

# Міністерство освіти і науки України Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського” Факультет інформатики та обчислювальної техніки

# Кафедра інформаційних систем та технологій

Практичне завдання №3

**Технології програмування на C/Embedded**

*«Зв’язаний список»*

# Виконав:

студент групи ІА-33

Мартинюк Ю.Р.

# Київ 2025

**Тема:** Зв’язаний список

**Мета:** Отримати початкові навички роботи з C

**Результат виконання роботи:**

1) посилання на додаток:

<https://github.com/masonabor/C-Embedded/tree/main/pr3>  
2) Скріншот виконання додатку

**Хід роботи:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct string\_node {

char \*str;

struct string\_node \*next;

};

struct list\_node {

struct string\_node \*sublist;

struct list\_node \*next;

};

struct string\_node\* create\_string\_node(const char \*text) {

struct string\_node \*new\_node = malloc(sizeof(\*new\_node));

if (!new\_node) {

fprintf(stderr, "Memory allocation failed\n");

exit(EXIT\_FAILURE);

}

new\_node->str = malloc(strlen(text) + 1);

if (!new\_node->str) {

fprintf(stderr, "Memory allocation failed\n");

exit(EXIT\_FAILURE);

}

strcpy(new\_node->str, text);

new\_node->next = NULL;

return new\_node;

}

void push\_string\_back(struct string\_node \*\*head, const char \*text) {

struct string\_node \*new\_node = create\_string\_node(text);

if (\*head == NULL) {

\*head = new\_node;

return;

}

struct string\_node \*temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = new\_node;

}

void push\_strings\_array(struct string\_node \*\*head, const char \*arr[], int size) {

for (int i = 0; i < size; i++) {

push\_string\_back(head, arr[i]);

}

}

void free\_string\_list(struct string\_node \*head) {

while (head) {

struct string\_node \*next = head->next;

free(head->str);

free(head);

head = next;

}

}

struct list\_node\* create\_list\_node(struct string\_node \*sublist) {

struct list\_node \*new\_node = malloc(sizeof(\*new\_node));

if (!new\_node) {

fprintf(stderr, "Memory allocation failed\n");

exit(EXIT\_FAILURE);і

}

new\_node->sublist = sublist;

new\_node->next = NULL;

return new\_node;

}

void push\_list\_back(struct list\_node \*\*head, struct string\_node \*sublist) {

struct list\_node \*new\_node = create\_list\_node(sublist);

if (\*head == NULL) {

\*head = new\_node;

return;

}

struct list\_node \*temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = new\_node;

}

void push\_lists\_array(struct list\_node \*\*head, struct string\_node \*sublists[], int size) {

for (int i = 0; i < size; i++) {

push\_list\_back(head, sublists[i]);

}

}

void free\_list\_of\_lists(struct list\_node \*head) {

while (head) {

struct list\_node \*next = head->next;

free\_string\_list(head->sublist);

free(head);

head = next;

}

}

void print\_book(const struct list\_node \*head, int number\_of\_book) {

const char \*book\_info[] = {

"Name of book: ",

"Price: ",

"Number of pages: ",

"Language: ",

"Weight: ",

"Year: "

};

int n = sizeof(book\_info) / sizeof(book\_info[0]);

const struct list\_node \*temp = head;

for (int i = 1; i < number\_of\_book; i++) {

if (temp == NULL) {

printf("Error: no such book\n");

return;

}

temp = temp->next;

}

if (temp == NULL) {

printf("Error: no such book\n");

return;

}

const struct string\_node \*sub = temp->sublist;

printf("\n");

for (int j = 0; j < n && sub != NULL; j++) {

printf("%s%s\n", book\_info[j], sub->str);

sub = sub->next;

}

}

int input\_number\_of\_book(const char \*prompt, int number\_of\_books) {

int number;

while (1) {

printf("%s\n", prompt);

if (scanf("%d", &number) != 1) {

while (getchar() != '\n');

printf("Invalid input. Try again.\n");

continue;

}

if (number >= 1 && number <= number\_of\_books) {

return number;

}

printf("Error: this book isn’t real\n");

