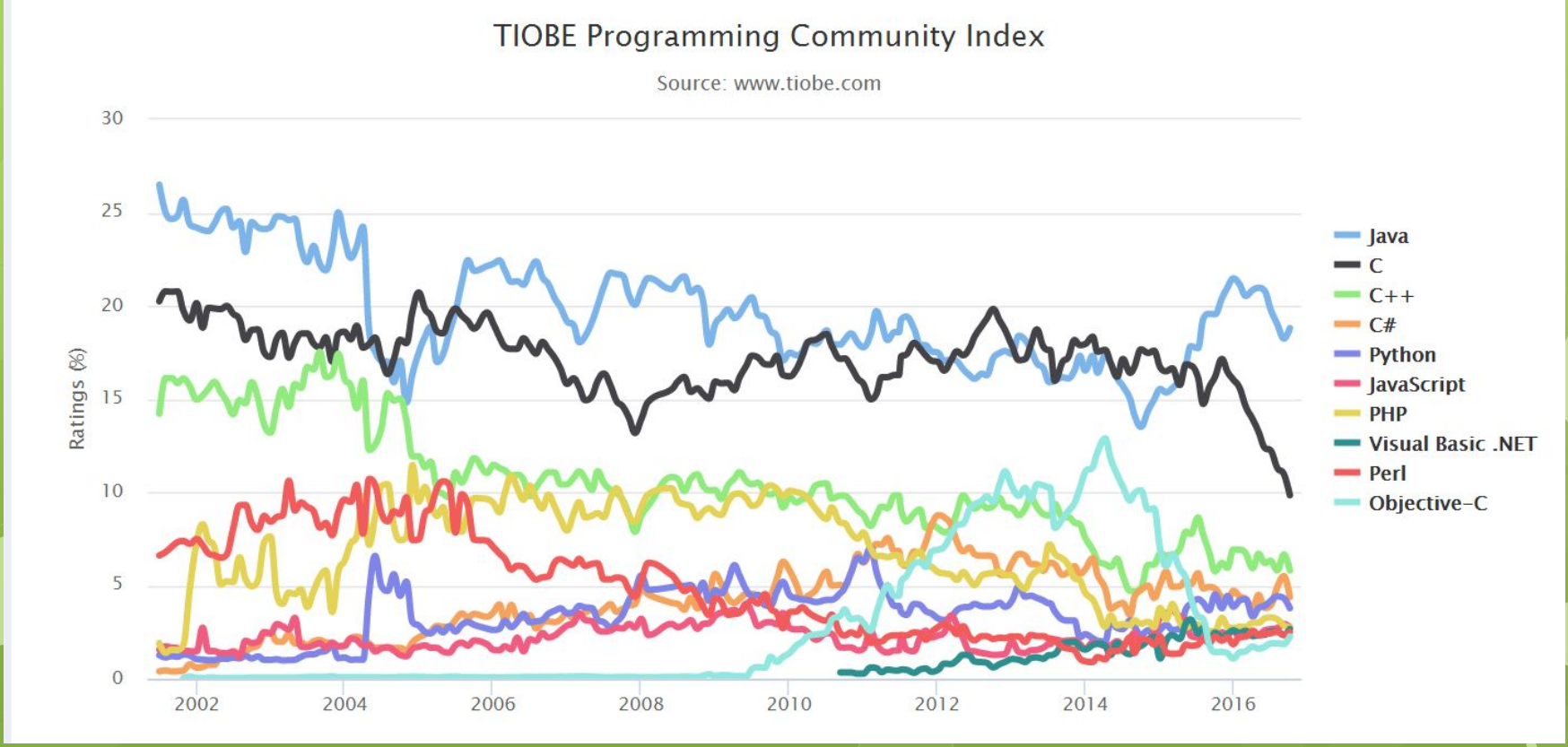
Вступна лекція
Програмування мовою C/C++

- **C** (укр. *Ci*) — універсальна, процедурна мова програмування загального призначення, розроблена у 1972 році Деннісом Рітчі у Bell Telephone Laboratories з метою написання нею операційної системи UNIX.
- Хоча C і було розроблено для написання системного програмного забезпечення, наразі вона досить часто використовується для написання прикладного програмного забезпечення.
- С імовірно, є найпопулярнішою у світі мовою програмування за кількістю вже написаного нею програмного забезпечення, доступного під вільними ліцензіями коду та кількості програмістів, котрі її знають. Версії компіляторів для мови C існують для багатьох операційних систем та апаратних архітектур. C здійснила великий вплив на інші мови програмування, особливо на C++, яка спочатку проектувалася, як розширення для C, а також на Java та C#, які запозичили у C синтаксис.

