



# СТРУКТУРИ ТА ІНШІ ПОХІДНІ ТИПИ

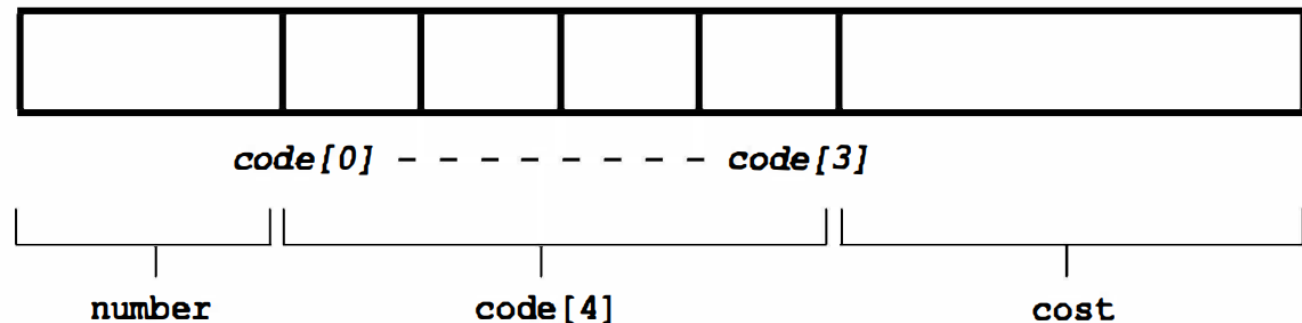
Питання 4.3.

# Похідний тип: структура

---

- *Змінні типу структури* поєднують різнотипні значення, представлені у вигляді *елементів (полів) структури*.
- Оголошення структури відбувається за допомогою ключового слова **struct**:

```
struct stuff {  
    int number;  
    char code[4];  
    float cost;  
};
```



- Описується структура, утворена з цілочисельної змінної, символьного масиву та змінної типу float.
- Такий опис не створює фактичного об'єкту даних, проте показує складові частини такого об'єкту.
- Дескриптор stuff не є обов'язковим, буде використовуватись оголошення: **struct stuff library;**
- Змінна library оголошена з типом структури, який використовує схему структури stuff.

# Демонстраційний приклад (функція виводу)

```
#include <stdio.h>
#include <string.h>

#define MAXTITL 41 /* maximum length of title + 1 */
#define MAXAUTL 31 /* maximum length of author's name + 1 */

struct book { /* structure template: tag is book */
    char title[MAXTITL];
    char author[MAXAUTL];
    float value;
}; /* end of structure template */
```

```
Please enter the book title.
C Programming Language
Now enter the author.
Stephen Prata
Now enter the value.
39.99
C Programming Language by Stephen Prata: $39.99
Stephen Prata: "C Programming Language" ($39.99)
Done.
```

```
int main(void)
{
    struct book library; /* declare library as a book variable */

    printf("Please enter the book title.\n");
    s_gets(library.title, MAXTITL); /* access to the title portion */
    printf("Now enter the author.\n");
    s_gets(library.author, MAXAUTL);
    printf("Now enter the value.\n");
    scanf("%f", &library.value);
    printf("%s by %s: $%.2f\n", library.title,
                                                library.author, library.value);
    printf("%s: \"%s\" ($%.2f)\n", library.author,
                                                library.title, library.value);

    printf("Done.\n");

    return 0;
}
```

# Коментарі до програми

---

```
char * s_gets(char * st, int n)
{
    char * ret_val;
    char * find;

    ret_val = fgets(st, n, stdin);
    if (ret_val)
    {
        find = strchr(st, '\n'); // look for newline
        if (find)                // if the address is not NULL,
            *find = '\0';        // place a null character there
        else
            while (getchar() != '\n')
                continue;        // dispose of rest of line
    }
    return ret_val;
}
```

- Для доступу до окремих елементів структури слугує операція крапки (.), або операція належності структурі.
  - Наприклад, `library.value` - це елемент `value` структури `library`.
  - Аналогічно, конструкція `library.title` використовується точно так же, як масив значень типу `char`.