}

}

int main(void) {

const int number\_of\_information = 6;

const int number\_of\_books = 7;

struct string\_node \*lists[number\_of\_books];

memset(lists, 0, sizeof(lists));

const char \*book1[] = {"Harry Potter and the Philosopher`s Stone", "10$", "223", "Ukrainian", "500", "1997"};

const char \*book2[] = {"Harry Potter and the Chamber of Secrets", "12$", "251", "Ukrainian", "550", "1998"};

const char \*book3[] = {"Harry Potter and the Prisoner of Azkaban", "15$", "317", "Ukrainian", "600", "1999"};

const char \*book4[] = {"Harry Potter and the Goblet of Fire", "18$", "636", "Ukrainian", "1000", "2000"};

const char \*book5[] = {"Harry Potter and the Order of the Phoenix", "20$", "766", "Ukrainian", "1100", "2003"};

const char \*book6[] = {"Harry Potter and the Half-Blood Prince", "20$", "607", "Ukrainian", "950", "2005"};

const char \*book7[] = {"Harry Potter and the Deathly Hallows", "22$", "759", "Ukrainian", "1200", "2007"};

push\_strings\_array(&lists[0], book1, number\_of\_information);

push\_strings\_array(&lists[1], book2, number\_of\_information);

push\_strings\_array(&lists[2], book3, number\_of\_information);

push\_strings\_array(&lists[3], book4, number\_of\_information);

push\_strings\_array(&lists[4], book5, number\_of\_information);

push\_strings\_array(&lists[5], book6, number\_of\_information);

push\_strings\_array(&lists[6], book7, number\_of\_information);

struct list\_node \*big\_list = NULL;

push\_lists\_array(&big\_list, lists, number\_of\_books);

int number = input\_number\_of\_book("Enter number of book about Harry Potter:", number\_of\_books);

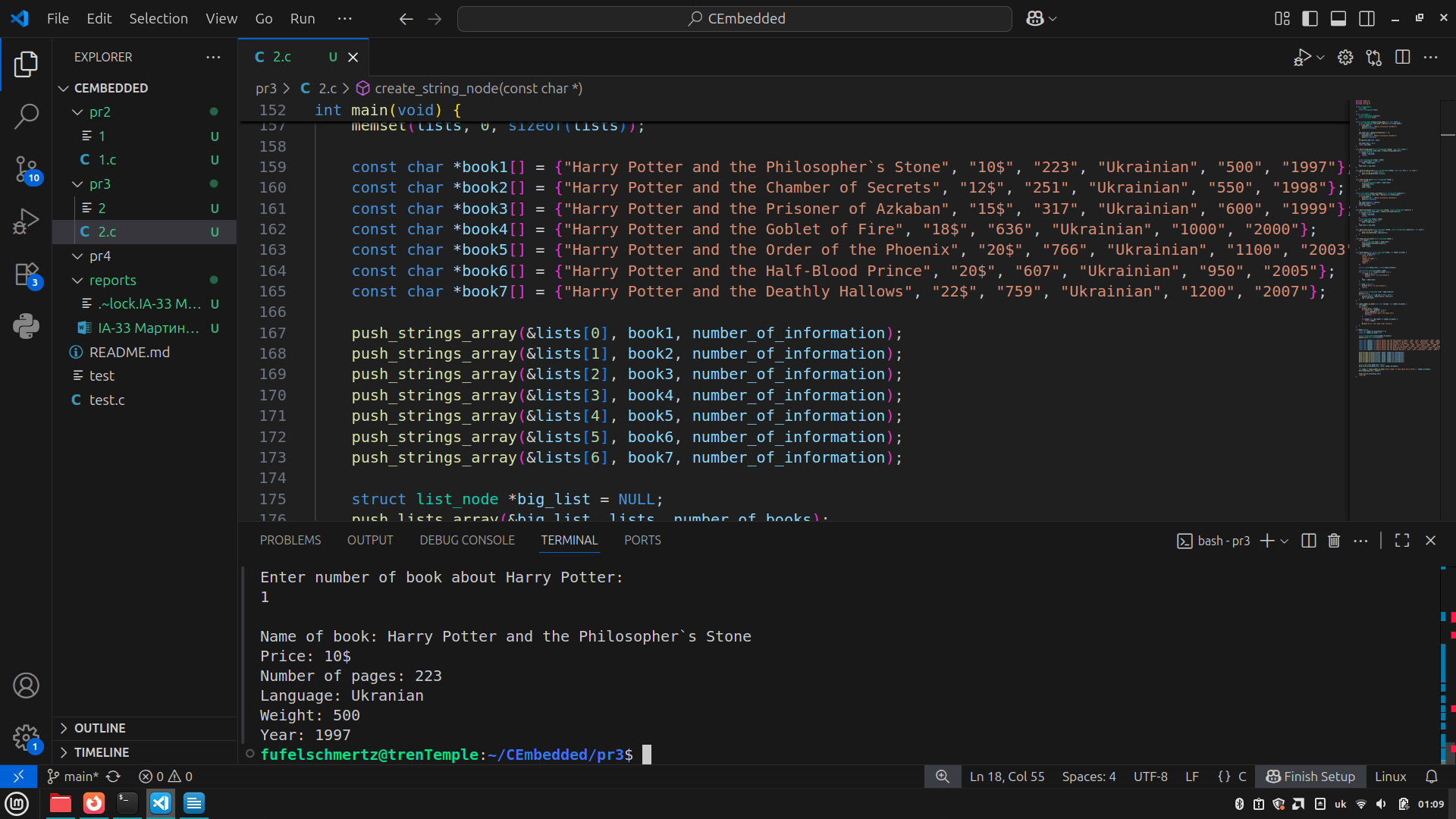
print\_book(big\_list, number);

