Інструкції вибору

Лекція №8

Дисципліна «Програмування»

Інструкція вибору if

Інструкція вибору **if** використовується для обрання однієї з наявних альтернатив.

Інструкція

```
if(grade >= 60)          // початок оператора if
{
    puts("Здано");
}                        // кінець оператора if
```

перевіряє істинність умови **grade >= 60**.

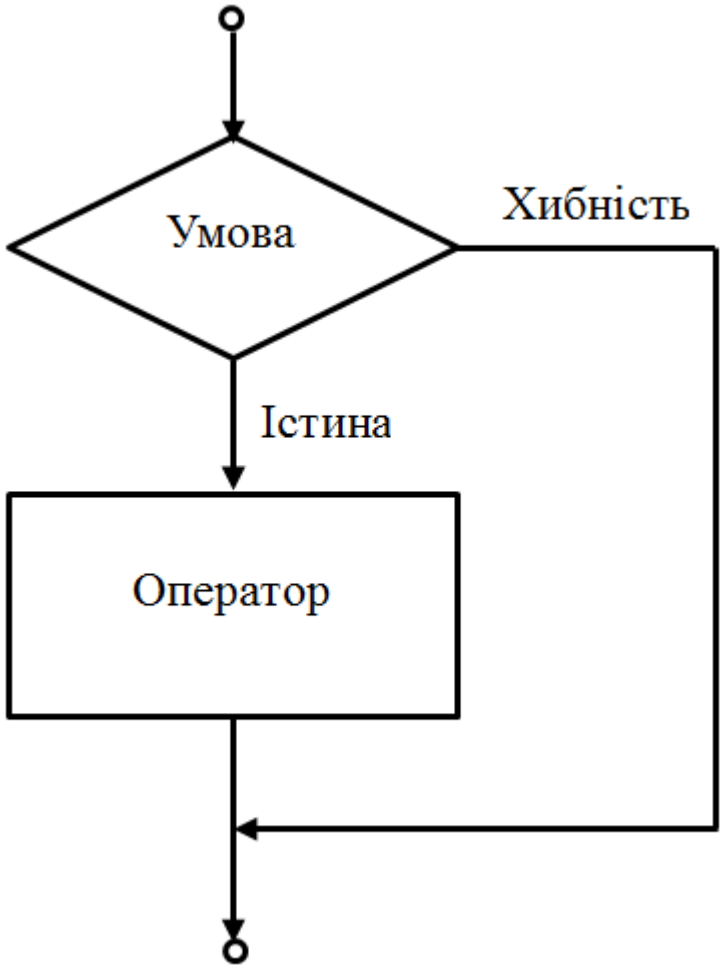
Якщо умова істинна, за допомогою функції **puts()** виводиться рядок "Здано" (пройдено), і виконання продовжується з інструкції, яка йде далі.

Інструкція вибору if

Функція `puts()` виводить рядок до стандартного потоку виводу.

Після виводу рядка відбувається перехід на новий рядок.

Символ кінця рядку (тобто нульовий символ) не виводиться.



Інструкція вибору if...else

Інструкція вибору **if...else** дозволяє вказати дві різні дії: одна виконується, коли умова істинна, друга – коли умова хибна.

