МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет математики і інформатики

Кафедра теоретичної та прикладної інформатики

Індивідуальна робота № 1

з курсу «Алгоритми і структури даних»
(назва дисципліни)

Виконала: студентка 2 курсу групи мф-21
напряму підготовки (спеціальності)
Комп'ютерні науки
122 Комп'ютерні науки
(шифр і назва напряму підготовки (спеціальності))
Лобанова Д.В.
(прізвище й ініціали студента)

Харків – 2024

Мета програми:
Сформувати два списка з інформацію про книги (Автор, назва, видавництво, рік видання.) Назви книжок можуть повторюватись (мають різне видання, або/чи рік видання). Написати функцію, яка об'єднує два впорядкованих по незростанню (за назвою книги) списки L1 і L2 в один впорядкований по незростанню список, змінюючи відповідним чином посилання в L1 і L2 і присвоїти отриманий список L1.

Вхідні дані:

Вхідні дані вводяться через стандартний ввід (консоль).

1. Кількість книг для кожного списку (L1 та L2).
2. Для кожної книги: автор, назва, видавництво, рік видання.

Вихідні дані:

1. Перший список книг L1, впорядкований за спаданням назви книги.
2. Другий список книг L2, впорядкований за спаданням назви книги.
3. Об'єднаний список L1, що містить всі елементи з обох списків, впорядкований за спаданням назви книги.

Обґрунтування вибору структури даних:
Зв'язаний список обрано для зберігання книг, оскільки:

1. Це дозволяє об'єднувати два списки без необхідності змінювати їх розмір або переміщати елементи, як це потрібно робити в масивах.
2. Легко змінювати посилання на елементи в списках L1 і L2, що дає змогу сформувати об'єднаний список без додаткових операцій копіювання.
3. Кожна книга в структурі має унікальні поля (автор, назва, видавництво, рік видання), що робить використання зв'язаного списку зручним для доступу до полів, створення послідовності книг і швидкого доступу до наступного елемента.

Опис програми

```
#include <iostream>
#include <string>
using namespace std;
```

Структура для зберігання інформації про книгу

```
struct Book {
    string author;
    string title;
    string publisher;
    int year;
    Book* next;

    // Конструктор для ініціалізації книги
    Book(string a, string t, string p, int y) : author(a), title(t), publisher(p), year(y), next(nullptr)
{}
};
```

Перевірка, чи книга вже є у списку

```
bool isBookExists(Book* head, const string& title, const string& author, const string&
publisher, int year) {
   while (head) {
      if (head->title == title && head->author == author && head->publisher == publisher &&
head->year == year) {
         return true; // Книга вже існує
      }
      head = head->next;
   }
   return false; // Книга не знайдена
}
```

Додавання нової книги до списку (впорядкованого за спаданням назви книги)

```
void addBook(Book*& head, string author, string title, string publisher, int year) {
   // Перевіряємо, чи така книга вже є в списку
   if (isBookExists(head, title, author, publisher, year)) {
      cout << "Book with title '" << title << "', author '" << author << "', publisher '" << publisher
<< "' and year " << year << " already exists in the list." << endl;
      return; // Якщо книга вже є, не додаємо
   }

   // Створення нової книги
   Book* newBook = new Book(author, title, publisher, year);

   // Якщо список порожній або нова книга має більшу або рівну назву, вставляємо її на
початок
   if (head == nullptr || head->title <= title) {
      newBook->next = head;
      head = newBook;
   } else {
      // Якщо нова книга має меншу назву, шукаємо місце для вставки
      Book* current = head;
      while (current->next && current->next->title > title) { // Порівнюємо за спаданням
         current = current->next;
      }
      newBook->next = current->next; // Вставляємо нову книгу
      current->next = newBook;
   }
}
```

