Тема 4

- Вирази
- Арифметичні вирази
- Логічні вирази
- Рядкові вирази
- Оператори
- Оператори привласнення і порівняння
- Операторні дужки
- Оператор if..else
- Оператори циклів

2

Вирази складаються

- Операндів
- Знаків операцій
- Круглих дужок

Операція привласнення

```
A=12;
X=a;
y=a;
x=x;
S=”група СУ-71”;
F=false;
M=true;
K=3.13;
U=3.543e-78;
```

Арифметичні операції

Операція (вираз)		Оператор	Синтаксис виразу
Привласнення		=	a = b
Додавання		+	a + b
Віднімання		-	a - b
Унарний плюс		+	+a
Унарний мінус		-	-a
Множення		*	a * b
Ділення		/	a / b
Залишок ділення		%	a % b
Інкремент	префіксний	++	++a
	суфіксний	++	a++
Декремент	префіксний	--	--a
	префіксний	--	a--

Основні арифметичні операції

Операція	Призначення	Типи операндів	Тип результату
+	Додавання	Цілочисельний Дійсний	Цілочисельний Дійсний
-	Віднімання	Цілочисельний Дійсний	Цілочисельний Дійсний
*	Множення	Цілочисельний Дійсний	Цілочисельний Дійсний
/	Ділення	Цілочисельний Дійсний	Дійсний Дійсний

Функції для роботи з числовими даними

Бібліотека **math.h**

- abs(X) – абсолютне значення X
- fabs(X) – абсолютне значення X
- sqrt(X) – квадратний корінь з X
- log(X) – натуральний логарифм X
- exp(X) – піднесення числа e до ступеню X
- sin(X) – синус кута, який задано у радіанах

Приклади арифметичних операцій

```
(x+12.3) / 30 * sin(2 *alpha)
Y + x
exp(3)
12 - 4.01e-4 * log(a2)
sqrt(2*x5)
acos(4-c1)
```

Операції цілочисельного ділення

- / – результат операції ділення залежить від типу змінної в яку заноситься частка від ділення двох чисел
- % - цілочисельний залишок від ділення двох чисел

Приклади

Тип змінної	Операції	Результат
`double`	`56 / 7`	`8.0`
`int`	`56 / 7`	`8`
`double`	`42 / 4`	`10.5`
`int`	`42 / 4`	`10`
`int`	`42 % 4`	`2`

Склад логічних виразів

- Логічні константи `true` та `false`
- Логічні змінні типу `bool`
- Операції порівняння (відношення)
- Логічні операції
- Круглі дужки

Операції порівняння (відношення)

Операція	Назва
==	Дорівнює
<	Менше
>	Більше
<=	Менше або дорівнює
>=	Більше або дорівнює
<>	Не дорівнює

Логічні операції

Операція	Опис	Операнд 1	Операнд 2	Результат
!	Заперечення	false true		true false
&&	Логічне «так»	false false true true	false true false true	false false false true
||	Логічне «або»	false false true true	false true false true	false true true true
^	Виключне «або»	false false true True	false true false True	false true true false

Складені оператори присвоєння

Ім'я оператора	Синтаксис	Сенс
Додавання з присвоєнням	a += b	a = a + b
Віднімання з присвоєнням	a -= b	a = a - b
Множення з присвоєнням	a *= b	a = a * b
Ділення з присвоєнням	a /= b	a = a / b
Отримання залишку з присвоєнням	a %= b	a = a % b
Побітове І з присвоєнням	a &= b	a = a & b
Побітове АБО з присвоєнням	a |= b	a = a | b
Побітове виключаюче АБО з присвоєнням	a ^= b	a = a ^ b
Побітовий зсув вліво з присвоєнням	a <<= b	a = a << b
Побітовий зсув вправо з присвоєнням	a >>= b	a = a >> b

Тернарний умовний оператор

Тернарна операція має наступний синтаксис:

```
<умова> ? <значення1>:<значення2>
```

Приклад тернарної операції :

```
min = (a < b) ? a : b;
```

Рядкові вирази містять

- UnicodeString – рядковий тип змінних
- Одну операцію «+» яка виконує з'єднання (конкатенацію) рядків
- Функції над рядками

Приклад конкатенації :

```
UnicodeString s =”Іван”;
s = s + ”Петренко”;
```

Результат :

```
”ІванПетренко”
```

Функції над рядками (приклад)

- `int s.Length()` – **визначення довжини рядка**
- `int s.Pos(const UnicodeString& subStr)` – **повертає індекс символу, з якого починається вказана підстрока.**
- `s.Insert(const UnicodeString & str, int Index)` – **вставка рядка** `str` **у рядок** `s` **з позиції** `Index`.
- `s.Delete(int n, int idx)` – **видалення з рядку** `s` `n` **символів починаючи з** `idx`.

