

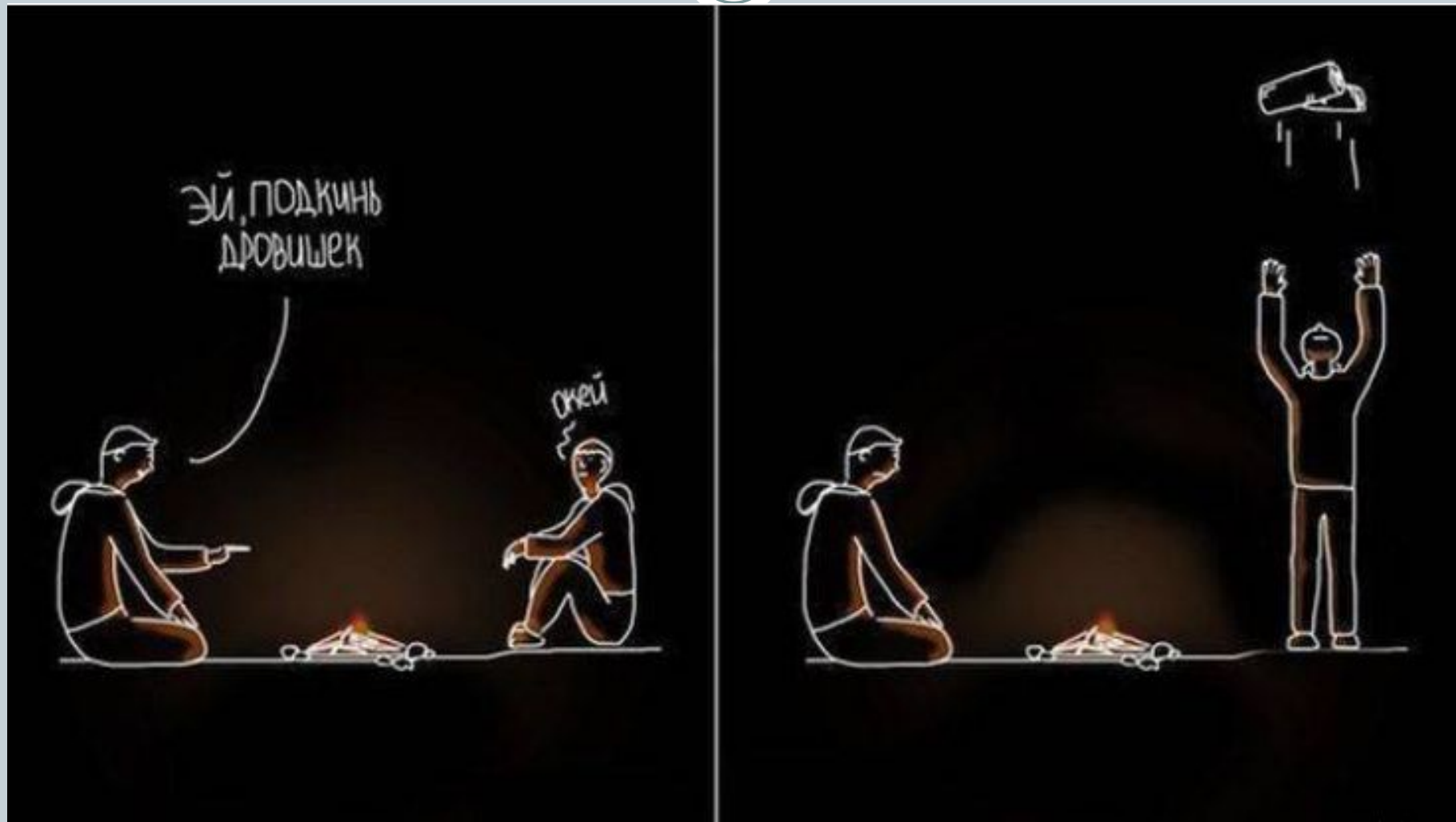
Заняття 2



ЛОГОС

Основи Програмування

Що повинна робити програма?



