Лабораторна робота № 3

Базові типи даних та ввід-вивід

Мета роботи.

Ознайомитись з функціями стандартної бібліотеки для введення даних з клавіатури та виведення на екран.

Короткі теоретичні відомості до роботи.

Підключення зовнішніх файлів:

Директива #include використовується для включення копії вказаного файлу в те місце програми, де знаходиться ця директива. Синтаксис :

```
#include "ім'я_файлу"
#include <ім'я_файлу>
```

Різниця між двома формами директиви полягає в методі пошуку препроцесором файлу, що включається. Якщо ім'я файлу розміщене в "кутових" дужках < >, то послідовність пошуку препроцесором заданого файлу в каталогах визначається встановленими каталогами включення (include directories). Якщо ж ім'я файлу заключне в лапки, то препроцесор шукає в першу чергу файл у поточній директорії, а потім вже у каталогах включення. Робота директиви #include зводиться практично до того, що директива #include прибирається, а на її місце заноситься текст із вказаного файлу. Текст файлу, що включається може містити директиви препроцесора, і директиву #include зокрема. Це означає, що директива #include може бути вкладеною. Допустимий рівень вкладеності директиви #include залежить від конкретної реалізації компілятора.

```
#include <stdio.h>      /* приклад 1*/
#include "defs.h"       /* приклад 2*/
```

В першому прикладі у головний файл включається файл з ім'ям stdio.h. Кутові дужки повідомляють компілятору, що пошук файлу необхідно здійснювати в директоріях, вказаних в командному рядку компіляції, а потім в стандартних директоріях.

Оператор printf: вивід на екран

Функцію printf можна використовувати для виведення будь-якої комбінації символів, цілих або дійсних чисел, рядків, беззнакових цілих, довгих та беззнакових довгих цілих.

Приклад:

```
printf("\nAge - %d. Salary $%.2f", age, salary);
```

Передбачається, змінна цілочисельного типу age (вік) та змінна дійсного типу salary (оклад) вже ініційовані якимись значеннями.

Послідовність символів "\n" переводить курсор на новий рядок.

Послідовність символів "Age - %d" буде виведено з початку нового рядка. Символи %d - це специфікація для цілої змінної age. Наступний літерний рядок "Salary $%.2f" - це специфікація (символ перетворення формату) для дійсного значення, а також вказівка формату для виводу лише двох цифр після десяткової крапки. Так виводиться значення змінної salary.

В загальному вигляді специфікація формату складається з обов'язкових та необов'язкових полів та має наступний вигляд:

%[flags] [width] [**.**precision] [{**h** | **l** | **ll** | **I** | **I32** | **I64**}]type

type – обов'язковий параметр який описує тип аргументу.

type	Тип	Інтерпретація
%c	char	символ
%s	рядок	текстовий рядок
%d, %i	int	знакове десятинне ціле значення
%o	unsigned int	без знакове восьмеричне ціле значення
%u	unsigned int	без знакове десятинне ціле значення
%x, %X	unsigned int	без знакове шістнадцятирічне ціле значення з використанням "abcdef" та "ABCDEF" відповідно
%f	float	значення з фіксованою крапкою ([–]*dddd.dddd*)
%e	float	значення з фіксованою крапкою у експоненціальній формі ([–]*d.dddd* e [sign]*dd[d]*)
%g	float	у вигляді f або e в залежності від значення

flags – необов'язковий параметр який визначає вирівнювання при виводі, наявність знаку результату, додавання порожніх символів, восьмеричний та шістнадцятирічний префікси.

width – необов'язковий параметр який визначає мінімальну кількість символів при виведенні значення.

precision – необов'язковий параметр який визначає мінімальну кількість символів після коми.

h| **l**| **ll**| **I**| **I32**| **I64** – визначають залежний від типу розмір аргументу.