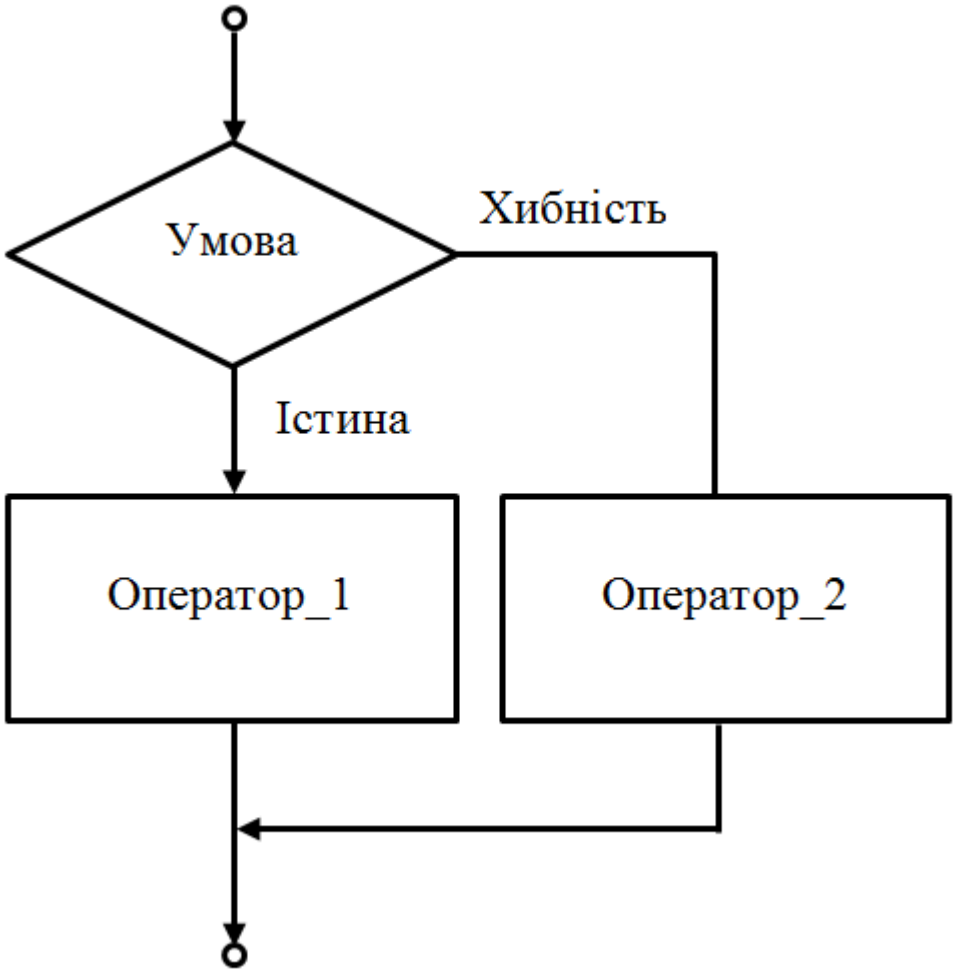
```
if(grade >= 60)
{
    puts("Здано");
} // кінець if
else
{
    puts("Не здано");
} // кінець else
```

Наприклад, інструкція виведе "Здано", якщо студент отримав оцінку **60** або вищу, і "Не здано", якщо оцінка нижче **60**.

В будь-якому випадку, після виводу того або іншого рядка, виконання буде продовжено з інструкції, яка йде наступною.

Тіло гілки **else** також має відступ.

Інструкція вибору if...else



Умовні оператор та вираз

В мові **C** є умовний оператор (**?:**), який дуже нагадує інструкцію **if...else**.

Умовний оператор – це єдиний **тернарний (тримісний) оператор** в мові **C**. Він приймає три операнди, і разом з умовним оператором вони утворюють **умовний вираз**.

Перший операнд – це умова.

Другий операнд – значення всього умовного виразу, коли **умова істинна**.

Третій операнд – значення всього умовного виразу, коли **умова хибна**.

Умовні оператор та вираз

Приклад: Функція **puts()**

```
puts(grade >= 60 ? "Здано" : "Не здано");
```

отримує в якості аргументу умовний вираз, який повертає рядок "Здано", якщо умова **grade >= 60** істинна, і рядок "Не здано", якщо умова хибна.

Функція **puts()**, по суті, виконує те ж саме, що й попередня інструкція **if...else**.

Умовні оператор та вираз

Другий і третій операнди в умовному виразі можуть бути також діями.

Наприклад, умовний вираз

```
grade >= 60 ? puts("Здано") : puts("Не здано");
```

читається так: «Якщо оцінка **grade** більше або дорівнює **60**, то виконати **puts("Здано")**, в протилежному випадку – **puts("Не здано")**».

Існують ситуації, коли умовний вираз використовувати можна, а інструкцію **if...else** – ні.

