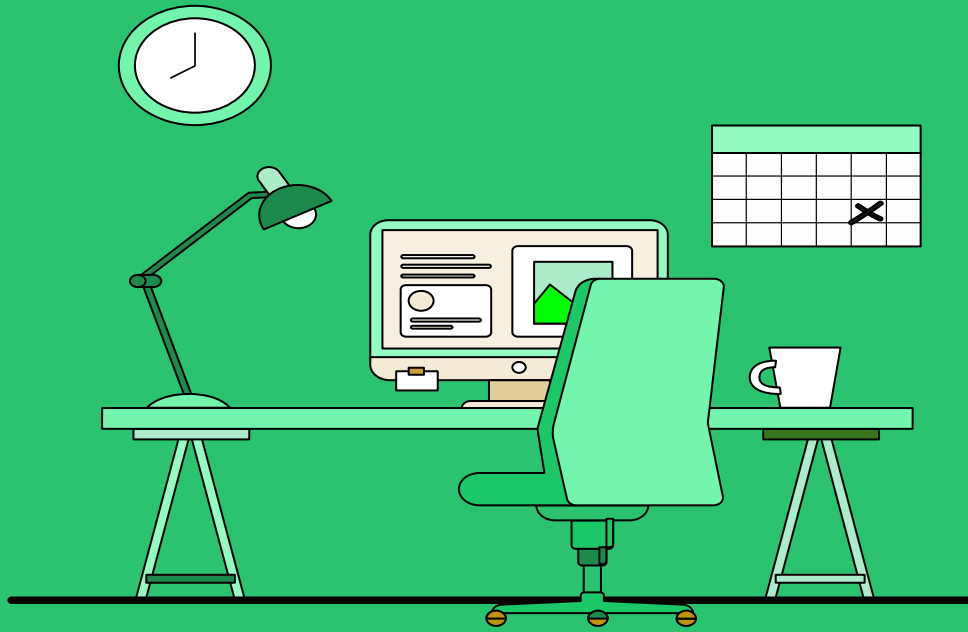
СТРУКТУРНИЙ ПІДХІД

C++

Підсумки частини 2

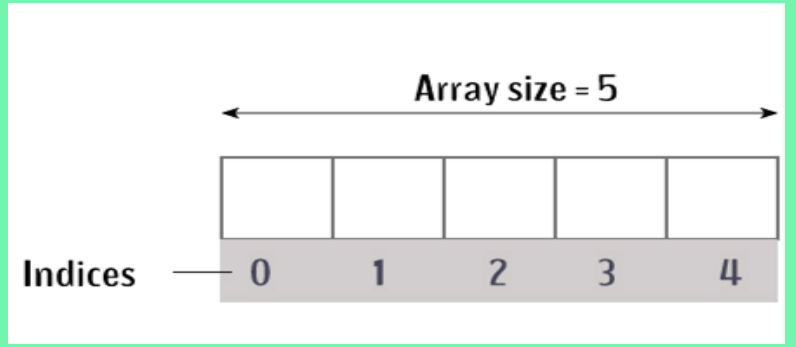


МАСИВИ

Базові типи поєднуються в похідні .
Масив має своє **ім’я , тип, розмір** та розташовується в пам'яті послідовно.

+ [ ] = масив

МАСИВИ



Масив має своє ім’я, а кожен його елемент - свій порядковий номер - індекс .

```
int mas[10];
double array[5];
```

ВВЕДЕННЯ МАСИВУ

```
for(i=0;i<5;i++){
   cin>>mas[i];
}
```

ВИВЕДЕННЯ МАСИВУ

```
for(i=0;i<5;i++){
   cout<<mas[i];
}
```

ЗАПОВНЕННЯ МАСИВІВ

ВВЕДЕННЯ

```
for(i=0; i<10; i++) cin>>mas[i];
```

RANDOM

```
for(i=0; i<10; i++) mas[i]=rand()%10;
```

КОНСТАНТАМ
М

```
mas[3]={5, -4, 555};
```

ФОРМУЛОЮ

```
for(i=0; i<10; i++) mas[i]=i+2;
```

МАСИВ ВИПАДКОВОСТЕЙ

```
rand()%100
rand()%10-5
rand()%100/10
```

```
#include <iostream>
using namespace std;
 int main()
{ int mas[6];
        for( int i=0;i<6;i++)
mas[i]=rand()%10;
     for( int i=0;i<6;i++)
      cout<<mas[i]<<", ";
     return 0;
}
```