Аргумент	Префікс	Значення поля type
long int	l	d, i, o, x, X
long unsigned int	l	o, u, x, X
long long	ll	d, i, o, x, X
short int	h	d, i, o, x, X
short unsigned int	h	o, u, x, X
__int32	I32	d, i, o, x, X
unsigned __int32	I32	o, u, x, X
__int64	I64	d, i, o, x, X
unsigned __int64	I64	o, u, x, X
double	l	f, e, g

Приклад використання функції printf:

```
int a = 10, b = 35;
printf("\na = %d, b = %d", a, b);      // друк на екран значень цілого типу,
                                       // які зберігаються в змінних a та b
char c = 'a';                          // друк на екран символу зі змінної c
printf("\nc = %c", c);
char *str = "Hello world";
printf("\nstr = %s", str);             // друк на екран вмісту рядка str
double sum = 12.347045;
printf("\nsum = %.3lf\n", str);        // друк на екран значення дійсної змінної
```

Оператор scanf: введення з клавіатури

Оператор scanf є одній з багатьох функцій введення значень з клавіатури, наявних в зовнішніх бібліотеках. Кожній змінній, що вводиться, в рядку функції scanf повинна міститися відповідна специфікація:

%[*] [width] [{**h** | **l** | **ll** | **I64** | **L**}]type

type – обов'язковий параметр який описує тип аргументу (див. розділ **printf)**.

width – необов'язковий параметр який визначає кількість символів при виведенні значення.

Перед іменами змінних необхідно помітити символ & (взяти адресу змінної).

Приклад використання функції scanf:

```
#include <stdio.h>
void main()
{
	int weight; /*вага*/
	int height; /*зріст*/
	printf(" Введіть вашу вагу: ");
	scanf("%d", &weight);
	printf(" Введіть ваш зріст: ");
	scanf("%d", &height);
	printf("\n\nВага = %d, Зріст = %d\n", weight, height);
}
```

Введення та виведення за допомогою стандартних потоків cin та cout.

Також для вирішення задачі введення / виведення значень можна використовувати об'єкти cin та cout. cin та cout надають можливість вводу та виводу з використанням вхідного потоку. Для їх використання необхідно підключити файл <iostream> Для введення використовують операцію >>, для виведення – операцію <<. Компілятор визначає тип змінної та відповідним чином образом форматує її.