Функції приведення типів

- `IntToStr()` – самостійно
- `StrToInt()` – самостійно
- `FloatToStr()` – самостійно
- `StrToFloat()` – самостійно

Функції перетворення рядків

- `s.UpperCase()` – самостійно
- `s.LowerCase()` – самостійно
- `s.Trim()` – самостійно
- `s.TrimLeft()` – самостійно
- `s.TrimRigth()` – самостійно

Прості оператори

- Оператор привласнення (=)
  - `x2=w1;`
- Оператор безумовного переходу
  - `goto <label>`
- Пустий оператор `;`
- Оператор виклику функції
  - `Close();`

Структуровані оператори

- Складений оператор
- Умовний оператор (оператор умовного переходу)
- Оператор вибору
- Оператори циклу
- Оператор доступу

Складений оператор

- Формат оператора

```
{
    <оператор1>;
    …;
    <операторN>;
}
```

- Приклад складеного оператора

```
{
    int x = 5, y = 7;

    x = x ^ y;
    y = x ^ y;
    x = x ^ y;

    m1->Lines->Add(IntToStr(x));
}
```

Умовний оператор (оператор умовного переходу)

- Формат оператора

```
if (<умова>)
    {
    <оператор1>;
    }
else
    {
    <оператор2>;
    }
```

```
if (<умова>)
{
    <оператор1>;
}
```

- Приклад умовних операторів

```
if (x>0) {x=x+1;} else {x=0;}

if (q==0) {a=1;}
```

Оператор вибору

- Формат оператора

```
switch(switch_variable) {casebreakdefault
    case constant_expression: statement; // [break;]
    // …
    default: statement;
```

- Приклад

```
int  DayNumber = 1;
UnicodeString str ;
          switch (DayNumber)
          {
              case 1 :
              case 2 :
              case 3 :
              case 4 :
              case 5 :
                    str = "Робочий день";
                    break;
               case 6 :
               case 7:
                    str = "Вихідний день";
                    break;
               default :
                   str = "Нема такого дня!!!";
          }
```

Оператори циклу

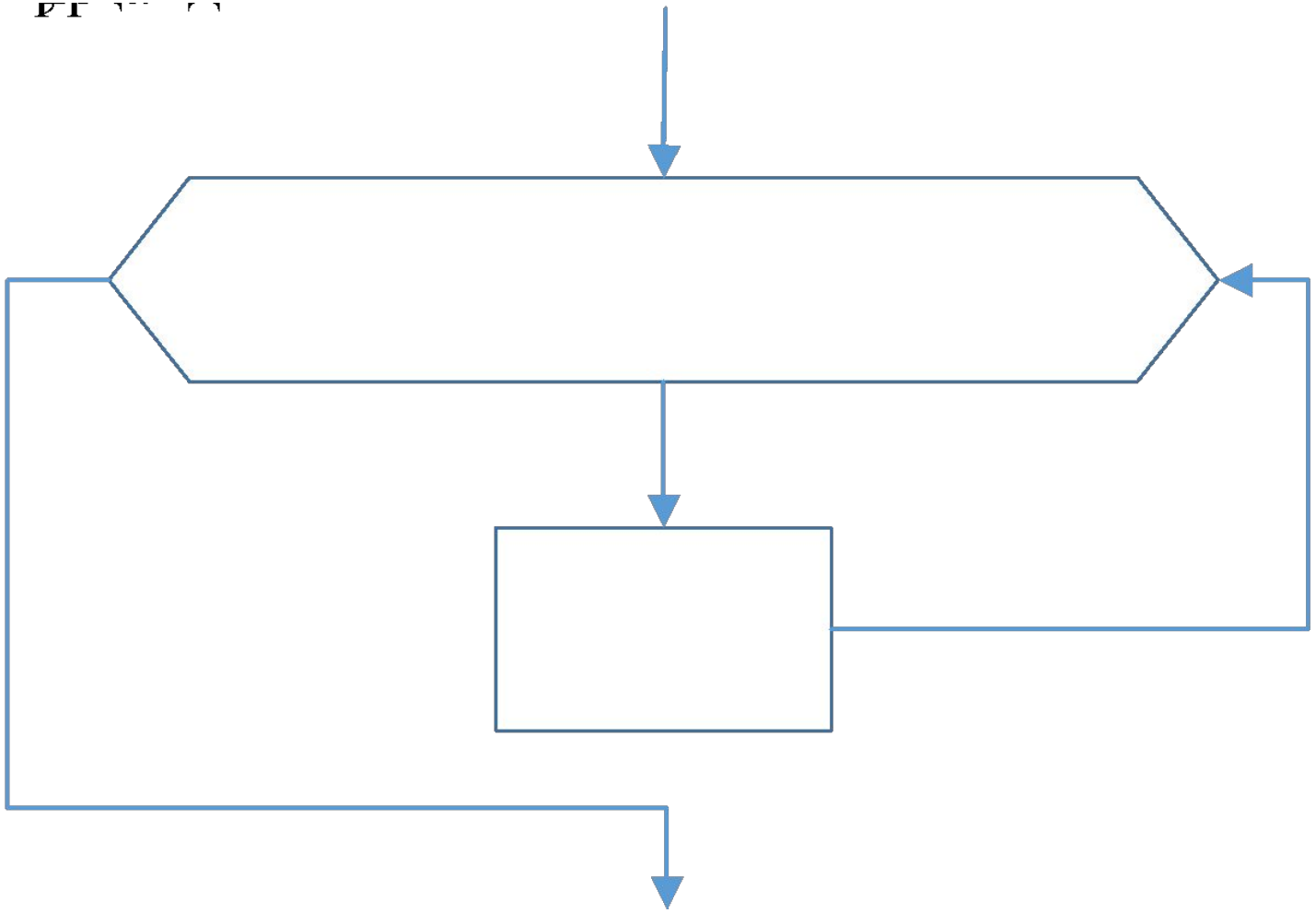
- Цикл з параметром (лічильником)
- Цикл з передумовою
- Цикл з післяумовою