Сучасний стан рейтингу застосування мов згідно TIOBE



http://www.tiobe.com/tiobe-index/

TIOBE Index for October 2016/2015

Oct 2016	Oct 2015	Change	Programming Language	Ratings	Change
1	**1**		**Java**	**18.799%**	**-0.74%**
2	**2**		**C**	**9.835%**	**-6.35%**
3	**3**		**C++**	**5.797%**	**+0.05%**
4	4		C#	4.367%	-0.46%
5	5		Python	3.775%	-0.74%
6	8		JavaScript	2.751%	+0.46%
7	6		PHP	2.741%	+0.18%
8	7		Visual Basic .NET	2.660%	+0.20%
9	9		Perl	2.495%	+0.25%
10	14		Objective-C	2.263%	+0.84%
11	12		Assembly language	2.232%	+0.66%
12	15		Swift	2.004%	+0.73%
13	10		Ruby	2.001%	+0.18%
14	13		Visual Basic	1.987%	+0.47%
15	11		Delphi/Object Pascal	1.875%	+0.24%
16	65		Go	1.809%	+1.67%
17	32		Groovy	1.769%	+1.19%
18	20		R	1.741%	+0.75%
19	17		MATLAB	1.619%	+0.46%
20	18		PL/SQL	1.531%	+0.46%

Історичні етапи розвитку мови С

Історичні етапи розвитку	Рік
Unix (в першу чергу Кеном Томпсоном, Деннісом Рітчі і Дугласом Макілроєм)	Кінець 60-х
Мова BCPL (Кеном Томпсоном)	1966
Мова B (оригінальна розробка Томпсона під UNIX)	1969
Мова C (Денніс Ріттчі)	1972
Ядро Unix переписується на C	1973
Книжка «Язык программирования Си» («K&R C») (Брайан Керніган и Денніс Рітчі)	1978
ANSI C или C89	1989
C99	1999
C11	2011

Поточний стан застосування С

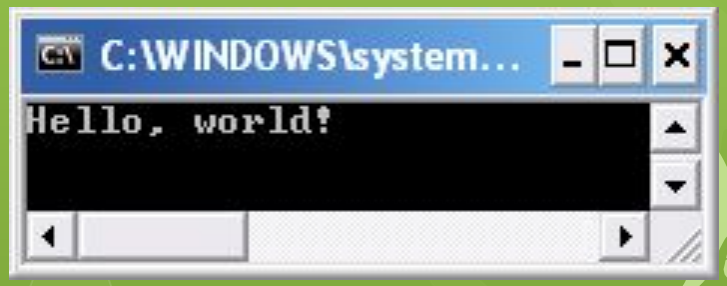
Ядро Unix, Linux, Android, … реалізовані мовою С.

Історичні етапи розвитку мови C++

Історичний етапи розвитку	Рік
Мова BCPL	1966
Мова B (оригінальна розробка Томпсона під UNIX)	1969
Мова C (Денніс Рітчі)	1972
C с класами (Б'ярн Страуструп)	1980
C84	1984
Cfront (транслятор, випуск E)	1984
Cfront (транслятор, 1.0)	1985
Множинне/віртуальне наслідування (успадкування)	1988
Узагальнене програмування (шаблони)	1991
ANSI C++ / ISO-C++	1996
ISO/IEC 14882:1998	1998
ISO/IEC 14882:2003	2003
C++/CLI (для підтримки .NET)	2005
TR1	2005
C++11	2011
C++14	2014

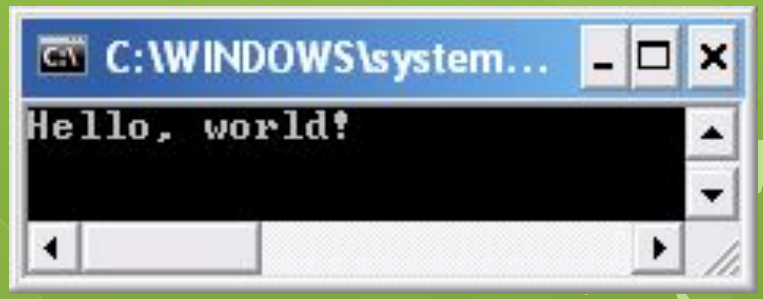
Перша програма мовою С

```
#include <stdio.h>
#include <conio.h>
int main()
{
    printf("Hello, world!\n");
    getch();
    return 0;
}
```