# Оголошення структури

---

- При оголошенні змінної типу структури шаблон `struct book` відіграє ту ж роль, що і слова `int` або `float` у більш простих оголошеннях.
  - `struct book doyle, panshin, *ptbook ;`
  - кожна змінна `doyle` і `panshin` типу структури буде мати три частини: `title`, `author`, `value`.
  - Оголошення структури `book` створює новий тип з назвою `struct book`.
- З точки зору комп'ютера оголошення **`struct book library;`**
- є скороченим варіантом оголошення

```
struct {  
    char title [MAXTITL] ;  
    char author [MAXAUTL] ;  
    float value;  
} library;
```

# Структури і typedef

---

- Ключове слово `typedef` використовується для надавання альтернативної назви для типу.
  - Наприклад, `typedef unsigned char BYTE;`
  - Доступні оголошення виду `BYTE b1, b2;`
- Можна використовувати для визначення імен користувацьких типів даних.
  - Дескриптор буде потрібен, якщо шаблон структури буде застосовуватись кілька разів або використовується альтернативний варіант оголошення за допомогою `typedef`

```
typedef struct Books {  
    char title[50];  
    char author[50];  
    char subject[100];  
    int book_id;  
} Book;
```

```
Book book;
```

# Ініціалізація структур

---

- Використовується синтаксис, аналогічний до ініціалізації масивів:

```
struct book library = {  
    "The Pirate and the Devious Damsel",  
    "Renee Vivotte",  
    1.95  
};
```

- У стандарті C99 передбачені виділені ініціалізатори структур (порядок елементів не важливий)

```
struct book gift = {  
    .value = 25.99,  
    .author = "James Broadfool",  
    .title = "Rue for the Toad" } ;
```

# Масиви структур (на прикладі бібліотеки книг)

---

```
int main(void)
{
    struct book library[MAXBKS]; /* array of book structures */
    int count = 0;
    int index;

    printf("Please enter the book title.\n");
    printf("Press [enter] at the start of a line to stop.\n");
    while (count < MAXBKS && s_gets(library[count].title, MAXTITL) != NULL && library[count].title[0] != '\0') {
        printf("Now enter the author.\n");
        s_gets(library[count].author, MAXAUTL);
        printf("Now enter the value.\n");
        scanf("%f", &library[count++].value);
        while (getchar() != '\n')
            continue; /* clear input line */
        if (count < MAXBKS)
            printf("Enter the next title.\n");
    }
```

*Вураз `s_gets(library[count].title, MAXTITL)` зчитує рядок назви книги;  
Приймає значення `NULL`, якщо функція `s_gets()` намагається прочитати символ, наступний після маркера кінця файлу.*

