ";

double number, counter = 1;

cin >> number;

// Праходзім па звязаным спісе

while (ptr) {

// Правяраем, ці роўны бягучы элемент шуканаму ліку

if (ptr->value == number) {

// Выводзім інфармацыю аб знойдзеным элеменце

cout << "Лік " << number << " знойдзены ў спісе. Гэта элемент пад нумарам " << counter << endl;

}

ptr = ptr->next; // Пераходзім да наступнага элемента

counter++; // Павялічваем лічыльнік сустрэтых элементаў

}

}

void sum(LinkedList\* ptr) {

double sum = 0; // Пераменная для захоўвання сумы лікаў

double counter = 0; // Лічыльнік для колькасці

// Праходзім па звязаным спісе

while (ptr) {

if (ptr->value > 0) { // Калі лік дадатны, дадаем яго да сумы

sum += ptr->value;

counter++;

}

ptr = ptr->next; // Пераходзім да наступнага вузла

}

// Выводзім вынік

if (sum != 0) {

cout << "Сярэдняе датных лікаў: " << sum / counter; // вылічаем сярэдняем

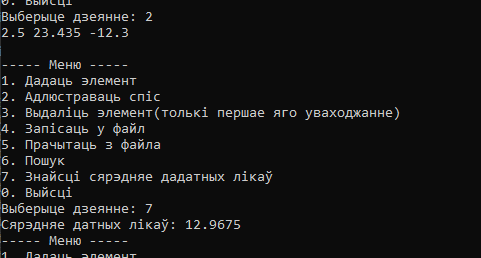
}

else {

cout << "Дадатных лікаў няма";

}

}



**Варыянт 8**



#include <Windows.h>

#include <iostream>

using namespace std;

// Структура для прадстаўлення вузла звязанага спісу

struct LinkedList {

int value;

LinkedList\* next;

};

// Прататыпы функцый

void insert(LinkedList\*& ptrNext);

void display(LinkedList\* ptr);

void deleteListItem(LinkedList\*& ptr);

void writeToFile(LinkedList\* ptr, const char\* fileName);

void readFromFile(LinkedList\*& ptr, const char\* fileName);

void search(LinkedList\* ptr);

void sum5(LinkedList\* ptr);

int main() {

SetConsoleOutputCP(1251);

LinkedList\* ptr = nullptr; // Указатель на текущий узел

int choice;

do {

// Вывад меню

cout << "\n----- Меню -----\n";

cout << "1. Дадаць элемент\n";

cout << "2. Адлюстраваць спіс\n";

cout << "3. Выдаліць элемент(толькі першае яго уваходжанне)\n";

cout << "4. Запісаць у файл\n";

cout << "5. Прачытаць з файла\n";

cout << "6. Пошук\n";

cout << "7. Знайсці суму адмоўных лікаў, кратных 2\n";

cout << "0. Выйсці\n";

cout << "Выберыце дзеянне: ";

// Увод выбару карыстальніка

cin >> choice;

switch (choice) {

case 1:

insert(ptr); // Даданне элемента

break;

case 2:

display(ptr); // Адлюстраванне спісу

break;

case 3:

deleteListItem(ptr); // Выдаленне элемента

display(ptr); // Адлюстраванне абноўленага спісу

break;

case 4:

writeToFile(ptr, "list\_data.txt"); // Запіс у файл

break;

case 5:

readFromFile(ptr, "list\_data.txt"); // Чытанне з файла

break;

case 6:

search(ptr); // Пошук

break;

case 7:

sum5(ptr); // Сума адмоўных лікаў, кратных 2

break;

case 0:

cout << "Выхад...";

default:

cout << "Няправільны выбар. Калі ласка, паспрабуйце яшчэ раз.\n";

}

} while (choice != 0);

// Вызваленне памяці, выдзеленай для вузлоў спісу

while (ptr) {

LinkedList\* temp = ptr;

ptr = ptr->next;

delete temp;

}

return 0;

}

// Функцыя для дадання новага элемента ў пачатак звязанага спісу

void insert(LinkedList\*& ptrNext) {

LinkedList\* ptr;

cout << "Увядзіце лік: ";

int num;

cin >> num;