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

void main()
{
	unsigned int weight = 75;
	cout << "please, input your weight: ";
	cin >> weight;
	cout << endl << "your weight = " << weight;
}
```

Форматування

Для управління форматом значення що вводиться або виводиться використовуються так звані *маніпулятори*. Це функції, які вставляються між значеннями що вводиться або виводиться та змінюють стан потоку.

Для використання маніпуляторів необхідно включити файл <iomanip>.

Декілька маніпуляторів мають параметр, який може бути визначений літералом або змінною. Зміни, зроблені всіма маніпуляторами, крім *setw*, залишаються в силі до відміни. Дія маніпулятора *setw* поширюється тільки на одне значення що вводиться або виводиться.

Маніпулятор	**Опис**	**Примітка**
boolalpha	Значення змінних типа *bool* виводяться як *true* и *false*.	
dec	Цілі значення виводяться в десятинній системі числення.	Використовується

		за замовчуванням
fixed	Для дійсних чисел використовується фіксований формат.	
hex	Цілі значення виводяться в шіснадцятеричній системі числення.	
internal	Знак вирівнюється по лівому краю, а само число – по правому краю.	
left	Вирівнювання по лівому краю.	
noboolalpha	Значення змінних типа *bool* виводяться як *1* и *0*.	Використовується за замовчуванням
noshowbase	Префікси *0* и *0x*, що позначують систему числення, не виводяться.	Використовується за замовчуванням
noshowpoint	Виведення тільки цілої части дійсного числа (без крапки), якщо дробова частина дорівнює 0.	Використовується за замовчуванням
noshowpos	Знак перед додатними числами не виводиться.	Використовується за замовчуванням
noskipws	Пробіл розглядається як признак закінчення введення.	
nouppercase	Шістьнадцятеричі цифри та символ експоненти в науковому форматі дійсного числа виводяться маленькими буквами.	Використовується за замовчуванням
oct	Цілі значення виводяться в восьмеричній системі числення.	
right	Вирівнювання по правому краю.	Використовується за замовчуванням
scientific	Для дійсних чисел використовується науковий формат.	
setfill(*c*)	Визначає символ для заповнення. За замовчуванням використовується пробіл.	
setprecision(*n*)	Визначає точність для дійсних чисел. За замовчуванням точність дорівнює 6. Якщо не встановлено ні фіксований, ні науковий формат дійсного числа, тоді точність визначає кількість цифр що виводяться (всього, до точки та після). Якщо число дуже велике, воно автоматично виводиться в науковому форматі, и тоді точність визначається кількістю цифр в мантисі. Якщо встановлено фіксований формат дійсного числа, точність задає кількість цифр після крапки. Якщо встановлено науковий формат дійсного числа, точність задає кількість цифр в мантисі.	
setw(*n*)	Встановлює мінімальну кількість символів, що використовуються для виведення значення. Якщо значення представляється меншою кількістю символів, інші позиції заповнюються символом, встановленим за допомогою маніпулятора *setfill*. Вирівнювання визначається маніпуляторами *left*, *right* и *internal*. Для того що встановити поведінку по замовчуванням, потрібно використовуватиманіпулятор *setw* с параметром *0*.	**Впливає тільки на одне значення що виводиться**
showbase	Виведення префіксів *0* и *0x* для позначення системи числення.	
showpoint	Виведення и цілої та дробової частин дійсного числа, навіть якщо дробова частина дорівнює 0.	
showpos	Виведення знака перед додатнім числом.	
skipws	Пробіли розглядаються як розділювачі між значеннями.	Використовується за замовчуванням

uppercase	Шістнадцятеричні цифри та символ експоненти в науковому форматі дійсного числа виводяться маленькими буквами.	

Робота з файлами

- Оголошення файлової змінної -- `FILE* file;`
- Відкриття файлу -- `FILE* fopen (char *name, char *mode);`
- Перевірка досягнення кінця файлу -- `int feof (FILE *file);`
- Закриття файлу -- `int fclose (FILE *file);`

FILE - спеціальна структура, оголошена у файлі <stdio.h>, яка використовується при роботі з файлами. Для роботи з файлами потрібно оголосити змінну FILE * <ім'я>.

Параметр *mode* задає потрібний тип доступу до файлу.

Mode	Дія
"r"	Відкриття для читання. Якщо файл не існує, або не може бути знайдений, функція *fopen* повертає ознаку помилки.
"w"	Відкриття файлу для запису. Якщо файл існує, його вміст знищується. Якщо файл не існує, він створюється.
"a"	Відкриття для додавання. Якщо файл не існує, він створюється.
"r+"	Відкриття для читання і запису. Файл повинен існувати.
"w+"	Відкриття порожнього файлу для читання і запису. Якщо файл існує, його вміст знищується.
"a+"	Відкриття для читання і додавання. Якщо файл не існує, він створюється.

Крім того, до параметрі *mode* можуть бути додані символи **t** і **b** для завдання текстового і двійкові режимів відповідно. За замовчуванням використовується текстовий режим.

У випадку помилки функція *fopen* повертає **NULL.**

Текстовий режим

При введенні / виведення в текстовому режимі відбувається перетворення між зовнішнім представленням значення і внутрішнім (машинним) представленням цього значення.

- Введення одного символу -- `int getc(FILE *file);`
- Вивід одного символу -- `int putc(int c, FILE *file);`
- Введення -- `int fscanf(FILE *file, char *format, ...);`
- Вивід -- `int fprintf(FILE *file, char *format, ...);`
- Введення рядка -- `char *fgets(char *line, int maxline, FILE *file);`

- Виведення рядка -- `int fputs(char *line, FILE *file);`

Двійковий режим

У двійковому режимі ніяких перетворень не проводиться, внутрішнє уявлення значення записується у файл. Для відкриття файлу в двійковому режимі необхідно в параметр *mode* функції *fopen* додати символ **b.**

- Введення з файлу `unsigned fread(void*buf, int bytes, int num, FILE *file);`
- Вивід в файл `unsigned fwrite(void*buf, int bytes, int num, FILE *file);`

Функція *fread* читає з файлу *file* у змінну *buf num* елементів, кожен розміром *bytes* байт. Функція *fwrite* записує у файл *file* із змінною *buf num* елементів, кожен розміром *bytes* байт.

Функції повертають кількість прочитаних / записаних елементів.

У двійковому режимі можливий прямий доступ до файлу:

`int fseek(FILE *file, long nbytes, int origin)`

Ця функція смещает покажчик у файлі *file* на *nbytes* байт з позиції, що визначається параметром *origin.* При цьому параметр *origin* може приймати наступні значення:

- SEEK_SET 0 - початок файлу;
- SEEK_CUR 1 - поточна позиція покажчика;
- SEEK_END 2 - кінець файлу.

`long ftell(FILE *file)` – повертає поточну позицію покажчика у файлі *file.*