Що повинна робити програма?



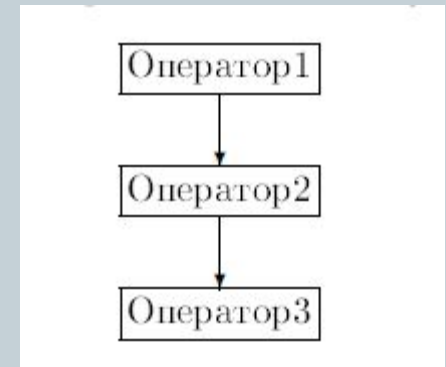
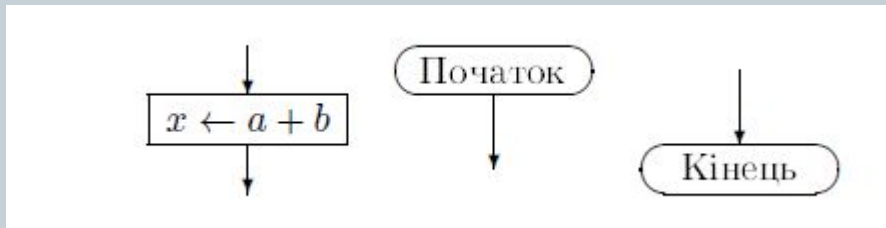
Оскільки програма являє собою опис способу вирішення певної задачі, записаний у мові з жорстко заданими, формалізованими правилами, то вона виконуватиме лише ті команди які ми їй сказали.

У програмуванні слова не мають двозначностей як у людській мові, яку щоб розуміти, не раз доводиться застосовувати логіку та інтуїцію. З мовами програмування все набагато простіше, одне слово – одна команда, немає двозначностей.

Алгоритм



Алгоритм – це сукупність елементарних операцій та правил, які визначають, в якому порядку ці операції виконуються .

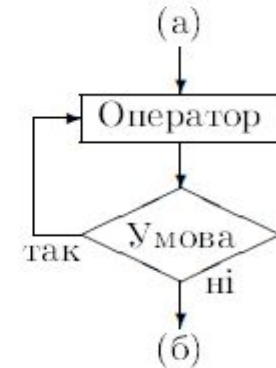
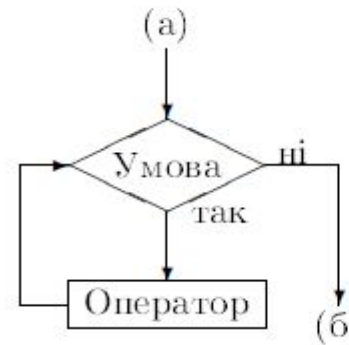
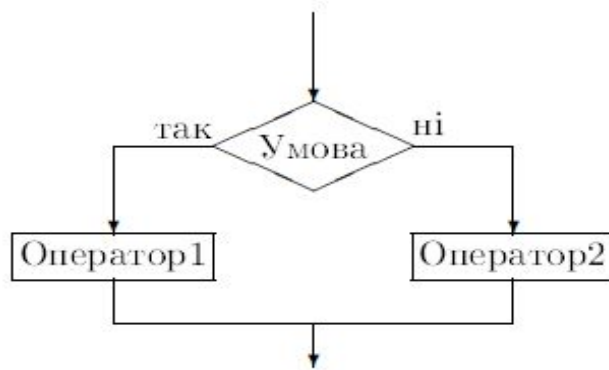


Найпростіші позначення в блоках схеми, та послідовне виконання операторів

Алгоритм



Алгоритм – це сукупність елементарних операцій та правил, які визначають, в якому порядку ці операції виконуються .