Об'єднання двох впорядкованих списків за спаданням

```
void mergeLists(Book*& L1, Book*& L2) {
   Book* merged = nullptr; // Порожній список для результату
```

```
    Book** lastPtr = &merged;

    // Об'єднуємо два списки поки є елементи в обох
    while (L1 && L2) {
        // Порівнюємо книги за спаданням назви
        if (L1->title >= L2->title) {
            // Якщо книга з L1 не існує в об'єднаному списку, додаємо її
            if (!isBookExists(merged, L1->title, L1->author, L1->publisher, L1->year)) {
                *lastPtr = L1;
                lastPtr = &((*lastPtr)->next);
            }
            L1 = L1->next; // Переміщаємося до наступної книги в L1
        } else {
            // Якщо книга з L2 не існує в об'єднаному списку, додаємо її
            if (!isBookExists(merged, L2->title, L2->author, L2->publisher, L2->year)) {
                *lastPtr = L2;
                lastPtr = &((*lastPtr)->next);
            }
            L2 = L2->next; // Переміщаємося до наступної книги в L2
        }
    }

    // Додаємо залишок списку L1 якщо він не порожній
    while (L1) {
        if (!isBookExists(merged, L1->title, L1->author, L1->publisher, L1->year)) {
            *lastPtr = L1;
            lastPtr = &((*lastPtr)->next);
        }
        L1 = L1->next;
    }

    // Додаємо залишок списку L2 якщо він не порожній
    while (L2) {
        if (!isBookExists(merged, L2->title, L2->author, L2->publisher, L2->year)) {
            *lastPtr = L2;
            lastPtr = &((*lastPtr)->next);
        }
        L2 = L2->next;
    }

    // Встановлюємо L1 як об'єднаний список
    L1 = merged;
}
```

Введення книг в список

```
void inputBooks(Book*& list) {
    int n;
```

```
int year;
string author, title, publisher, yearStr;

// Введення кількості книг з перевіркою
while (true) {
    cout << "Enter number of books: ";
    getline(cin, yearStr); // Читаємо введення як рядок

    try {
        n = stoi(yearStr); // Перетворюємо рядок на ціле число
        if (n < 0) {
            throw invalid_argument("Number must be a positive integer.");
        }
        break; // Виходимо з циклу, якщо введено правильно
    } catch (const exception& e) {
        cout << "Invalid input. Please enter again." << endl;
    }
}

// Введення книг
for (int i = 0; i < n; ++i) {
    // Введення автора з перевіркою на порожність
    while (true) {
        cout << "Enter author for book " << i + 1 << ": ";
        getline(cin, author);
        if (author.empty()) {
            cout << "Author cannot be empty. Please enter again." << endl;
        } else {
            break;
        }
    }

    // Введення назви книги з перевіркою на порожність
    while (true) {
        cout << "Enter title for book " << i + 1 << ": ";
        getline(cin, title);
        if (title.empty()) {
            cout << "Title cannot be empty. Please enter again." << endl;
        } else {
            break;
        }
    }

    // Введення видавництва з перевіркою на порожність
    while (true) {
        cout << "Enter publisher for book " << i + 1 << ": ";
        getline(cin, publisher);
        if (publisher.empty()) {
```

```
            cout << "Publisher cannot be empty. Please enter again." << endl;
        } else {
            break;
        }
    }

    // Введення року видання з перевіркою
    while (true) {
        cout << "Enter year for book " << i + 1 << ": ";
        getline(cin, yearStr); // Читаємо введення як рядок

        try {
            year = stoi(yearStr); // Перетворюємо рядок на ціле число
            if (year <= 0) {
                throw invalid_argument("Year must be a positive integer.");
            }
            break; // Виходимо з циклу, якщо введено правильно
        } catch (const exception& e) {
            cout << "Invalid year. Please enter again." << endl;
        }
    }

    // Додавання книги в список
    addBook(list, author, title, publisher, year);
  }
}
```

Виведення списку книг

```
void printList(Book* head) {
  if (head == nullptr) {
    cout << "The list is empty." << endl;
  }
  while (head) {
    cout << "Author: " << head->author << ", Title: " << head->title << ", Publisher: " <<
head->publisher << ", Year: " << head->year << endl;
    head = head->next;
  }
}
```

Функція для очищення списку

```
void clearList(Book*& head) {
  while (head) {
    Book* temp = head;
    head = head->next;
    delete temp;
  }
```

```
    head = nullptr;
}
```

Основна функція програми

```
int main() {
    Book* L1 = nullptr;
    Book* L2 = nullptr;

    // Введення книг для списку L1
    cout << "Enter books for list L1:" << endl;
    inputBooks(L1);

    // Введення книг для списку L2
    cout << "\nEnter books for list L2:" << endl;
    inputBooks(L2);

    // Виведення списку L1
    cout << "\nList L1:" << endl;
    printList(L1);

    // Виведення списку L2
    cout << "\nList L2:" << endl;
    printList(L2);

    // Об'єднання списків L1 і L2 в один список L1
    mergeLists(L1, L2);

    // Виведення об'єднаного списку L1
    cout << "\nMerged list L1:" << endl;
    printList(L1);

    // Звільнення пам'яті для списку L1
    clearList(L1);

    return 0;
}
```