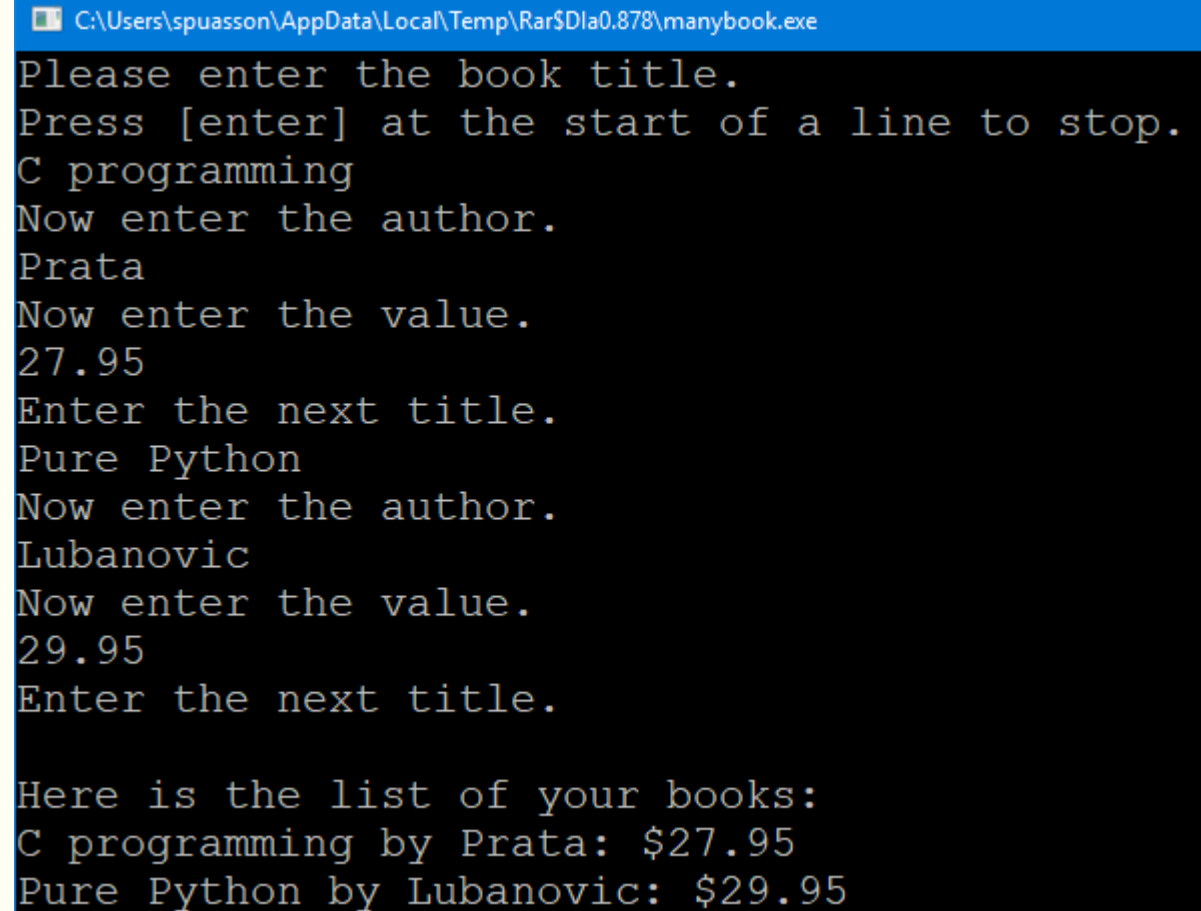


# Продовження лістингу та вивід програми

---

```
if (count > 0)
{
    printf("Here is the list of your books:\n");
    for (index = 0; index < count; index++)
        printf("%s by %s: $%.2f\n", library[index].title,
            library[index].author, library[index].value);
}
else
    printf("No books? Too bad.\n");

return 0;
}
```