ПОШУК МАКСИМУМІВ

Знайти найбільше значення з:

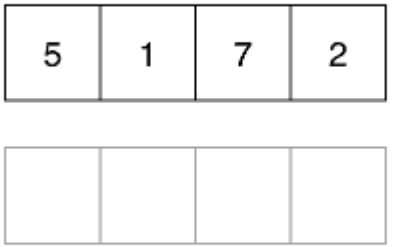
1. двох змінних
2. трьох змінних
3. n змінних
4. **елементів масиву**

```
//пошук максимума
max=mas[0];
for( int i=0;i<6;i++)
if (mmax>mas[i])  max=mas[i];
```

Алгоритми роботи з масивами

Операції над масивами:

- Пошук елементів
- Вставка елементів
- Видалення елементів
- Злиття масивів
- Сортування масивів



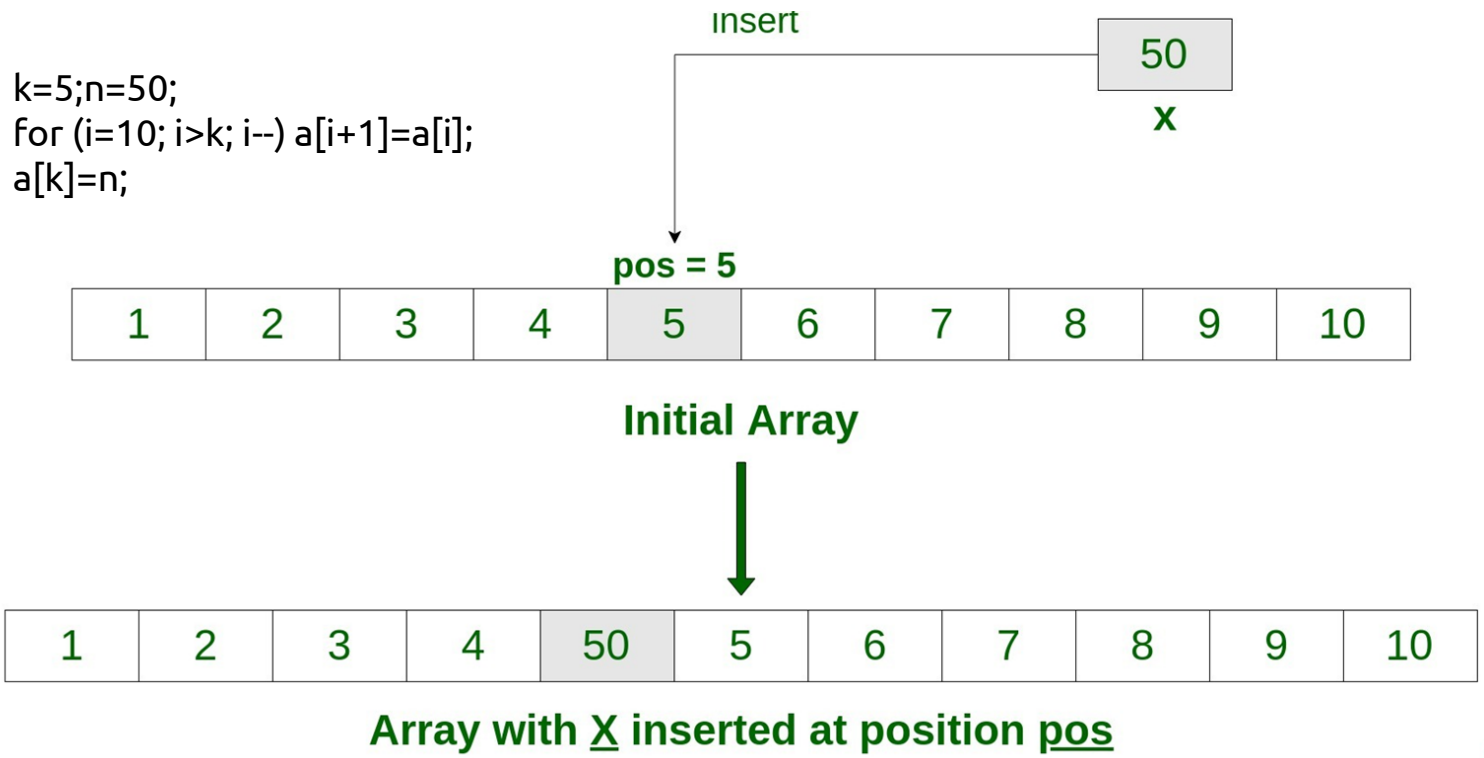
ПОШУК МАКСИМУМІВ

Заповнимо масив та знайдемо максимальне значення в ньому

```
#include <iostream>
using namespace std;
 int main()
{ int mmax; int mas[6];
        for(int i=0;i<6;i++)
              mas[i]=rand()%10;