Код програми

```
/* Сформувати два списка з інформацію про книги (Автор, назва, видавництво, рік видання.)
* Назви книжок можуть повторюватись (мають різне видання, або/чи рік видання).
* Написати функцію, яка об'єднує два впорядкованих по незростанню (за назвою книги )
* списки L1 і L2 в один впорядкований по незростанню список,
* змінюючи відповідним чином посилання в L1 і L2 і присвоїти отриманий список L1.
* Daryna Lobanova mf-21
*/

#include <iostream>
#include <string>
using namespace std;
```

```cpp
// Структура книга
struct Book {
   string author;
   string title;
   string publisher;
   int year;
   Book* next;

   // Конструктор для ініціалізації книги
   Book(string a, string t, string p, int y) : author(a), title(t), publisher(p), year(y), next(nullptr) {}
};

// Функція для перевірки, чи книга вже є у списку
bool isBookExists(Book* head, const string& title, const string& author, const string& publisher, int year)
{
   while (head) {
       if (head->title == title && head->author == author && head->publisher == publisher && head->year ==
year) {
           return true; // Книга вже існує
       }
       head = head->next;
   }
   return false; // Книга не знайдена
}

// Функція для додавання нової книги до списку (впорядкованого за спаданням назви книги)
void addBook(Book*& head, string author, string title, string publisher, int year) {
   // Перевіряємо, чи така книга вже є в списку
   if (isBookExists(head, title, author, publisher, year)) {
       cout << "Book with title '" << title << "', author '" << author << "', publisher '" << publisher <<
"' and year " << year << " already exists in the list." << endl;
       return; // Якщо книга вже є, не додаємо
   }

   // Створення нової книги
   Book* newBook = new Book(author, title, publisher, year);

   // Якщо список порожній або нова книга має більшу або рівну назву, вставляємо її на початок
   if (head == nullptr || head->title <= title) {
       newBook->next = head;
       head = newBook;
   } else {
       // Якщо нова книга має меншу назву, шукаємо місце для вставки
       Book* current = head;
       while (current->next && current->next->title > title) { // Порівнюємо за спаданням
           current = current->next;
       }
       newBook->next = current->next; // Вставляємо нову книгу
       current->next = newBook;
   }
}

// Функція для об'єднання двох впорядкованих списків за спаданням
void mergeLists(Book*& L1, Book*& L2) {
   Book* merged = nullptr; // Порожній список для результату
   Book** lastPtr = &merged;

   // Об'єднуємо два списки поки є елементи в обох
   while (L1 && L2) {
       // Порівнюємо книги за спаданням назви
       if (L1->title >= L2->title) {
           // Якщо книга з L1 не існує в об'єднаному списку, додаємо її
           if (!isBookExists(merged, L1->title, L1->author, L1->publisher, L1->year)) {
               *lastPtr = L1;
               lastPtr = &((*lastPtr)->next);
           }
           L1 = L1->next; // Переміщаємося до наступної книги в L1
       } else {
           // Якщо книга з L2 не існує в об'єднаному списку, додаємо її
           if (!isBookExists(merged, L2->title, L2->author, L2->publisher, L2->year)) {
               *lastPtr = L2;
               lastPtr = &((*lastPtr)->next);
           }
           L2 = L2->next; // Переміщаємося до наступної книги в L2
       }
   }
```

```
    // Додаємо залишок списку L1 якщо він не порожній
    while (L1) {
        if (!isBookExists(merged, L1->title, L1->author, L1->publisher, L1->year)) {
            *lastPtr = L1;
            lastPtr = &((*lastPtr)->next);
        }
        L1 = L1->next;
    }

    // Додаємо залишок списку L2 якщо він не порожній
    while (L2) {
        if (!isBookExists(merged, L2->title, L2->author, L2->publisher, L2->year)) {
            *lastPtr = L2;
            lastPtr = &((*lastPtr)->next);
        }
        L2 = L2->next;
    }

    // Встановлюємо L1 як об'єднаний список
    L1 = merged;
}

// Функція для введення книг в список
void inputBooks(Book*& list) {
    int n;
    int year;
    string author, title, publisher, yearStr;

    // Введення кількості книг з перевіркою
    while (true) {
        cout << "Enter number of books: ";
        getline(cin, yearStr); // Читаємо введення як рядок

        try {
            n = stoi(yearStr); // Перетворюємо рядок на ціле число
            if (n < 0) {
                throw invalid_argument("Number must be a positive integer.");
            }
            break; // Виходимо з циклу, якщо введено правильно
        } catch (const exception& e) {
            cout << "Invalid input. Please enter again." << endl;
        }
    }

    // Введення книг
    for (int i = 0; i < n; ++i) {
        // Введення автора з перевіркою на порожність
        while (true) {
            cout << "Enter author for book " << i + 1 << ": ";
            getline(cin, author);
            if (author.empty()) {
                cout << "Author cannot be empty. Please enter again." << endl;
            } else {
                break;
            }
        }

        // Введення назви книги з перевіркою на порожність
        while (true) {
            cout << "Enter title for book " << i + 1 << ": ";
            getline(cin, title);
            if (title.empty()) {
                cout << "Title cannot be empty. Please enter again." << endl;
            } else {
                break;
            }
        }

        // Введення видавництва з перевіркою на порожність
        while (true) {
            cout << "Enter publisher for book " << i + 1 << ": ";
            getline(cin, publisher);
            if (publisher.empty()) {
                cout << "Publisher cannot be empty. Please enter again." << endl;
            } else {
```

```
            break;
        }
    }

    // Введення року видання з перевіркою
    while (true) {
        cout << "Enter year for book " << i + 1 << ": ";
        getline(cin, yearStr); // Читаємо введення як рядок

        try {
            year = stoi(yearStr); // Перетворюємо рядок на ціле число
            if (year <= 0) {
                throw invalid_argument("Year must be a positive integer.");
            }
            break; // Виходимо з циклу, якщо введено правильно
        } catch (const exception& e) {
            cout << "Invalid year. Please enter again." << endl;
        }
    }

    // Додавання книги в список
    addBook(list, author, title, publisher, year);
  }
}

// Функція для виведення списку книг
void printList(Book* head) {
  if (head == nullptr) {
      cout << "The list is empty." << endl;
  }
  while (head) {
      cout << "Author: " << head->author << ", Title: " << head->title << ", Publisher: " <<
head->publisher << ", Year: " << head->year << endl;
      head = head->next;
  }
}

// Функція для очищення списку
void clearList(Book*& head) {
  while (head) {
      Book* temp = head;
      head = head->next;
      delete temp;
  }
  head = nullptr;
}

int main() {
  Book* L1 = nullptr;
  Book* L2 = nullptr;

  // Введення книг для списку L1
  cout << "Enter books for list L1:" << endl;
  inputBooks(L1);

  // Введення книг для списку L2
  cout << "\nEnter books for list L2:" << endl;
  inputBooks(L2);

  // Виведення списку L1
  cout << "\nList L1:" << endl;
  printList(L1);

  // Виведення списку L2
  cout << "\nList L2:" << endl;
  printList(L2);

  // Об'єднання списків L1 і L2 в один список L1
  mergeLists(L1, L2);

  // Виведення об'єднаного списку L1
  cout << "\nMerged list L1:" << endl;
  printList(L1);

  // Звільнення пам'яті для списку L1
  clearList(L1);
```

```
    return 0;
}
```