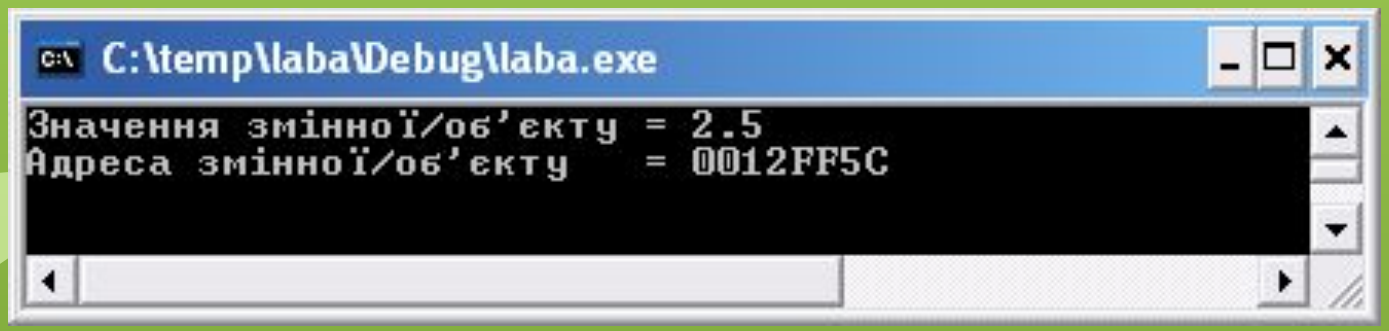
Перша програма мовою C++

```
#include <iostream>
int main()
{
    std::cout<<"Hello, world!\n";
    std::cin.get();
    return 0;
}
```



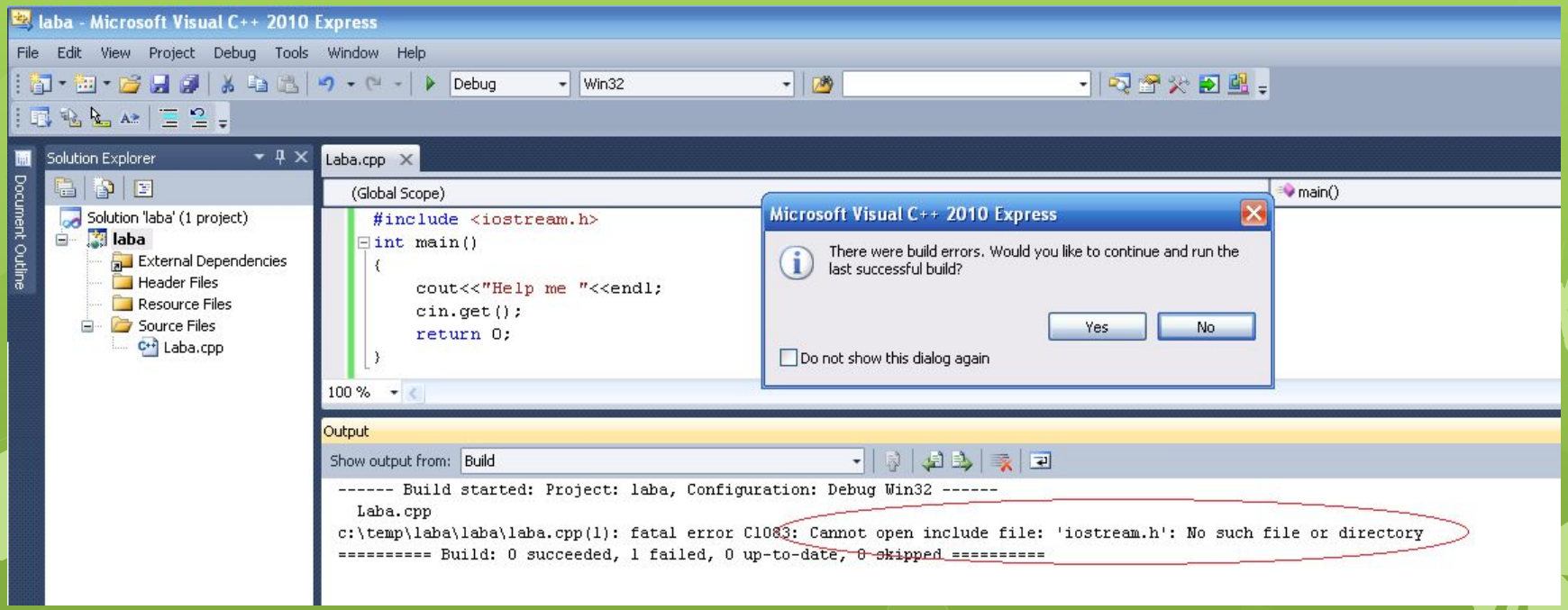
Налаштування виводу кирилиці

```
#include <iostream>
int main()
{
    setlocale(LC_ALL, "Ukr");
    double x= 2.5;
    std::cout<<"Значення змінної/об'єкту = "<<x<<std::endl;
    std::cout<<"Адреса змінної/об'єкту   = "<<&x<<std::endl;
    std::cin.get();
    return 0;
}
```