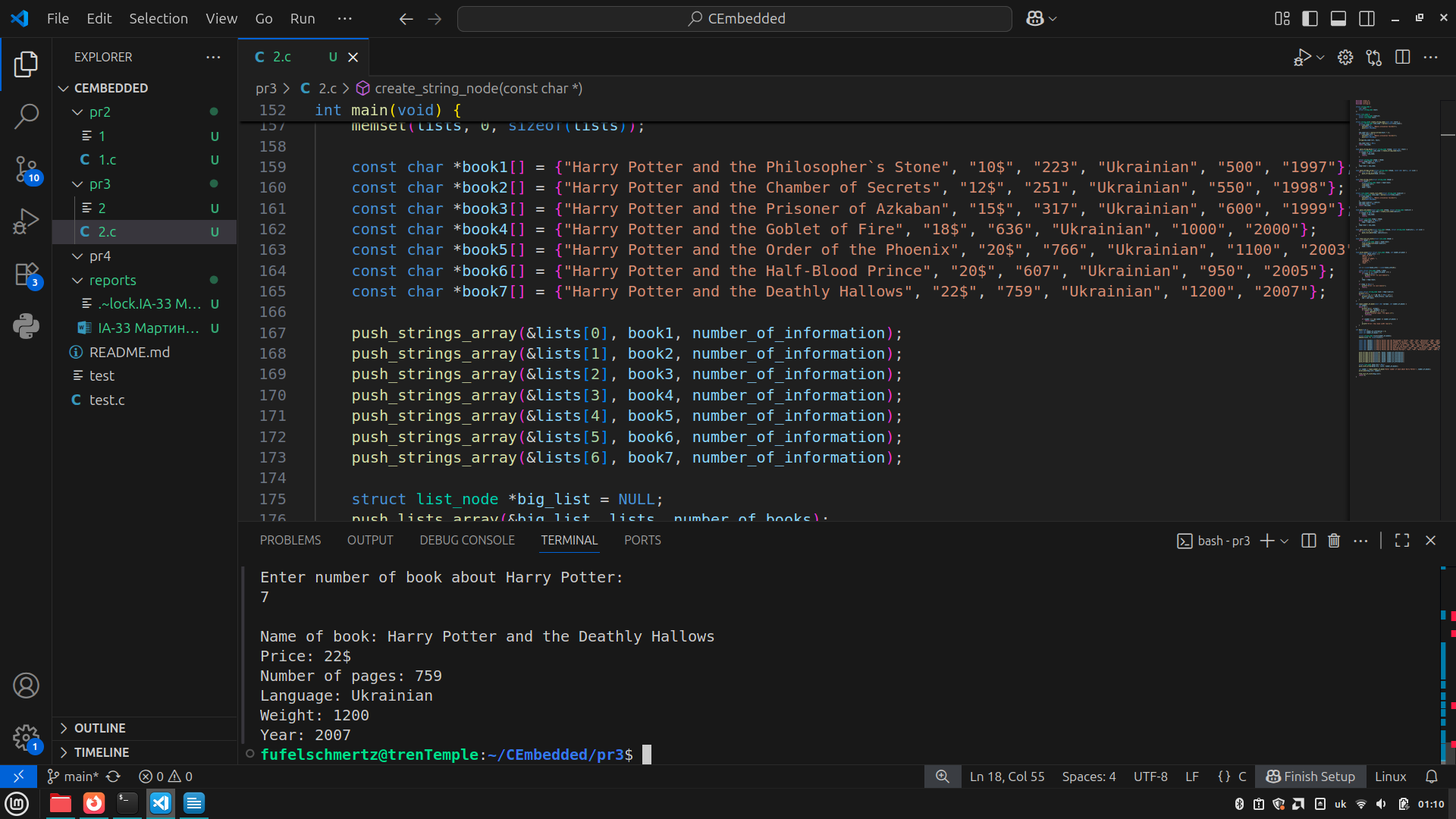
free\_list\_of\_lists(big\_list);