Результати програми

Тест 1. Введення двох списків без повторів

```
Enter books for list L1:
Enter number of books:3
 Enter author for book 1:Someone1
 Enter title for book 1: Algebra
 Enter publisher for book 1:Publisher1
 Enter year for book
1:1999
 Enter author for book 2:Someone2
 Enter title for book 2:Mathe
 Enter publisher for book 2:Publisher2
 Enter year for book 2:2001
 Enter author for
book 3:Someone3
 Enter title for book 3:Mathematik
 Enter publisher for book 3:Publisher3
 Enter year for book 3:2005

Enter books for list L2:
Enter number of books:2
 Enter author for book 1:A1
 Enter title for book 1:Informatik
 Enter publisher for book 1:P1
 Enter year for book
1:1998
 Enter author for book 2:A2
 Enter title for book 2:Uni
 Enter publisher for book 2:P2
 Enter year for book 2:1789
```

```
 List L1:
 Author: Someone3, Title: Mathematik, Publisher: Publisher3, Year: 2005
 Author: Someone2, Title: Mathe, Publisher: Publisher2, Year: 2001
 Author: Someone1, Title:  Algebra, Publisher: Publisher1, Year: 1999

 List L2:
 Author: A2, Title: Uni, Publisher: P2, Year: 1789
 Author: A1, Title: Informatik, Publisher: P1, Year: 1998

 Merged list L1:
 Author: A2, Title: Uni, Publisher: P2, Year: 1789
 Author: Someone3, Title: Mathematik, Publisher: Publisher3, Year: 2005
 Author: A1, Title: Informatik, Publisher: P1, Year: 1998
 Author: Someone1, Title:  Algebra, Publisher: Publisher1, Year: 1999

 Process finished with exit code 0
```