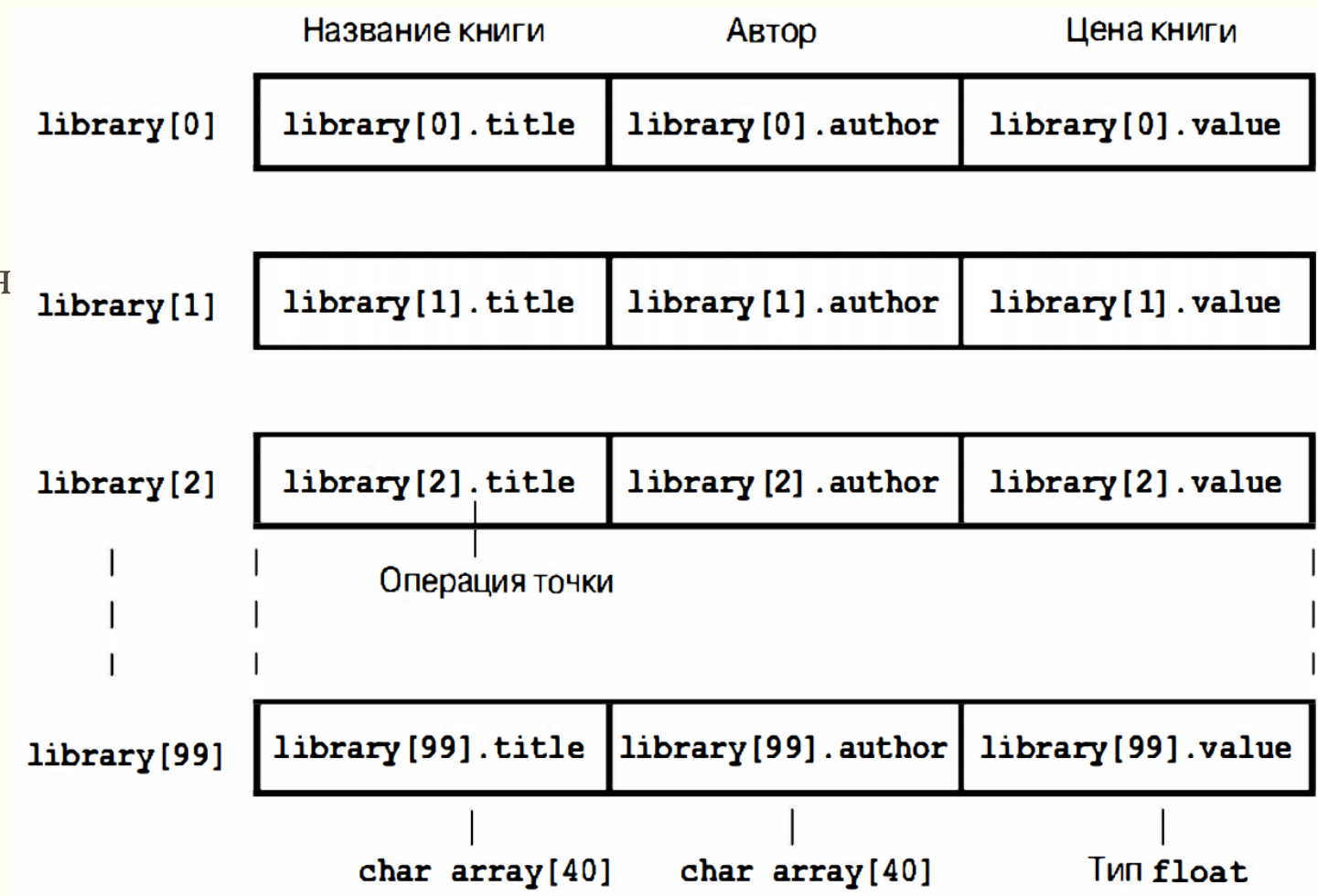


```
C:\Users\spuasson\AppData\Local\Temp\Rar$Dla0.878\manybook.exe
Please enter the book title.
Press [enter] at the start of a line to stop.
C programming
Now enter the author.
Prata
Now enter the value.
27.95
Enter the next title.
Pure Python
Now enter the author.
Lubanovic
Now enter the value.
29.95
Enter the next title.

Here is the list of your books:
C programming by Prata: $27.95
Pure Python by Lubanovic: $29.95
```

# Оголошення масиву структур та ідентифікація його елементів

- У прикладі:
  - `struct book library [MAXBKS];`
- Для ідентифікації елементів масиву структур застосовуються ті ж правила, що й до індивідуальних структур:
  - `library[0].value`
  - `library[4].title`
  - `library[2].title[4]`



# Вкладені структури (одна структура містить іншу всередині)

---

```
// friend.c -- пример вложенной структуры
#include <stdio.h>
#define LEN 20
const char * msgs[5] =
{
    "    Благодарю вас за чудесно проведенный вечер, ",
    "Вы однозначно продемонстрировали, что ",
    "являет собою особый тип мужчины. Мы обязательно должны встретиться",
    "за восхитительным ужином с ",
    "и весело провести время."
};

struct names {                // первая структура
    char first[LEN];
    char last[LEN];
};

struct guy {                  // вторая структура
    struct names handle;      // вложенная структура
    char favfood[LEN];
    char job[LEN];
    float income;
};
```

Нехай Шейла Піроскі створює структуру з інформацією про своїх друзів.

- Одним з членів структури є ім'я друга, представлене структурою з окремими записами для імені та прізвища.

Вкладена структура оголошується так, як і звичайна змінна типу int:

- struct names handle;

# Продовження коду

```
int main(void)
{
    struct guy fellow = {
        { "Билли", "Бонс" },
        "запечеными омарами",
        "персональный тренер",
        68112.00
    };
    printf("Дорогой %s, \n\n", fellow.handle.first);
    printf("%s%s.\n", msgs[0], fellow.handle.first);
    printf("%s%s\n", msgs[1], fellow.job);
    printf("%s\n", msgs[2]);
    printf("%s%s%s", msgs[3], fellow.favfood, msgs[4]);
    if (fellow.income > 150000.0)
        puts("!!");
    else if (fellow.income > 75000.0)
        puts("!");
    else
        puts(".");
    printf("\n%40s%s\n ", " ", "До скорой встречи,");
    printf("%40s%s\n", " ", "Шейла");

    return 0;
}
```