mmax=mas[0];
for(int i=0;i<6;i++)
if (mmax<mas[i])  mmax=mas[i];
cout<<mmax;
return 0;
}
```

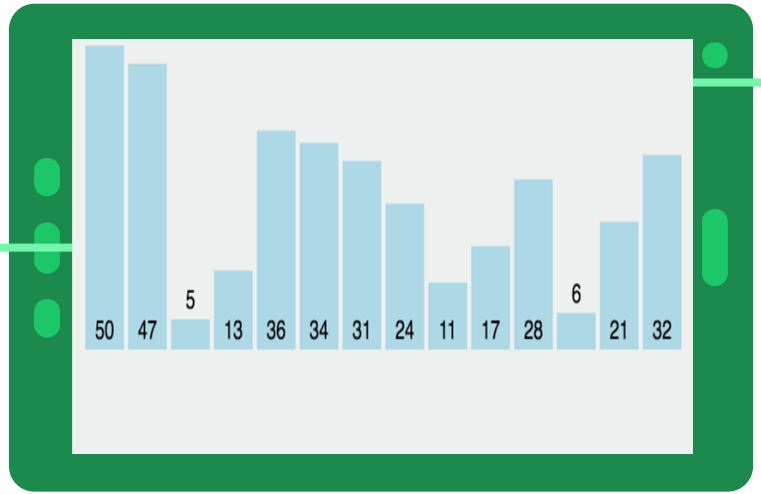
Вставка елемента в масив

```
k=5;n=50;
for (i=10; i>k; i--) a[i+1]=a[i];
a[k]=n;
```



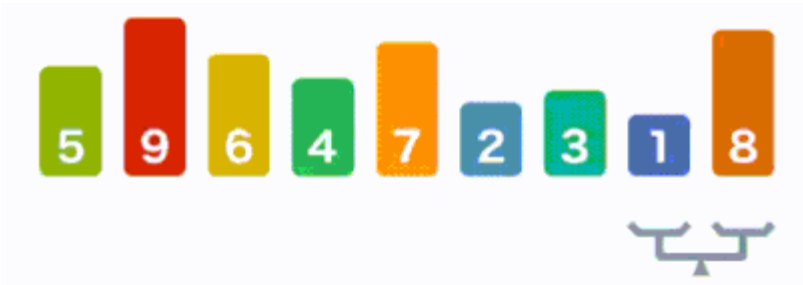
АЛГОРИТМИ СОРТУВАННЯ

ДАНІ



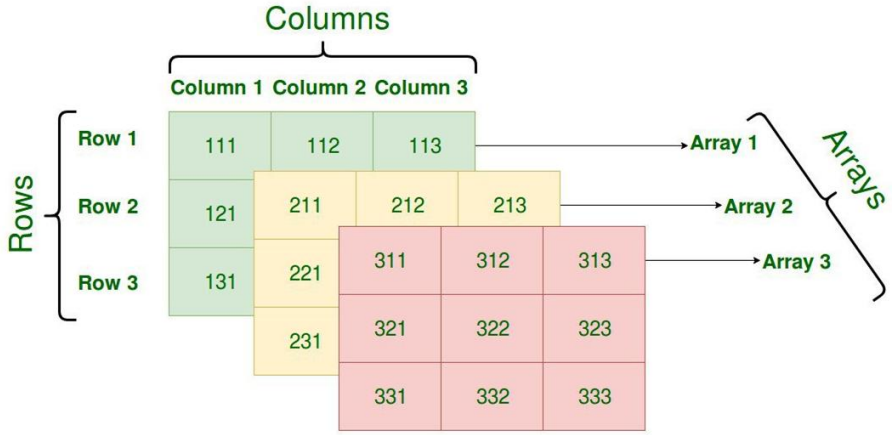
ВПОРРЯЛКОВНІ ДААНІ

```
// Сортування масиву бульбашкою
      for (int i =0 ; i < 10; i++)
   {
       for (int j = i; j < 10; j++)     {
           if (mas[j] > mas[i])     {
               tmp = mas[i];
               mas[i] = mas[j];
               mas[j] = tmp;
                    }
  ;}
```