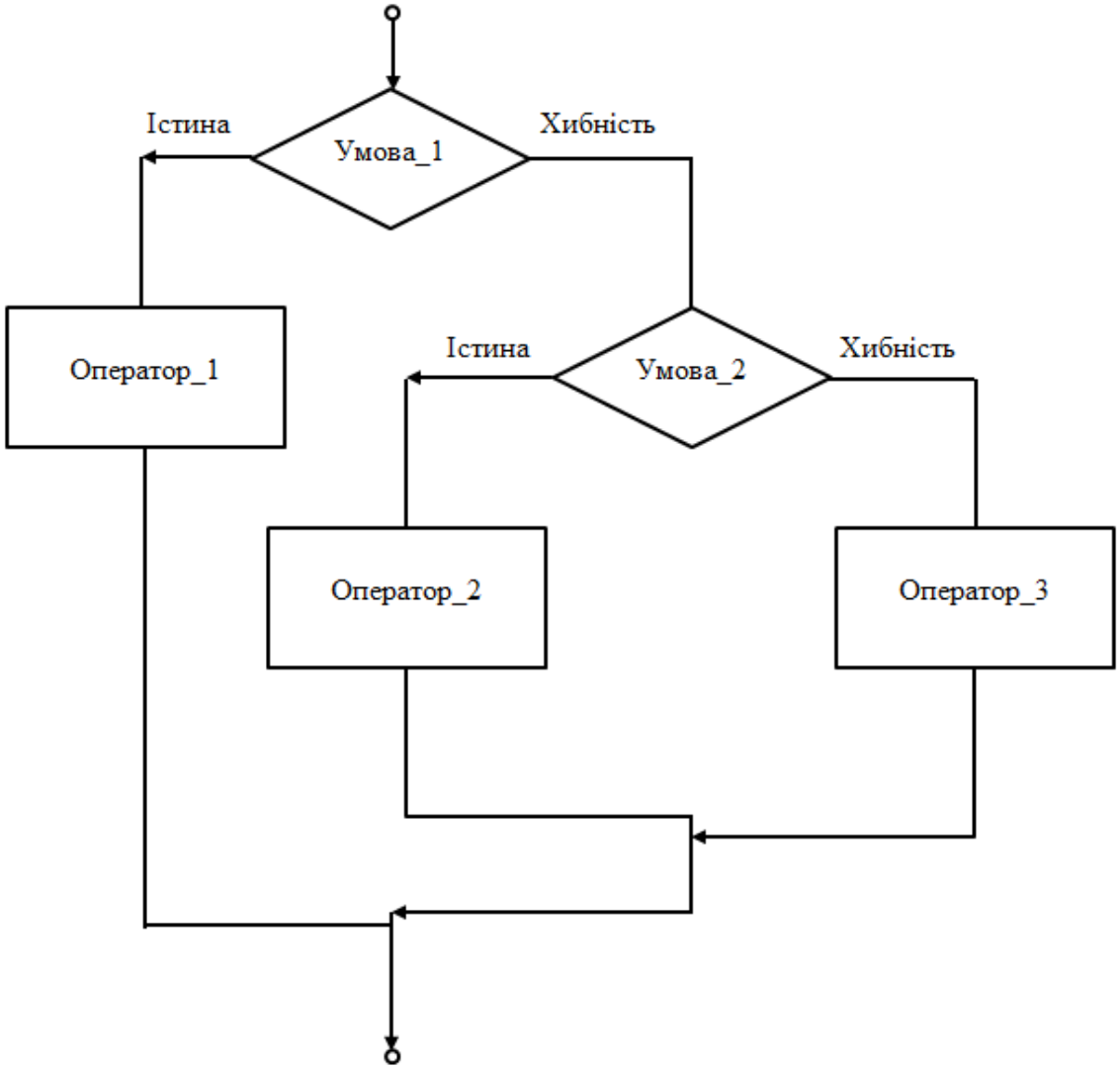
Вкладені інструкції if...else

Вкладені інструкції **if...else** дозволяють перевірити декілька умов за допомогою вкладення інструкцій **if...else** одна в одну.

Наприклад,

```
if(вираз) оператор_1;
else if(вираз) оператор_2;
     else оператор_3;
```

Вкладені інструкції if...else



Вкладені інструкції if...else

```
if(вираз)
    інструкція
else if(вираз)
    інструкція
else if(вираз)
    інструкція
else if(вираз)
    інструкція
else
    інструкція
```