- Отримання доступу до члена вкладеної структури:
  - `printf("Дорогой %s!\n\n", fellow.handle.first);`
- Інтерпретація наступна: `(fellow.handle).first`

Дорогой Билли,

Благодарю вас за чудесно проведенный вечер, Билли.  
Вы однозначно продемонстрировали, что персональный тренер  
являет собою особый тип мужчины. Мы обязательно должны встретиться  
за восхитительным ужином с запеченными омарами и весело провести время.  
До скорой встречи,  
Шейла

# Вказівники на структури

---

```
#include <stdio.h>
#define LEN 20

struct names {
    char first[LEN];
    char last[LEN];
};

struct guy {
    struct names handle;
    char favfood[LEN];
    char job[LEN];
    float income;
};
```

- Причини використання вказівників у структурах:
  - 1) простіше маніпулювати вказівниками на масиви простіше, ніж самими масивами (наприклад, в задачі сортування), із вказівниками на структури часто працювати легше, ніж із самими структурами.
  - 2) у деяких старих реалізаціях структура не може передаватись як аргумент функції, проте вказівник на структуру — може.
  - 3) незважаючи на те, що структуру можна передавати в якості аргумента, передача вказівника часто ефективніша.
  - 4) багато складних представлень даних застосовують структури, які містять вказівники на інші структури.

```
int main(void)
{
```

## Продовження коду

```
    struct guy fellow[2] = {
        ({ "Билли", "Бонс" },
         "запеченными омарами",
         "персональный тренер",
         68112.00
        ),
        ({ "Джим", "Хокинс" },
         "рыбным фрикасе",
         "редактор таблоида",
         232400.00
        )
    };
```

```
    struct guy * him;    /* указатель на структуру */
    printf("адрес #1: %p #2: %p\n", &fellow[0], &fellow[1]);
    him = &fellow[0];    /* сообщает указателю, на что указывать */
    printf("указатель #1: %p #2: %p\n", him, him + 1);
    printf("him->income равно $%.2f: (*him).income равно $%.2f\n",
           him->income, (*him).income);
    him++;                /* указатель на следующую структуру */
    printf("him->favfood равно %s: him->handle.last равно %s\n",
           him->favfood, him->handle.last);

    return 0;
}
```

```
адрес #1: 0x7fff5fbff820 #2: 0x7fff5fbff874
указатель #1: 0x7fff5fbff820 #2: 0x7fff5fbff874
him->income равно $68112.00: (*him).income равно $68112.00
him->favfood равно рыбным фрикасе: him->handle.last равно Стюарт
```

- Оголошення не призводить до створення нової структури, проте вказівник him тепер може посилатись на будь-яку існуючу структуру типу guy.
  - Наприклад, якщо barney — структура типу guy, то можна написати оператор:
  - him = &barney;
  - Назва структури не є вказівником, на відміну від масивів.
- Додавання 1 до him призводить до додавання 84 (байта) до адреси.
  - $874_{16} - 820_{16} = 54_{16} = 84_{10}$ :
  - names.first – 20 байтів, names.last – 20 байтів,
  - favfood — 20 байтів, job — 20 байтів
  - income — 4 байта.

# Доступ до елементів структури через вказівники

---

- Як за допомогою `him` отримати значення члену `fellow[0]`?
- **Метод 1. Застосування операції ->**
  - `him->income` дорівнює `barney.income`, якщо `him == &barney`
  - `him->income` дорівнює `fellow[0].income`, якщо `him == &fellow[0]`
- **Метод 2. `fellow[0].income == (*him).income`**
  - Дужки обов'язкові, оскільки операція «.» пріоритетніша, ніж «\*».

## Додаткові можливості структур

---

- Сучасна мова С дозволяє присвоїти одну структуру іншій (що неможливо з масивами).
  - Працює, навіть якщо член виявляється масивом.
  - Також структуру можна ініціалізувати іншою структурою того ж типу:
  - `struct names right_field = {"Джеймс", "Бонд"};`
  - `struct names captain = right_field;`      *//ініціалізація структури іншою структурою*
- Вказівники на структури також допускають двусторонній обмін даними.