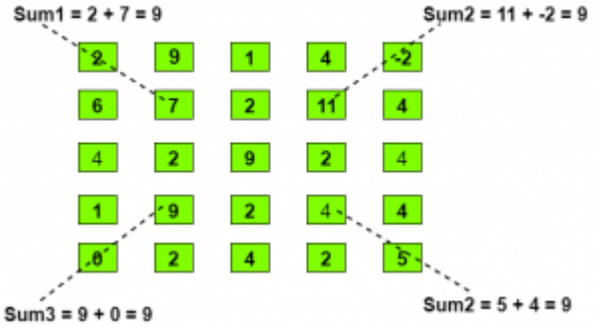


ДВОВИМІІРНІ МАСИВИ

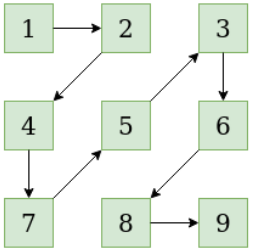
Двовимірні масиви легше уявляти як таблицю.
Це абстракція, бо в пам’яті елементи розташовані послідовно



Працюємо з матрицями



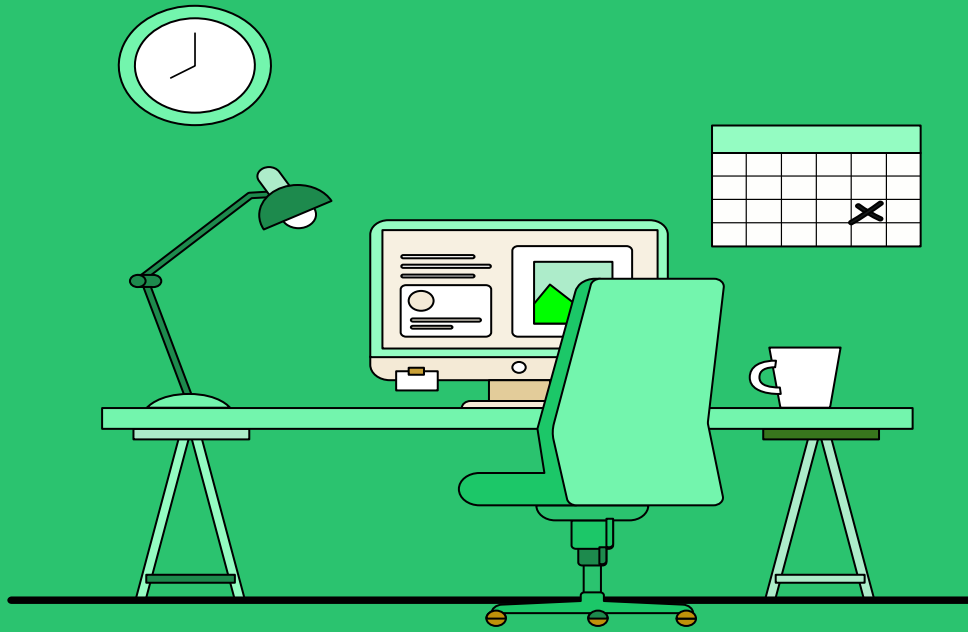
Input Array:



Output: `1, 2, 4, 7, 5, 3, 6, 8, 9`

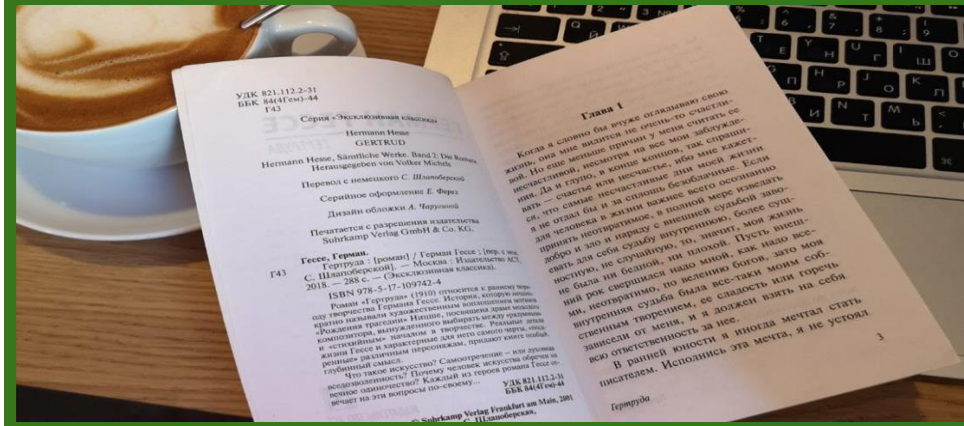
СИМВОЛИ ТА РЯДКИ

C++



СИМВОЛИ CHAR

Символьний тип (char) - тип даних, призначений для зберігання одного символа у певному кодуванні.



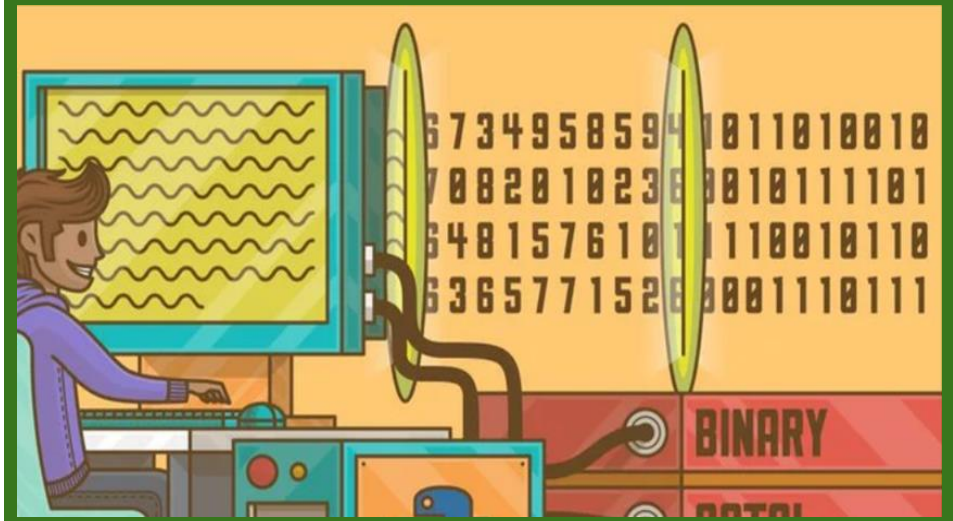

```
char c='₴';
char ch, buk;
```

ВВЕДЕННЯ-ВИВЕДЕННЯ СИМВОЛІВ

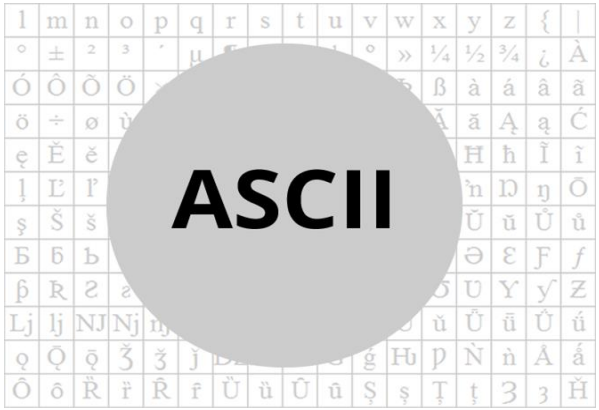

```
char c, ch, buk;
cin>>c;
ch='$';
buk='A';
cout<<c<<' '<<ch<<buk<<' '<<sizeof(c);
```

```
#include <iostream>

using namespace std;

int main() {
    char  x;

        cin>>x;
        cout<< x<<endl;

           cout<<int(x);

        return 0;
}
```

СИМВОЛИ ЗАКОДОВАНІ??