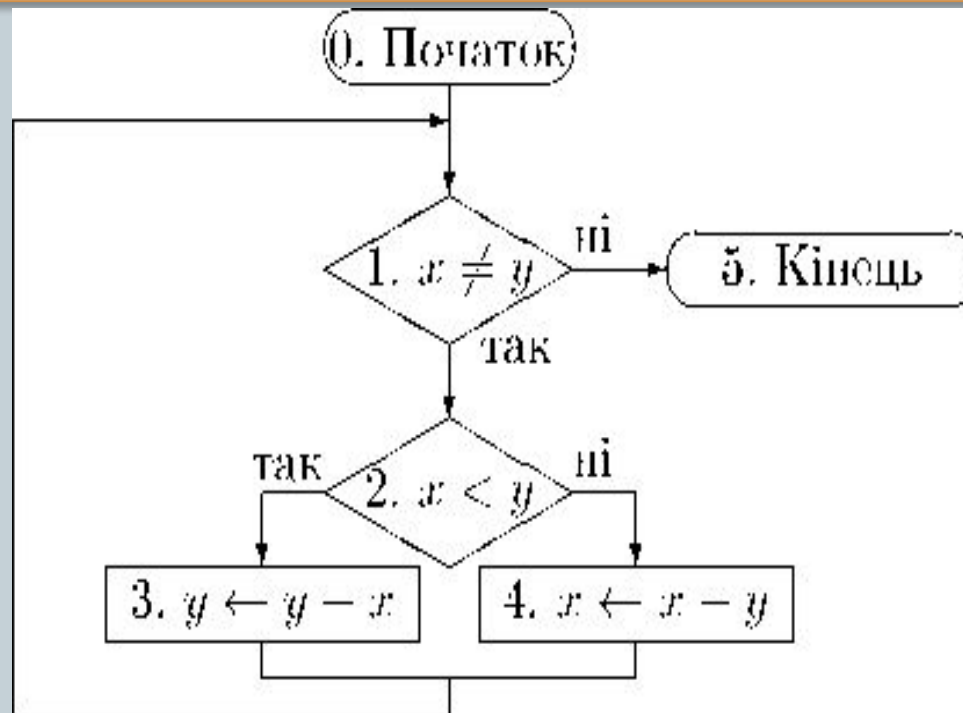


Умозна конструкція (розгалуження), цикли з передумовою та пост-умовою

Алгоритм



Приклад алгоритму Евкліда - для пошуку найбільшого спільного кратного двох чисел



Блок-схема алгоритму Евкліда

Алгоритм



Протокол процесу виконання алгоритму Евкліда

Крок	Блок	x	y	Примітка
0	0	15	9	Початок. Перехід до 1
1	1	15	9	$x = y$ істинна, перехід до 2
2	2	15	9	$x < y$ хибна, перехід до 4
3	4	6	9	перехід до 1
4	1	6	9	$x = y$ істинна, перехід до 2
5	2	6	9	$x < y$ істинна, перехід до 3
6	3	6	3	перехід до 1
7	1	6	3	$x = y$ істинна, перехід до 2
8	2	6	3	$x < y$ хибна, перехід до 4
9	4	3	3	перехід до 1
10	1	3	3	$x = y$ хибна, перехід до 5
11	5	3	3	Алгоритм завершено

Арифметичні операції



Оператор	Операція	Оператор	Операція
+	Додавання	+=	Додавання з присвоєнням
-	віднімання (а також унарний мінус)	-=	Віднімання з присвоєнням
*	Множення	* =	Множення з присвоєнням
/	Ділення	/=	Ділення з присвоєнням
%	Залишок ділення по модулю	%=	Залишок ділення по модулю з присвоєнням
++	Інкремент (збільшення на 1)	--	Декремент (зменшення на 1)

Оператори відношення



Оператор	Опис
==	Рівно
!=	Не рівно
>	Більше
<	Менше
>=	Більше рівне
<=	Менше рівне