```
// Визначення розміру файлу
fseek(file, 0, SEEK_END);
n = ftell(file);
```

Файлові потоки

Для введення / виводу з та у файл існують потоки, які можуть бути пов'язані з файлом на диску. Для використання файлових потоків необхідно включити заголовки <fstream>. Існує три різновиди файлових потоків: *fstream, ifstream* і *ofstream.* Різниця між ними полягає в тому, що потік *fstream* за замовчуванням відкривається для введення та виведення, потік *ifstream* за замовчуванням відкривається для введення, а потік *ofstream* за замовчуванням відкривається для виводу. Змінити поведінку за замовчуванням, а також задати інші режими відкриття файлу можна за допомогою наступних констант:

- ios_base:: app - відкриття файлу для додавання;
- ios_base:: binary - відкриття двійкового, а не текстового файлу;
- ios_base:: in - відкриття файлу для читання;
- ios_base:: out - відкриття файлу для запису;
- ios_base:: trunc - видалення вмісту файлу при відкритті.

Режими відкриття файлу комбінуються з допомогою операції поразрядного АБО(**|** **).**

Для відкриття файлу можна задати ім'я файлу безпосередньо в конструкторі потоку або скористатися функцією *open.*

```
fstream fs("f1.txt");         // Відкриття файлу для читання та запису
ifstream ifs("f2.txt");       // Відкриття файлу для читання
ofstream ofs("f3.txt");       // Відкриття файлу для запису
// Відкриття файлу для читання і запису з видаленням вмісту файлу
fstream fs("f1.txt", ios_base:: in | ios_base:: out | ios_base:: trunk);
// Зображення двійкового файлу для читання
ifstream ifs("f2.txt", ios_base:: in | ios_base:: binary);
ofstream ofs;                   // Створюємо потік, не пов'язаний з файлом
ofs.open("f3.txt");           // Відкриваємо файл для запису
```

Якщо планується використання файлу тільки для читання, або тільки для запису безпечніше скористатися відповідним файловим потоком.

Для перевірки того, чи відкрито файл служить функція *is_open.*

Потоки автоматично закриваються при завершенні програми. Однак при необхідності можна закрити потік функцією *close* і потім знову відкрити його, зв'язавши з іншим файлом.

Для роботи з текстовими потоками використовуються операції “**<<**” та “**>>”.** Також можливе використання маніпуляторів для форматування вводять / виводяться значень.

```
int x;
fstream f;
// Відкриваємо файл для читання
f.open( "in.txt", ios_base::in);
// Перевіряємо відкриття файлу
if(!f.is_open())
{
      cout << "Неможливо відкрити файл 'in.txt' \n";
      return;
}
f >> x;          // Читання змінної x з файлу
if(f.fail())    // Перевірка помилок читання
{
      cout << "Помилка читання з файлу 'in.txt' \n";
      return;
}
f.close();       // Закриваємо файл
// Знову відкриваємо файл, тепер для запису
f.open( "out.txt", ios_base::out);
// Перевіряємо відкриття файлу
if(!f.is_open())
{
      cout << "Неможливо відкрити файл 'out.txt' \n";
      return;
}
// Виводимо значення змінної x в 16-ричного системі
f << hex << x << endl;
f.close();   // Закриваємо файл
```

Для того, щоб перевірити, чи досягнуто кінець файлу, використовується функція *eof.*