ASCII КОДУВАННЯ

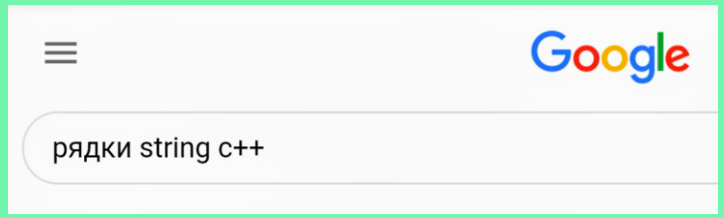


32	пробел	48	0	64	@	80	P	96	`
33	!	49	1	65	A	81	Q	97	a
34	"	50	2	66	B	82	R	98	b
35	#	51	3	67	C	83	S	99	c
36	$	52	4	68	D	84	T	100	d
37	%	53	5	69	E	85	U	101	e
38	&	54	6	70	F	86	V	102	f
39	'	55	7	71	G	87	W	103	g
40	(	56	8	72	H	88	X	104	h
41	)	57	9	73	I	89	Y	105	i
42	*	58	:	74	J	90	Z	106	j
43	+	59	;	75	K	91	[	107	k
44	,	60	<	76	L	92	\	108	l
45	-	61	=	77	M	93	]	109	m
46	.	62	>	78	N	94	^	110	n
47	/	63	?	79	O	95	_	111	o

ТИП ДАНИХ STRING

```
#include <cstring>
або
#include <string.h>
```

```
string s1;
string s2=”Hello”;
```



ПОЄДНУЄМО СИМВОЛИ В РЯДКИ

ТИП STRING

ВВЕДЕННЯ - ВИВЕДЕННЯ

```
string s; string s1=”Hello”;
```

СЛОВА

```
cin>>s;
```

РЯДКИ

```
getline(cin, s);
```

БІБЛІОТЕКА

```
#include <cstring>
```

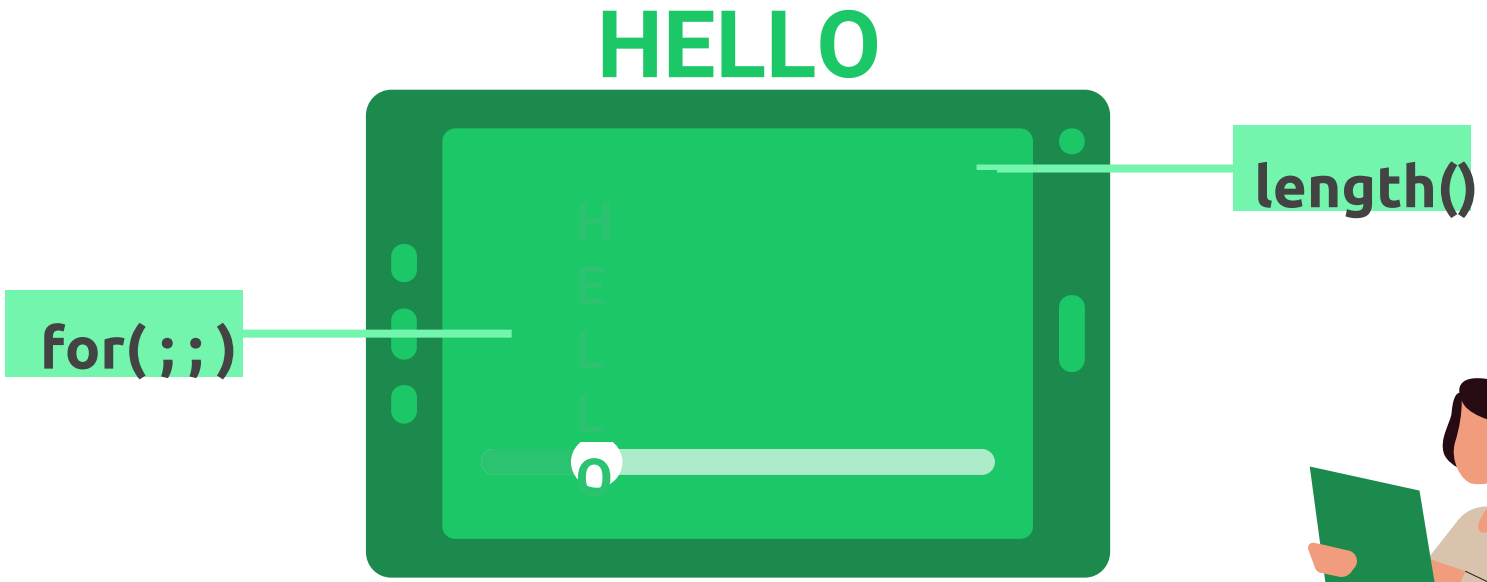
СЛОВО
cin

```
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
string s="Hello", s2;
  cin>>s2;
    cout<<s2<<endl;
      cout<<s<<endl;

return 0;
}
```