# Об'єднання (union)

## Structure

```
struct Employee {  
    int id;           // size 1 byte  
    char name;        // size 2 byte  
    float salary;     // size 4 byte  
} e1; // size of e1 = 7 byte
```

## Union

```
union Employee {  
    int id;           // size 1 byte  
    char name;        // size 2 byte  
    float salary;     // size 4 byte  
} e1; // size of e1 = 4 byte
```

- Нехай проводиться опитування.
  - Одна з цілей – записати ріст (мм) та масу (кг) різних людей.
  - Ріст – порядку 1700мм (int), маса – порядку 74.23кг.
  - Проблема: отримуємо або ріст, або масу для кожної особи.
  - Проте місце виділяється для обох значень, що веде до витрат пам'яті.

```
struct Person {  
    int height;  
    double weight;  
}
```



```
union Person {  
    int height;  
    double weight;  
}
```

```
union Person person1 = {1700};  
union Person person2 = {.weight=74.23};
```

# Об'єднання vs структури

```
1 #include <stdio.h>
2
3 union PersonUnion {
4     int height;
5     double weight;
6 };
7
8 struct PersonStruct {
9     int height;
10    double weight;
11 };
12
13 int main() {
14     union PersonUnion person1_union = {1700};
15     union PersonUnion person2_union = {.weight=74.23};
16
17     printf("Details of Union\n");
18     printf("Height of person1_union: %d\n", person1_union.height);
19     printf("Weight of person2_union: %f\n", person2_union.weight);
20     printf("Size of person1_union object: %lu\n", sizeof(person1_union));
21     printf("Size of person2_union object: %lu\n", sizeof(person2_union));
22     printf("\n");
23
24     struct PersonStruct person1_struct = {1700, 0.0};
25     struct PersonStruct person2_struct = {0, 74.23};
26
27     printf("Details of Struct\n");
28     printf("Height of person1_struct: %d\n", person1_struct.height);
29     printf("Weight of person2_struct: %f\n", person2_struct.weight);
30     printf("Size of person1_struct object: %lu\n", sizeof(person1_struct));
31     printf("Size of person2_struct object: %lu\n", sizeof(person2_struct));
32
33     return 0;
34 }
```

```
E:\GDisk\[College]\[фёютш яёюуёрьстрээ ёр рыуюёшсь|ўэ| ьютш]\хър 03. 4юї|фэ|
Details of Union
Height of person1_union: 1700
Weight of person2_union: 74.230000
Size of person1_union object: 8
Size of person2_union object: 8

Details of Struct
Height of person1_struct: 1700
Weight of person2_struct: 74.230000
Size of person1_struct object: 16
Size of person2_struct object: 16
```



PersonUnion.c

# Перелічувані типи

---

- Перелічуваний (enumerated) тип слугує для оголошення символічних імен, що представлятимуть цілочисельні константи.
  - За допомогою ключового слова `enum` можна створити новий "тип" та описати можливі значення.
  - Константи `enum`, по суті, мають тип `int`;
- Перелічувані типи підвищують розбірливість програми.
  - `enum spectrum { red, orange, yellow, green, blue, violet };`
    - `spectrum` – назва дескриптора, який дозволяє використовувати `enum spectrum` у якості назви типу.
  - `enum spectrum color;`
    - робить `color` змінною даного типу.
- Перелічення в мові C мають можливості, які не переносяться в C++.
  - Наприклад, операція інкременту «++» до перелічуваної змінної застосовна в C-кодi, проте не в C++.

## Константи типу enum

---

```
int c;  
color = blue;  
if (color == yellow)  
    ...;  
for (color = red; color <= violet; color++)  
    ...;
```

- Що являють собою blue і red?
  - З технічної точки зору – константи типу int.
- Перелічувані константи допускаються там же, де й цілочисельні константи.
  - Наприклад, в оголошеннях масивів або як мітки в операторі switch.
- За замовчуванням константам у перелічуваному списку присвоюються значення 0, 1, 2 і т. д.
  - `enum kids { nippy , slats , skippy , nina , liz };`
  - `nina` отримує значення 3.
- Цілі значення для констант можна обирати:
  - `enum levels { low = 100 , medium = 500 , high = 2000 };`