Булеві оператори



Оператор	Опис
	Коротке АБО
&&	Коротке І
!	Логічне одиничне НІ (NOT)
==	Рівність
!=	Не рівність

Типи даних



Тип	Довжина (в байтах)	Діапазон або набір значень
boolean	не визначено	true, false
byte	1	-128..127
char	2	0.. $2^{16}-1$, або 0..65535
short	2	-2^{15} .. $2^{15}-1$, або -32768..32767
int	4	-2^{31} .. $2^{31}-1$, або -2147483648..2147483647
long	8	-2^{63} .. $2^{63}-1$, або приблизно $-9.2 \cdot 10^{18}$.. $9.2 \cdot 10^{18}$
float	4	$-(2 \cdot 2^{-23}) \cdot 2^{127}$.. $(2 \cdot 2^{-23}) \cdot 2^{127}$, або приблизно $-3.4 \cdot 10^{38}$.. $3.4 \cdot 10^{38}$, а також $-\infty$, ∞ , NaN
double	8	$-(2 \cdot 2^{-52}) \cdot 2^{1023}$.. $(2 \cdot 2^{-52}) \cdot 2^{1023}$, або приблизно $-1.8 \cdot 10^{308}$.. $1.8 \cdot 10^{308}$, а також $-\infty$, ∞ , NaN

Створення змінної



```
int k;  
double variable;  
char myChar;  
long a123;
```

Оголошення змінних розпочинаються з обов'язкового вказування типу даних, після чого йде назва змінної.

Створення змінної



```
int i; //оголошення змінної
```

```
int j=1; //оголошення з ініціалізацією
```

```
char key;
```

```
key = 'Y'; //ініціалізація
```

Ініціалізацію змінної можна здійснити як при оголошенні так і в подальшому в програмі.

Створення змінної



```
int a, b, c;
```

В одному рядку можна оголошувати декілька змінних одного типу, проте в такому разі ускладнюється читання коду програми.

УМОВНІ КОНСТРУКЦІЇ



```
if(boolean expression){  
    do something...}
```

Умова повинна бути оточена дужками і, якщо, умова **вірна (true)** буде виконана інструкція за умовою, інакше вона не буде виконана, а буде виконана наступна інструкція після умовної інструкції.

УМОВНІ КОНСТРУКЦІЇ



```
if (boolean expression){  
    do something...  
} else {  
    do something else...  
}
```

Конструкція **if-else**

УМОВНІ КОНСТРУКЦІЇ



```
if (boolean expression) {  
    do something...  
} else if (another boolean expression) {  
    do something else...  
}
```

Конструкція **if-else-if**

УМОВНІ КОНСТРУКЦІЇ



```
if (boolean expression){  
    do something...  
} else if (another boolean expression) {  
    do something else...  
} else if (another boolean expression2) {  
    do something else...  
} else if (another boolean expression3) {  
    do something else...  
} else{  
    do something else...  
}
```

if-ів може бути безліч

Цикли



Цикли – це послідовність інструкцій, які можуть **повторно** виконуватись певну кількість раз в залежності від заданої в програмі умови.

Розрізняють цикли з передумовою, з післяумовою та з лічильником.

while



```
while (boolean expression) {  
    do something...  
}
```

- **Цикл `while`** (перекладається як «доки») – це цикл з передумовою, тіло якого виконується, якщо умова **істинна**. Якщо умова з самого початку хибна, то цикл не виконається жодного разу.

Цикл do/while [z

[z



```
do {  
    do something...  
} while (boolean expression);
```

Якщо необхідно, щоб умова виконувалася **хоча б один раз** можна скористатися циклом з післяумовою **do/while**

Цикл for



```
for (int i=1; i<=10; i++){  
    do something...  
}
```

- **Цикл for** – доволі часто вживаний цикл. Він застосовується при необхідності виконати інструкції **певну кількість раз** з одночасним **збільшенням** або **зменшенням** певної **змінної**. Часто використовується для здійснення перебору певних масивів даних.