Переривання циклу

```
break;

continue;
```

Цикл з лічильником (цикл «для»)



Цикли з параметром

- Формат оператора

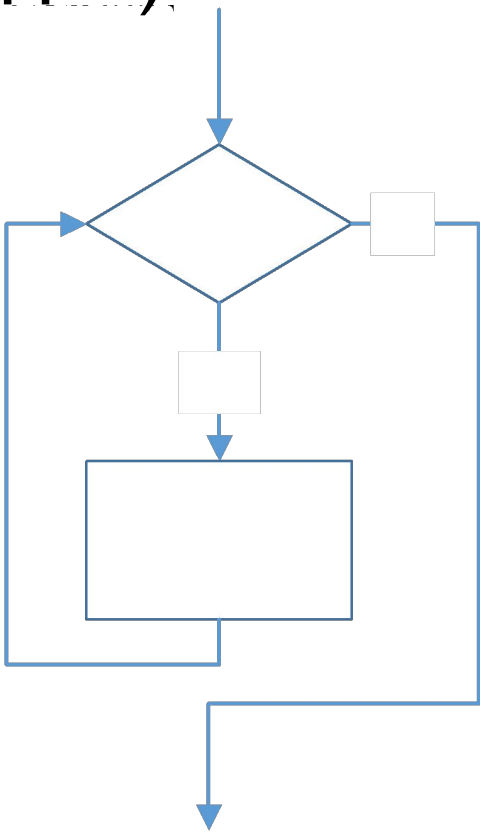
```
for (<ініціалізація параметра>; <умова>; <зміна значення параметра>)
{
<тіло циклу>
}
```

- Приклад

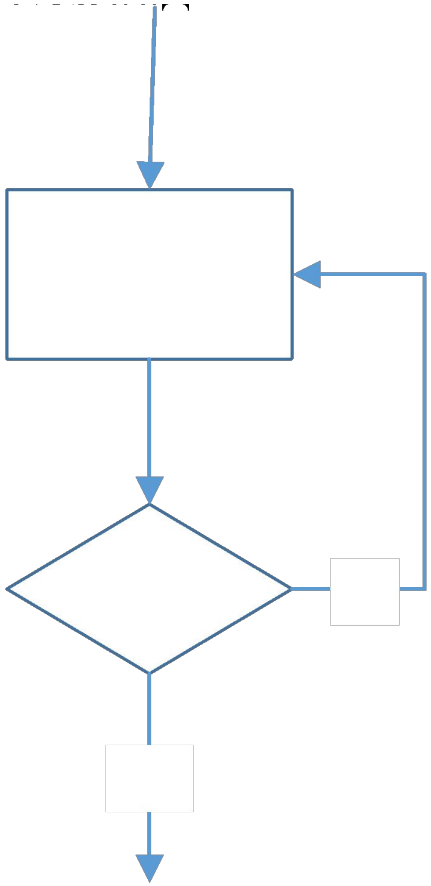
```
double Sum=0;
for (i=1; i<10; i++)
{
    sum += m[i];
}
```

Цикли з умовою

- Цикл з передумовою (цикл «до»)



- Цикл з післяумою (цикл «після»)



Цикли з умовою

- Цикл з передумовою (до)

```
while (<умова>)
{
    <оператор1>;
}
```

- Цикл з післяумовою (після)

```
do
{
    <оператор1>;
    …;
    <операторN>;
} while (<умова>);
```

Приклади

```
i = 1; sum = 0;
while (i<=10)
{
    sum += m[i];
    i++;
}
```

```
i = 1; sum = 0;
do
{
    sum += m[i];
    i++;
} while (i<=10)
```