# Використання ключового слова enum

---

```
#include <stdio.h>
#include <string.h>    // for strcmp(), strchr()
#include <stdbool.h>    // C99 feature
char * s_gets(char * st, int n);

enum spectrum {red, orange, yellow, green, blue, violet};
const char * colors[] = {"red", "orange", "yellow", "green", "blue", "violet"};
#define LEN 30

int main(void)
{
    char choice[LEN];
    enum spectrum color;
    bool color_is_found = false;

    puts("Enter a color (empty line to quit):");
    ...
```

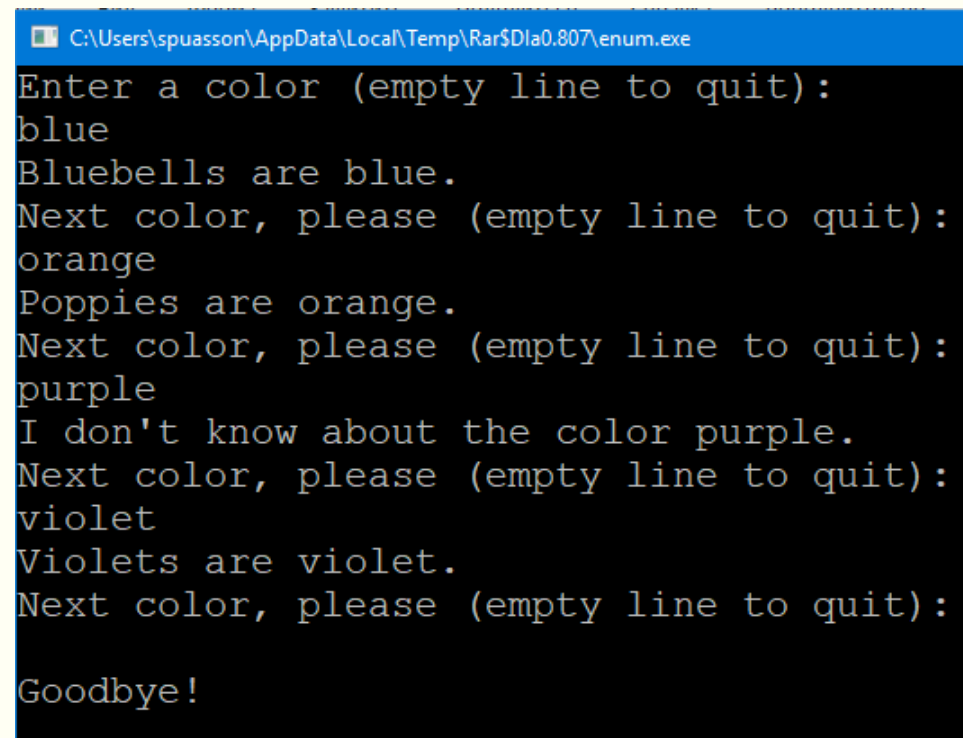
# Продовження коду та вивід програми

```
while (s_gets(choice, LEN) != NULL && choice[0] != '\0') {
    for (color = red; color <= violet; color++) {
        if (strcmp(choice, colors[color]) == 0) {
            color_is_found = true;
            break;
        }
    }
    if (color_is_found)
        switch(color) {
            case red      : puts("Roses are red."); break;
            case orange   : puts("Poppies are orange."); break;
            case yellow    : puts("Sunflowers are yellow."); break;
            case green     : puts("Grass is green."); break;
            case blue      : puts("Bluebells are blue."); break;
            case violet    : puts("Violets are violet."); break;
        }
    else
        printf("I don't know about the color %s.\n", choice);
    color_is_found = false;
    puts("Next color, please (empty line to quit):");
}
puts("Goodbye!");

return 0;
}
```

18.11.2020

@Марченко С.В., ЧДБК, 2020



```
C:\Users\spuasson\AppData\Local\Temp\Rar$Dla0.807\enum.exe
Enter a color (empty line to quit):
blue
Bluebells are blue.
Next color, please (empty line to quit):
orange
Poppies are orange.
Next color, please (empty line to quit):
purple
I don't know about the color purple.
Next color, please (empty line to quit):
violet
Violets are violet.
Next color, please (empty line to quit):

Goodbye!
```



ДЯКУЮ ЗА УВАГУ!