Застосування iostream.h

Сучасні компілятори не підтримують застарілу бібліотеку C++ <iostream.h>
Ви отримаєте наступну помилку при спробі підключення # include <iostream.h>



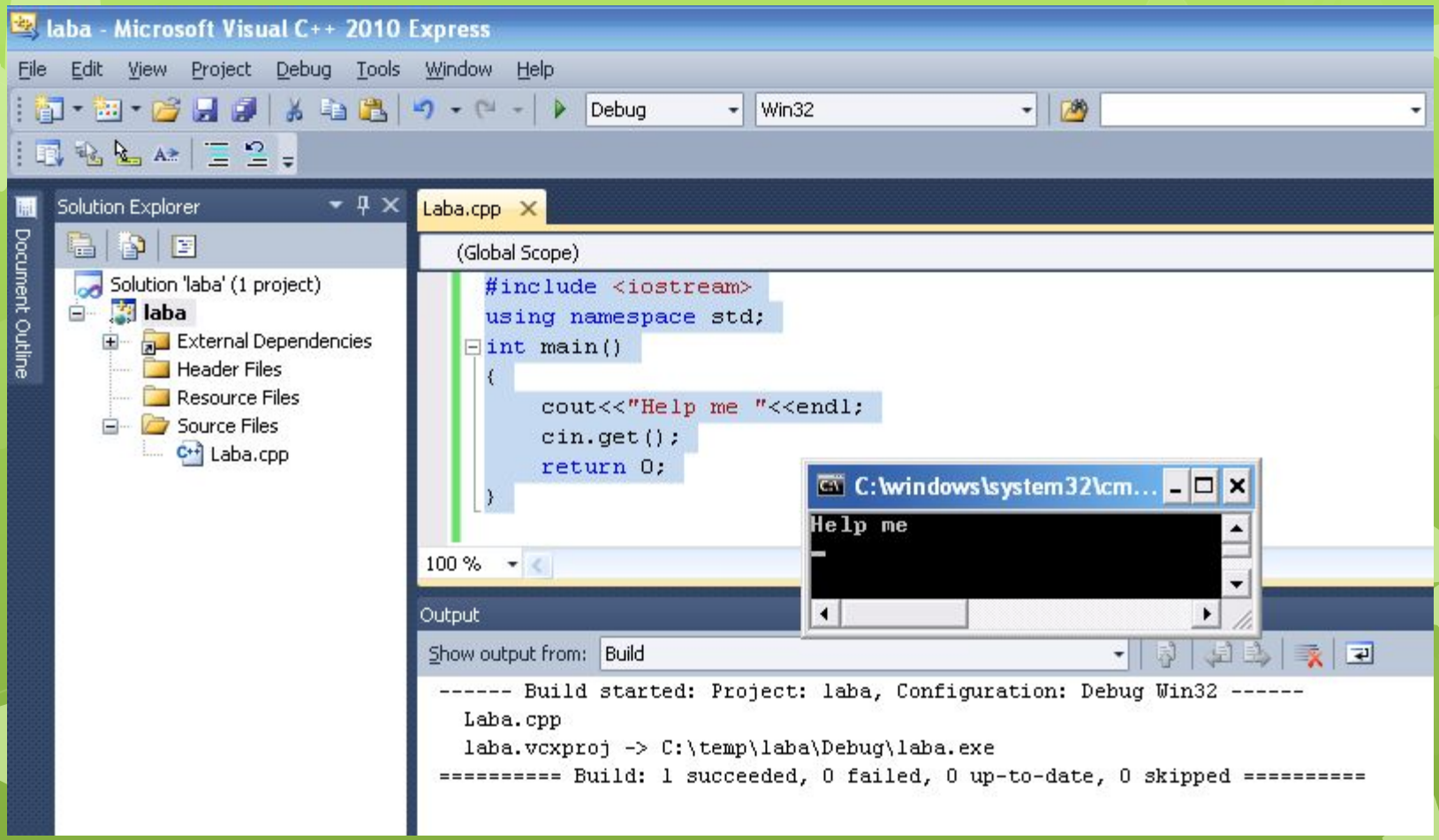
Щоб запустити приклад (наприклад, з деякого підручника), який застосовує цю бібліотеку, необхідно:

- Замінити <iostream.h> на нову бібліотеку <iostream>
- Добавити підключення «Простору імен» std наступною конструкцією: using namespace std;
- Наприклад, попередня програма набуде наступного вигляду:

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Help me "<<endl;
    cin.get();
    return 0;
}
```

Примітка: Не слід зловживати конструкцією using namespace std; Оскільки вона «вбиває» усю ідеологію застосування «Простору імен». Детальніше в лекції «Створення/Застосування namespace (Простір імен)»

Запустивши цю програму (кнопка F5), отримаємо наступний результат:


laba - Microsoft Visual C++ 2010 Express
File Edit View Project Debug Tools Window Help
Debug
Win32
Solution Explorer
Solution 'laba' (1 project)
laba
External Dependencies
Header Files
Resource Files
Source Files
Laba.cpp
Document Outline
Laba.cpp
(Global Scope)
#include <iostream>
using namespace std;
int main()
{
cout<<"Help me "<<endl;
cin.get();
return 0;
}
100 %
C:\windows\system32\cm...
Help me
Output
Show output from: Build
------ Build started: Project: laba, Configuration: Debug Win32 ------
Laba.cpp
laba.vcxproj -> C:\temp\laba\Debug\laba.exe
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========

Запуск програми на виконання

Існує два основних методи виконати програму

- Створити файл (*.exe), який виконується (executable). Для цього застосовують компілятор.
- За допомогою транслятора порядково виконувати програму без створення executable файла.

Примітка: мови програмування поділяють на ті, які компілюються (Ci, C++,…), інтерпретовані мови (perl, python, Matlab,....) та змішані.

- Транслятор
- Компілятор
- Інтерпретатор

Трансляція

- Транслятор - програма або технічний засіб, що виконує трансляцію програми.
- Трансляція програми – перетворення (переклад) програми, представленої на одній із мов програмування, в іншу мову програмування (у тому числі в машину мову).
- Наприклад, мова C++ набула швидкої популярності тому, що Б’ярн Страуструп створив і підтримував транслятор мови C++ в C, в той час, коли ще компіляторів для C++ не існувало.

Компіляція

- Компілятор (різновид транслятора) - програма або технічний засіб, що виконує компіляцію.
- Компіляція (різновид трансляції) - трансляція програми, створеної мовою оригіналу високого рівня, в еквівалентну програму, близьку до машинного коду (абсолютний код, об'єктний модуль, іноді на мову асемблера).
- Компілювати - проводити трансляцію машинної програми з предметно-орієнтованої мови на машинно-орієнтовану мову.

Мови які компілюються : C, C++, Objective-C, Fortran, Swift, Delphi, тощо.

Інтерпретатор

- Інтерпретатор (різновид транслятора) - програма або технічний засіб, що виконує інтерпретацію.
- Інтерпретація - пооператорний (порядковий) аналіз, обробка та виконання програми або запиту (на відміну від компіляції, при якій програма транслюється без її виконання).

Примітка: якщо в програмі в 9-тому рядку буде синтаксична помилка, то перших 8 рядків буде виконано а потім програма зупиниться.

Інтерпретаторні мови: bash, perl, pyton, Matlab, тощо.

Основні етапи створення executable в C/C++ (компіляція програми у загальному сенсі)

Назва етапу	Опис	Вхідні файли	Вихідні файлі	Примітка
Препроцесорна обробка (для С і для С++)	Виконуються директиви препроцесора (#include, #define,….)	file1.h …. fileN.h file1.cpp (.c) …. fileN.cpp (.c)	file1.cpp (.c) …. fileN.cpp (.c) *Змінені *.cpp за правилами препроцесора.* **.h файли будуть вставлені в *.cpp*	
Розбір шаблонів (тільки для С++)	Шаблони (template) замінюються на конкретну реалізацію відповідно до типу	file1.cpp (.c) …. fileN.cpp (.c) *(файли з шаблонами)*	file1.cpp (.c) …. fileN.cpp (.c) *(файли без шаблонами)*	Якщо в початкових файлах шаблони відсутні, цей етап пропускається.
Компіляція	Перевірка синтаксису та створення об’єктних файлів *.cpp (.c) транслюється в *.obj (.o)	file1.cpp (.c) …. fileN.cpp (.c)	file1.obj (.o) …. fileN. obj (.o)	Компіляція успішна тільки у випадку відсутності синтаксичних помилок
Лінковка (компоновка)	Усі об’єктні файли «збираються» (лінкуються) в один файл який запускається (executable)	file1.obj (.o) …. fileN. obj (.o)	file.exe	

Де починається C++?
(«Невеличке» забігання наперед)

C++ починається, коли ми переходимо до класів (class), а значить переходимо до об'єктно-орієнтованого програмування (ООП), а відповідно до основних його постулатів:

- Інкапсуляція
- Наслідування
- Поліморфізм

Клас є просто представлення типу об'єкта; його можна представити як план (креслення), що описує об'єкт.

Подібно до того, як один план (креслення) може бути використаний для побудови декількох будівель, окремий клас може бути використаний для створення необхідної кількості об'єктів.

Інкапсуляція

- **Інкапсуляція** (encapsulation) - це механізм, який об'єднує дані з кодом, що обробляє ці дані, а також захищає і те, і інше від зовнішнього втручання або неправильного використання. В об'єктно-орієнтованому програмуванні код і дані можуть бути об'єднані разом; в цьому випадку говорять, що створюється так званий «чорний ящик». Коли коди і дані об'єднуються таким способом, створюється об'єкт (object). Іншими словами, об'єкт - це те, що підтримує інкапсуляцію.
- Засіб реалізації інкапсуляції в C++ це class.

Одне з визначень класу: **Клас** - це механізм, який об'єднує дані з кодом, який обробляє ці дані. Захист даних виконується за допомогою специфікаторів доступу public, protected, private (детальніше у наступних лекціях).

Приклад класу який зберігає дані цілого типу (детальне пояснення цього прикладу у наступних лекціях).

