Al-Mustaqbal University
College of Sciences
Department of Cybersecurity

كليـــة العلـــوم
قسم الأمن السيبراني

Subject: Object Oriented Programming (OOP)

Second Stage

Lecturer: Dr. Abdulkadhem A. Abdulkadhem

Lecture (1)

Back to Basics: Revisiting Key C++ Principles

2024-2025

Programming is the process of writing instructions for a computer in a certain order to solve a problem.

Special Characters: In C++ , all characters other than listed treated as special characters for example:

+	-	*	/	^
(	[	{	}	]
)	<	=	>	, (Comma)
" (Double Conations)	. (Dot)	: (Colon)	; (Semicolon)	␣ (Blank Space)

Reserved words cannot be used as variable names or constant. The following words are reserved for use as keywords:

Some of C++ Language Reserved Words:				
break	case	char	cin	cout
delete	double	else	enum	false
float	for	goto	if	int
long	main	private	public	short
sizeof	switch	true	union	void

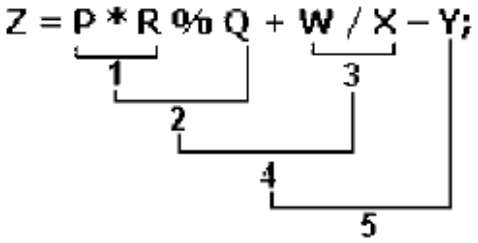
Example 2:

State the order of evaluation for the following expression:

Z = P * R % Q + W / X – Y;

Solution:

1. *
2. %
3. /
4. +
5. -



The "math.h" library contains the common mathematical function used in the scientific equations.

Common function from math.h library:	
Mathematical Expression	**C++ Expression**
e^n	Exp(x)
Log(x)	Log10(x)
Ln(x)	Log(x)
Sin(x)	Sin(x)
x^n	Pow(x,n)
$\sqrt{x}$	Sqrt(x)

Example:

Write the following equation as a C++ expression and state the order of evaluation of the binary operators:

$$f = \sqrt{\frac{\sin(x) - x^5}{\ln(x) + \frac{x}{4}}}$$

Solution:

f = sqrt ((sin(x) – pow(x,5)) / (log(x) + x/4))

Using For Statement	Using While Statement	Using Do/While Statement

Q1: Find the summation of the numbers between 1 and 100.

```
for( i=1 ; i<=100 ; i++ )
   s = s + i;
```

```
i = 1;
while ( i <= 100)
{
   s = s + i;
   i++;
}
```

```
i = 1;
do
{
   s = s + i;
   i++;
}
while ( i <= 100 );
```

Q2: Find the factorial of n.

```
cin >> n;
for( i=2 ; i<=n ; i++ )
   f = f * i;
```

```
cin >> n;
i = 2;
while ( i <= n)
{
   f = f * i;
   i++;
}
```

```
cin >> n;
i = 2;
do
{
   f = f * i;
   i++;
}
while ( i <= n);
```

Q3: To find the result of the following: $\sum_{i=1}^{20} a_i^2$.

```
for( i=1 ; i<=20 ; i++ )
   s = s + (i *i);
```

```
i = 1;
while ( i <= 20)
{
   s = s + (i *i);
   i++;
}
```

```
i = 1;
do
{
   s = s + (i *i);
   i++;
}
while ( i <= 20);
```

Q4: Read 10 numbers, and find the sum of the positive numbers only.

```
for( i=1 ; i<=10 ; i++ )
{
   cin >> x;
   if ( x>0 ) s = s + x;
}
```

```
i = 1;
while ( i <= 10)
{
   cin >> x;
   if ( x>0 ) s = s + x;
   i++;
}
```

```
i = 1;
do
{
   cin >> x;
   if ( x>0 ) s = s + x;
   i++;
}
while ( i <= 10);
```

Q5: Represent the following series: 1, 2, 4, 8, 16, 32, 64.

```
for( i=1 ; i<65 ; i*=2 )
  cout << i;
```

```
i = 1;
while ( i<65)
{
   cout << i;
   i*=2;
}
```

```
i = 1;
do
{
   cout << i;
   i*=2;
}
while ( i<65);
```

Q6: Find the sum of the following s = 1 + 3 + 5 + 7 + ... + 99.

```
for( i=1 ; i<=99 ; i+=2 )
   s = s + i;
```

```
i = 1;
while ( i<=99)
{
   s = s + i;
   i+=2;
}
```

```
i = 1;
do
{
   s = s + i;
   i+=2;
}
while ( i<=99);
```

Q7: Find the sum and average of the 8 degrees of the student.

```
for( i=1 ; i<=8 ; i++ )
{
   cin >> d;
   s = s + d;
}
av = s / 8;
```

```
i = 1;
while ( i<=8)
{
   cin >> d;
   s = s + d;
   i++;
}
av = s / 8;
```

```
i = 1;
do
{
   cin >> d;
   s = s + d;
   i++;
}
while ( i<=8);
av = s / 8;
```

Structures are typically used to group several data items together to form a single entity. It is a collection of variables used to group variables into a single record. Thus a structure (the keyword struct is used in C++) is used. Keyword struct is a data-type, like the following C++ data-types (int, float, char, etc ...). This is unlike the array, which all the variables must be the same type. The data items in a structure are called the members of the structure.

```
#include <iostream.h>

struct data
{
   char *name;
   int age;
};

void main()
{
   struct data student;
   student.name="ahmed";
   student.age=20;
     :
}
```

Structured/Procedural Programming	Object Oriented Programming
Code is divided into modules or functions	Code is made up of classes and objects
Town-down approach	Bottom-up approach
Difficult to modify / manage	Easy to modify / manage
Main function calls other functions	Objects communicate by passing messages
Data is not secured	Data is secure
Less reusability of code	More reusability of code
Less flexibility and abstraction	More flexibility and abstraction