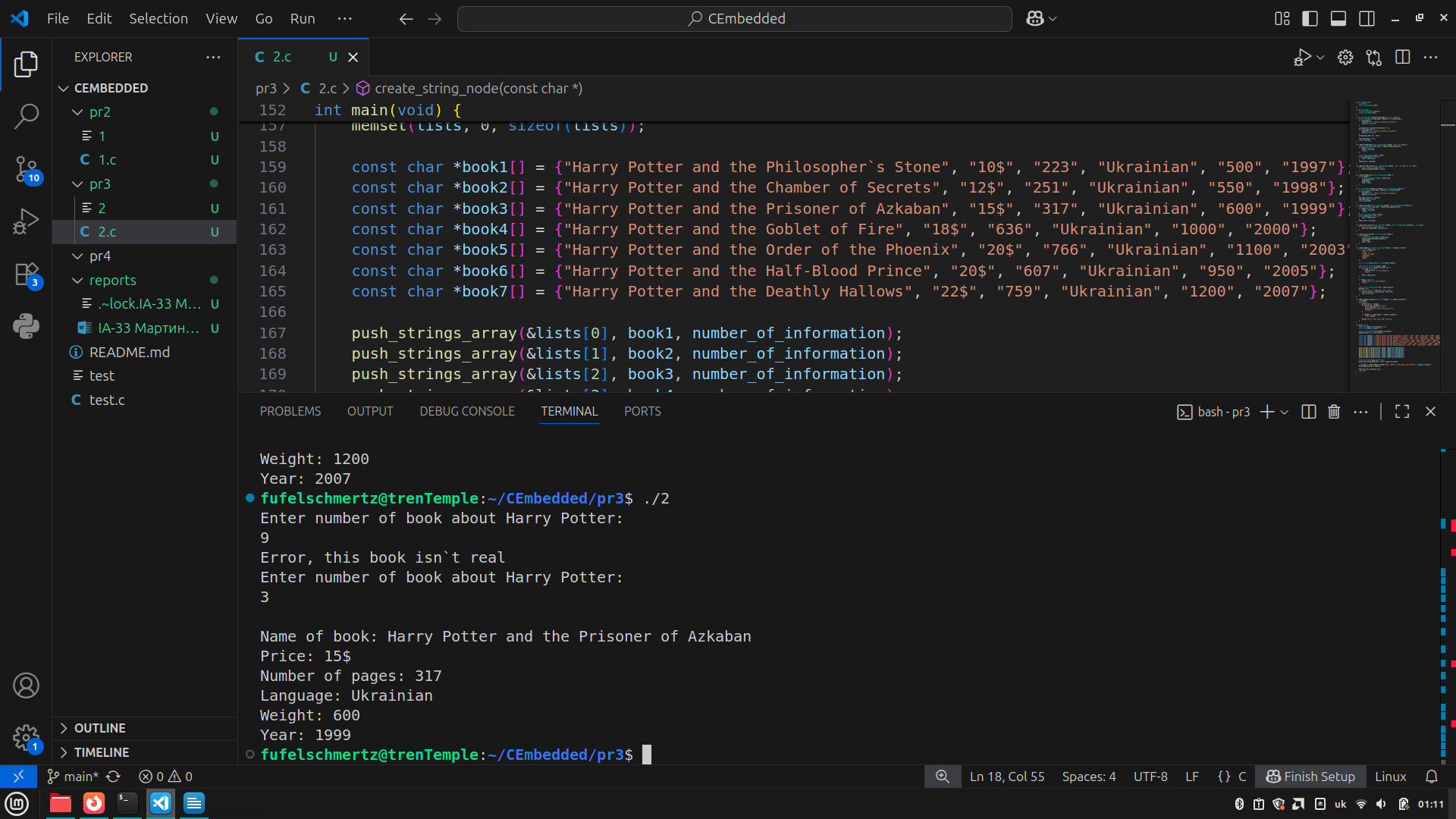
return 0;

}

**Результат виконання:**







**Висновок:** Під час виконання даного практичного завдання я навчився створювати зв’язані списки засобами мови С, попрактикувався в написанні коду та освоїв синтаксис мови С, попрактикувався з виділенням та вивільненням пам’яті.