Тест 2. Порожні списки

```
Enter books for list L1:
Enter number of books:0

Enter books for list L2:
Enter number of books:0

List L1:
The list is empty.

List L2:
The list is empty.

Merged list L1:
The list is empty.

Process finished with exit code 0
```

Тест 3. Список L1 порожній, L2 містить книги

```
Enter books for list L1:
Enter number of books:0

Enter books for list L2:
Enter number of books:3
 Enter author for book 1:A1
 Enter title for book 1:Mathe
 Enter publisher for book 1:P1
 Enter year for book
1:2009
 Enter author for book 2:A2
 Enter title for book 2:Dog
 Enter publisher for book 2:P2
 Enter year for book 2:2000
 Enter author for
book 3:A3
 Enter title for book 3:Cat
 Enter publisher for book 3:P3
 Enter year for book 3:1789

List L1:
The list is empty.

List L2:
Author: A1, Title: Mathe, Publisher: P1, Year: 2009
Author: A2, Title: Dog, Publisher: P2, Year: 2000
Author: A3, Title: Cat, Publisher: P3, Year: 1789

Merged list L1:
Author: A1, Title: Mathe, Publisher: P1, Year: 2009
Author: A2, Title: Dog, Publisher: P2, Year: 2000
Author: A3, Title: Cat, Publisher: P3, Year: 1789

Process finished with exit code 0
```

Тест 4. Списки з однаковими книгами (різні роки видання)

```
Enter books for list L1:
Enter number of books:2
 Enter author for book 1:A
 Enter title for book 1:Something
 Enter publisher for book 1:P
 Enter year for book
1:2000
 Enter author for book 2:A2
 Enter title for book 2:Dog
 Enter publisher for book 2:P2
 Enter year for book 2:2010

Enter books for list L2:
Enter number of books:1
 Enter author for book 1:A
 Enter title for book 1:Something
 Enter publisher for book 1:P
 Enter year for book
1:2005

List L1:
Author: A, Title: Something, Publisher: P, Year: 2000
Author: A2, Title: Dog, Publisher: P2, Year: 2010

List L2:
Author: A, Title: Something, Publisher: P, Year: 2005

Merged list L1:
Author: A, Title: Something, Publisher: P, Year: 2000
Author: A, Title: Something, Publisher: P, Year: 2005
Author: A2, Title: Dog, Publisher: P2, Year: 2010

Process finished with exit code 0
```

Тест 5. Дублювання книг в списку L1

```
Enter books for list L1:
Enter number of books:3
 Enter author for book 1:A
 Enter title for book 1:Mathe
 Enter publisher for book 1:P
 Enter year for book
1:2009
 Enter author for book 2:A
 Enter title for book 2:Mathe
 Enter publisher for book 2:P
 Enter year for book 2:2009
 Book with title '
Mathe', author 'A', publisher 'P' and year 2009 already exists in the list.
Enter author for book 3:A3
 Enter title for book 3:Algebra
 Enter publisher for book 3:P3
 Enter year for book 3:1999

Enter books for list L2:
Enter number of books:1
 Enter author for book 1:Willm
 Enter title for book 1:Dog
 Enter publisher for book 1:P0
 Enter year for book
1:1777
```

```
List L1:
Author: A, Title: Mathe, Publisher: P, Year: 2009
Author: A3, Title: Algebra, Publisher: P3, Year: 1999

List L2:
Author: Willm, Title: Dog, Publisher: P0, Year: 1777

Merged list L1:
Author: A, Title: Mathe, Publisher: P, Year: 2009
Author: Willm, Title: Dog, Publisher: P0, Year: 1777
Author: A3, Title: Algebra, Publisher: P3, Year: 1999

Process finished with exit code 0
```

Тест 6. Дублювання книг при об'єднанні

```
Enter books for list L1:
Enter number of books:2
 Enter author for book 1:A
 Enter title for book 1:Mathematik
 Enter publisher for book 1:P
 Enter year for book
1:2024
 Enter author for book 2:B
 Enter title for book 2:Algebra
 Enter publisher for book 2:P000
 Enter year for book 2:1888

Enter books for list L2:
Enter number of books:1
 Enter author for book 1:A
 Enter title for book 1:Mathematik
 Enter publisher for book 1:P
 Enter year for book
1:2024

List L1:
Author: A, Title: Mathematik, Publisher: P, Year: 2024
Author: B, Title: Algebra, Publisher: P000, Year: 1888

List L2:
Author: A, Title: Mathematik, Publisher: P, Year: 2024

Merged list L1:
Author: A, Title: Mathematik, Publisher: P, Year: 2024
Author: B, Title: Algebra, Publisher: P000, Year: 1888

Process finished with exit code 0
```