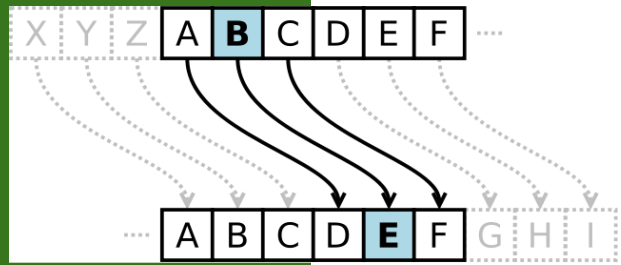
РЯДОК
getline()

```
#include <iostream>
#include <cstring >
using namespace std;

int main()
{
string s="FiveOne", s2;
  getline(cin, s2);
    cout<<s2<<endl;
      cout<<s<<endl;

return 0;
}
```

ВИВЕДЕННЯ РЯДКІВ ПОСИМВОЛЬНО




```
for(int i=0; i<=s.length(); i++)
      cout<<s[i]<<endl;
```



ШИФР ЦЕЗАРЯ

```
#include <iostream>
#include <string.h>
using namespace std;
 int main()
{string s="ABCDE";
string s1="";
int sdvig=2;
for(int i=0; i<s.length();i++)
{if (s[i]>'Z')
s1+=char(int(s[i])+sdvig - 28);
    else
s1+=char(int (s[i])+sdvig);
}
cout<<s1;
}
```

2 способи опису рядків:

МАСИВ СИМВОЛІВ

```
char s [100];
```

ТИП STRING

```
string s;
```

char c;
char c='Q';

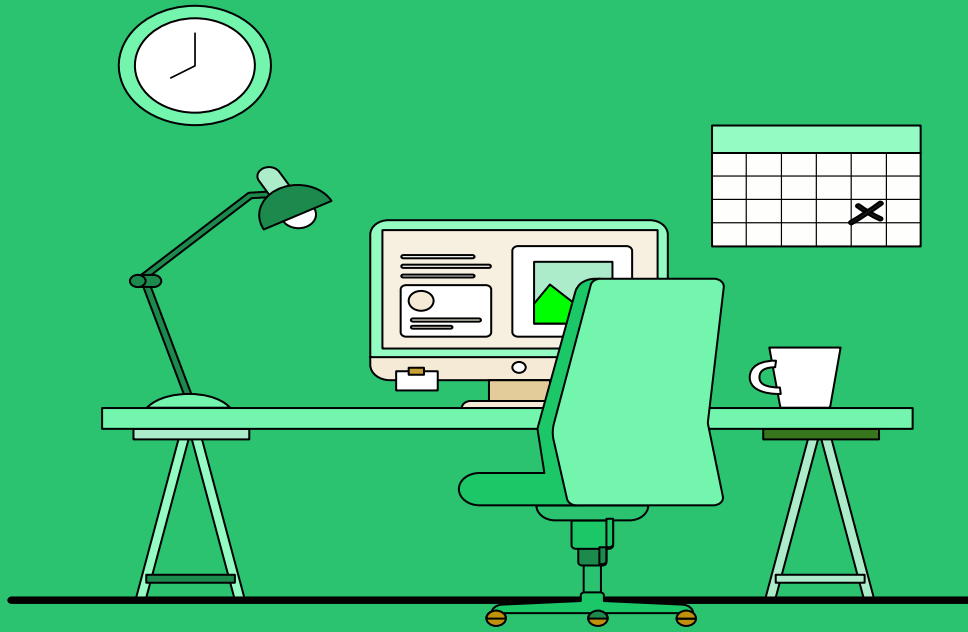
char []

```
char str1 [10];
char str2 [10] = "Hello";
char str5 [] = "Very long
  line";
char * str6;
```