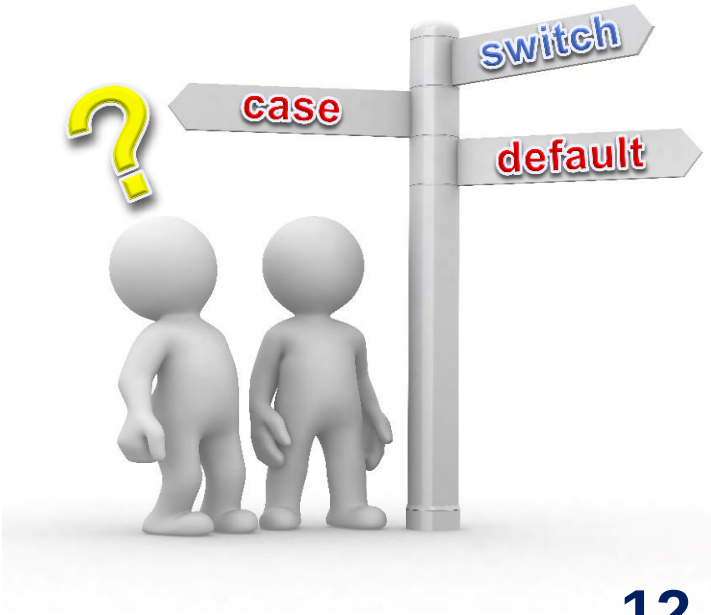
Остання **else**-частина спрацьовує, якщо не виконуються всі попередні умови.

Іноді в останній частині не треба виконувати жодних дій, в цьому випадку цей фрагмент можна опустити або використовувати для фіксації помилкової («неможливої») ситуації.

Інструкція switch

Інструкція **switch** використовується для множинного вибору. Вона перевіряє, чи збігається значення виразу з одним зі значень, що входять до деякої множини цілих констант, і виконує гілку програми, яка відповідає цьому значенню:

```
switch(вираз)
{
     case конст-вираз: інструкції;
                       break;
     case конст-вираз: інструкції;
                       break;
               default: інструкції;
                       break;
}
```



Інструкція switch

Кожна гілка **case** помічена однією або декількома цілочисловими константами або ж константними виразами.

Обчислення починаються з тієї гілки **case**, в якій константа співпадає зі значенням виразу.

Константи усіх гілок **case** повинні відрізнятися одна від одної.

Якщо з'ясується, що жодна з констант не підходить, то виконується гілка, яка помічена словом **default** (якщо така має ться), в іншому випадку нічого не робиться.

Гілки **case** і **default** можна розміщувати в будь-якому порядку.

Інструкція switch

Інструкція **break** призводить до миттєвого виходу з інструкції **switch**.

Оскільки вибір гілки **case** реалізується як перехід на мітку, то після виконання однієї гілки **case**, якщо нічого не застосовувати, програма почне виконувати наступну гілку.

Інструкції **break** і **return** – найбільш поширені засоби виходу з інструкції **switch**.

Інструкція **break** застосовуються також для примусового виходу з циклів **while**, **for** і **do...while**.

Інструкція switch

Послідовний прохід по гілкам – річ ненадійна, це може бути пов’язано з помилками, особливо під час модифікації програми. За виключенням випадку з декількома мітками для одного обчислення.

Намагайтеся, за можливістю, не користуватися наскрізним проходом, але якщо ви його застосовуєте, обов’язково коментуйте ці особливі місця.

Навіть наприкінці останньої гілки (після **`default`**) розміщуйте інструкцію **`break`**, хоча з точки зору логіки в ній немає жодної необхідності. Але ця маленька обережність врятує, коли одного разу вам необхідно буде додати в кінець ще одну гілку **`case`**.

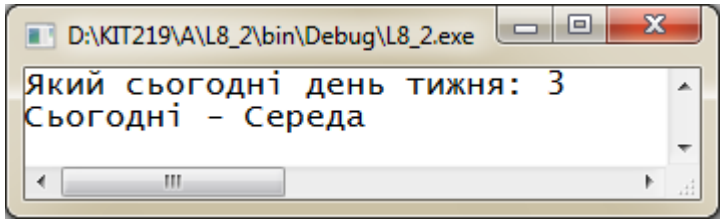
Приклади використання

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

int main(void)
{
    unsigned day;

    SetConsoleOutputCP(1251);

    printf("Який сьогодні день тижня: ");
    scanf("%d", &day);
    if (day > 7)
        day = day % 7;
    printf("Сьогодні - ");
```



Приклади використання

```
switch(day)
{
        case 0: printf("Неділя\n");
                break;
        case 1: printf("Понеділок\n");
                break;
        case 2: printf("Вівторок\n");
                break;
        case 3: printf("Середа\n");
                break;
        case 4: printf("Четвер\n");
                break;
        case 5: printf("П'ятниця\n");
                break;
      default: printf("Субота\n");
                break;
}
return 0;
}
```