// Ствараем новы вузел і запаўняем яго дадзенымі

ptr = new LinkedList;

ptr->value = num;

ptr->next = ptrNext; // Усталёўваны паказальнік на наступны элемент(перададзены ў функцыю вузел спісу)

ptrNext = ptr; // Усталёўваны паказальнік на новы элемент у пачатак спісу

}

// Функцыя для адлюстравання элементаў звязанага спісу

void display(LinkedList\* ptr) {

if (ptr == nullptr) {

cout << "Спіс пусты\n";

}

while (ptr) {

cout << ptr->value << " "; // Выводзім значэнне бягучага элемента

ptr = ptr->next; // Пераходзім да наступнага элемента

}

cout << '\n';

}

// Функцыя для выдалення элемента з звязанага спісу

void deleteListItem(LinkedList\*& ptr) {

cout << "Увядзіце лік, чкі трэба выдаліць: ";

int num;

cin >> num;

LinkedList\* current = ptr, \* prev = nullptr, \* tmp;

// Шукаем элемент з зададзеным значэннем

while (current != nullptr && current->value != num) {

prev = current;

current = current->next;

}

// Правяраем, ці знойдзены элемент

if (current == nullptr) {

cout << "Няма такога элемента ў спісе\n";

}

else if (prev == nullptr) { // Калі выдаляем першы элемент

tmp = ptr;

ptr = ptr->next;

delete tmp;

}

else { // Выдаляем элемент з сярэдзіны або канца спісу

prev->next = current->next;

delete current;

}

}

// Функцыя для запісу элементаў звязанага спісу ў файл

void writeToFile(LinkedList\* ptr, const char\* fileName) {

if (ptr == nullptr) {

cout << "Спіс пусты, няма чаго запісаць у файл\n";

return;

}

FILE\* file;

if (fopen\_s(&file, fileName, "wt") != 0) {

cout << "Не атрымалася адкрыць файл для запісу\n";

return;

}

// Запісваем элементы спісу ў файл

while (ptr) {

fprintf\_s(file, "%d ", ptr->value);

ptr = ptr->next;

}

fprintf\_s(file, "\n");

fclose(file);

// Правяраем паспяховае закрыццё файла

fclose(file);

cout << "Дадзеныя запісаны ў файл" << fileName << " паспяхова\n";

}

// Функцыя для чытання элементаў з файла і стварэння звязанага спісу

void readFromFile(LinkedList\*& ptr, const char\* fileName) {

LinkedList\* p = nullptr;

int d;

FILE\* file;

// Спрабуем адкрыць файл для чытання

if (fopen\_s(&file, fileName, "r") != 0) {

cout << "Не атрымалася адкрыць файл для чытання\n";

return;

}

// Счытваем лікі з файла і ствараем новыя вузлы спісу

while (fscanf\_s(file, "%d", &d) == 1) {

p = new LinkedList;

p->value = d;

p->next = ptr;

ptr = p;

}

// Калі былі лічаныя дадзеныя, усталёўваны паказальнік на пачатак спісу

if (p) {

ptr = p;

}

// Правяраем паспяховае закрыццё файла

fclose(file);

cout << "Дадзеныя запісаны ў спіс паспяхова\n";

}

// Функцыя для пошуку элемента ў звязаным спісе

void search(LinkedList\* ptr) {

cout << "Увядзіце лік, якія трэба знайсці: ";

int number, counter = 1;

cin >> number;

// Праходзім па звязаным спісе

while (ptr) {

// Правяраем, ці роўны бягучы элемент шуканаму ліку

if (ptr->value == number) {

// Выводзім інфармацыю аб знойдзеным элеменце

cout << "Лік " << number << " знойдзены ў спісе. Гэта элемент пад нумарам " << counter << endl;

}

ptr = ptr->next; // Пераходзім да наступнага элемента

counter++; // Павялічваем лічыльнік сустрэтых элементаў

}

}

void sum5(LinkedList\* ptr) {

int sum5 = 0; // Пераменная для захоўвання сумы лікаў, кратных 2

// Праходзім па звязаным спісе

while (ptr) {

if (ptr->value % 2 == 0 && ptr->value < 0)

sum5 += ptr->value; // Калі лік кратны 2, дадаем яго да сумы

ptr = ptr->next; // Пераходзім да наступнага вузла

}

// Выводзім вынік

if (sum5 != 0) {

cout << "Сума адмоўных лікаў, кратных 2: " << sum5;

}

else {

cout << "Адмноўных лікаў, кратных 2 няма";

}

}