```
int n;
```

```
ifstream f("in.txt");
if(!f.is_open())
{
	cout << "Неможливо відкрити файл 'in.txt' \n";
	return;
}
while(!f.eof())    // Поки не досягнуто кінець файлу
{
	F >> n;
	cout << n << endl;
}
```

Для організації прямого доступу до файлу використовуються функції *seekg / seekp* і *tellg / tellp.* Різниця між функціями полягає в тому, що функції, з ім'ям, закінчуються символом 'g', використовуються для роботи з потоками введення, а функції, з ім'ям, закінчуються символом 'p', - для роботи з потоками виводу.

Функції *seekg / seekp* переміщають внутрішній покажчик файлу на задану позицію. Позиції у байтах, нумерація починається з 0. Існує два різновиди функцій - з одним параметром і з двома параметрами. Один цілочисельний параметр задає абсолютну позицію в файлі. Два параметри задають зсув (ціле число) і точку відліку. Точка відліку може приймати наступні значення:

- ios_base:: beg - початок файлу;
- ios_base:: cur - поточна позиція покажчика;
- ios_base:: end - кінець файлу.

Функції *tellg / tellp* не мають параметрів. Вони повертають поточну позицію покажчика у файлі.

Функції *seekg / seekp* і *tellg / tellp* працюють як з текстовими, так і з двійковими потоками. У будь-якому випадку бажано або знати структуру файлу, або працювати з файлами, всі записи у яких мають однакову довжину. В іншому випадку можливе переміщення покажчика на позицію, яка не є початком запису.

Порядок виконання роботи.

1. Створити проект що містить консольну програму Win32.
2. Вивести на екран ПІБ, назву групи та номер варіанту.
3. Зчитати значення з клавіатури вказані у варіантах до роботи.
4. Вивести всі зчитані дані у текстовий рядок за допомогою функції sprintf (з параметрами функції sprintf розібратися самостійно).
5. Вивести сформований рядок на екран обов’язково використовуючи восьмеричне та шіснадцятеричне представлення для цілих чисел та різні формати для дійсних значень.
6. Виконати пункт № 6 першої лабораторної роботи з виведенням на екран всіх значень послідовності та результату обчислень (завдання на оцінку задовільно).
7. Записати до файлу рядок сформований в пункті 4, та зчитати його з файлу в інший рядок (завдання на оцінку добре).

8. Записати до файлу значення змінних зчитані при виконанні пункту 3, та зчитати їх з файлу в інші змінні (завдання на оцінку відмінно).

УВАГА:

Обов’язковими для виконання є пункти які не помічено відмітками: «для отримання оцінки задовільно», «для отримання оцінки добре» та «для отримання оцінки відмінно».

Оцінка задовільно ставиться у випадку виконання всіх обов’язкових пунктів роботи та пункту який має відмітку «для отримання оцінки задовільно».

Оцінка добре ставиться у випадку виконання всіх обов’язкових пунктів роботи та пунктів які мають відмітку «для отримання оцінки задовільно» та «для отримання оцінки добре».

Оцінка відмінно ставиться у випадку виконання всіх обов’язкових пунктів роботи та пунктів які мають відмітку «для отримання оцінки задовільно», «для отримання оцінки добре» та «для отримання оцінки відмінно».

Варіанти до роботи.

Таблиця 1. Варіанти до пункту № 3.

	char	int	unsigned int	short	float	double	long	string
1.	√	√	√		√	√		√
2.	√	√		√	√		√	√
3.	√	√	√	√	√			√
4.	√	√		√	√		√	√
5.	√	√	√	√	√	√		√
6.	√	√			√	√	√	√
7.	√	√	√		√		√	√
8.	√	√		√	√			√
9.	√	√			√		√	√
10.	√	√	√	√	√	√		√
11.	√	√		√	√	√		√
12.	√	√	√		√		√	√
13.	√	√	√		√			√
14.	√	√	√	√	√		√	√
15.	√	√		√	√	√	√	√
16.	√	√	√		√		√	√
17.	√	√		√	√	√		√
18.	√	√			√			√
19.	√	√		√	√	√	√	√
20.	√	√			√	√	√	√
21.	√	√	√	√	√			√
22.	√	√	√		√	√	√	√
23.	√	√	√	√	√	√		√
24.	√	√			√		√	√
25.	√	√		√	√		√	√
26.	√	√	√		√			√
27.	√	√		√	√	√	√	√
28.	√	√			√	√	√	√
29.	√	√		√	√		√	√
30.	√	√	√		√			√