```
#include <iostream>
class Demo
{
    int data;
public:
    int getData() const {return data;}
    void setData( int d) {data=d;}
};
int main()
{
    Demo d;
    d.setData(45);
    std::cout<<d.getData()<<std::endl;
    std::cin.get();
    return 0;
}
```

Наслідування

- Наслідування (inheritance) - це процес, за допомогою якого один об'єкт може набувати властивостей іншого з можливістю розширити або перевизначити властивості базового об'єкту.

Поліморфізм

- Поліморфізм (від грецького polymorphos) - це властивість, яка дозволяє одне і те ж ім'я використовувати для вирішення двох або більше схожих, але технічно різних завдань. Метою поліморфізму, стосовно об'єктно-орієнтованого програмування, є використання одного імені для завдання загальних для класу дій.
- У більш загальному сенсі, концепцією поліморфізму є ідея «один інтерфейс, безліч реалізацій методів». Це означає, що можна створити загальний інтерфейс для групи близьких за змістом дій.
- *Наприклад, навчившись керувати одним легковим автомобілем (тобто Ви освоїли інтерфейс автомобіля) Ви можете керувати також і іншими легковими автомобілями. І Вас «не цікавить», що, наприклад, коробка передач чи гальма у різних автомобілях технічно реалізовано по різному, Ви просто їх використовуєте.*

Приклад наслідування/поліморфізму
(детальне пояснення прикладу у наступних лекціях)

```
#include <iostream>
class Point
{
    int x,y;
public:
    int getX() const {return x;}
    int getY() const {return y;}
    void setXY( int _x, int _y) {x=_x; y=_y;}
    virtual void print() const {std::cout<< "x="<<x<<" y="<<y;}
};
class ColorPoint : public Point
{
    int color;
public:
    int getC() const {return color;}
    void setC(int c)  {color =c;}
    virtual void print() const {Point::print(); std::cout<<" color="<<color;} // тут поліморфізм
};
void printPoint (Point& point) { point.print(); std::cout<<std::endl;}

int main()
{
    Point p1;
    p1.setXY(5,6);
    p1.print();
    std::cout<<std::endl;
    ColorPoint p2;
    p2.setXY(5,6);
    p2.setC(256);
    p2.print();
    std::cout<<std::endl;
    std::cout<<"\nCall function printPoint:\n"<<std::endl;
    printPoint(p1);
    printPoint(p2);
    std::cin.get();
    return 0;
}
```