МАСИВ СИМВОЛІВ

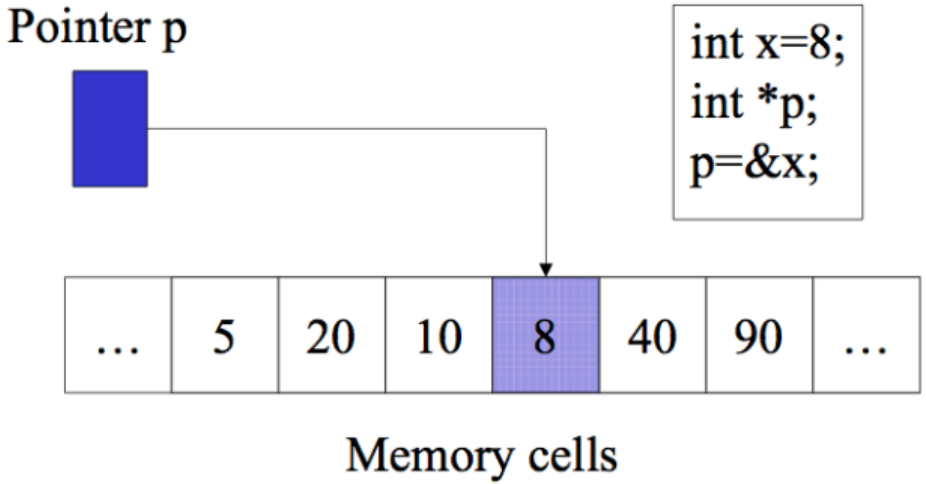
ВКАЗІВНИКИ

C++



Вказівники

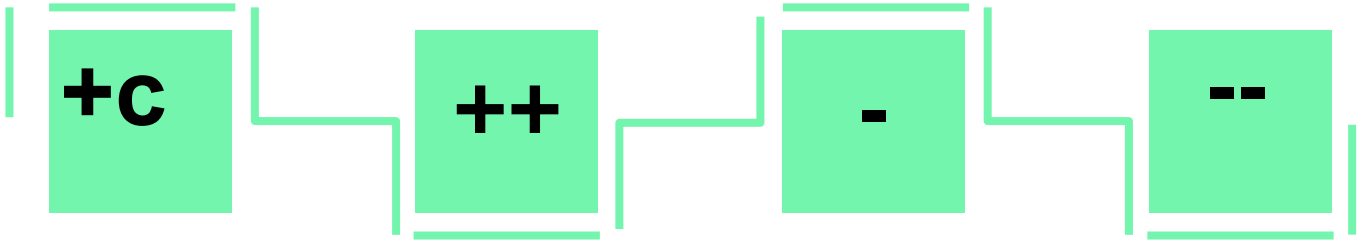
Вказівник - це змінна, значенням якої є адреса пам'яті, по якому зберігається об'єкт певного типу (інша змінна).



ОПЕРАЦІЇ З ВКАЗІВНИКАМИ

інкремент

декремент

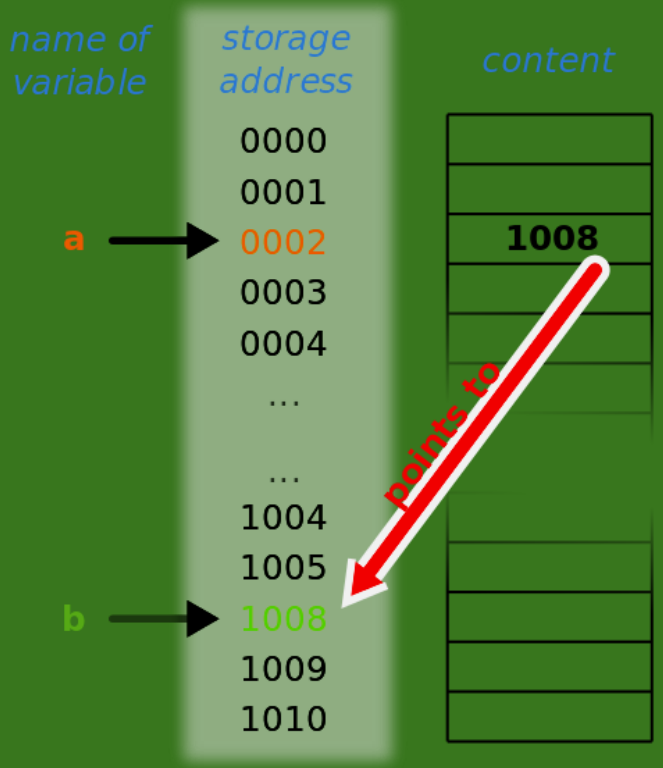


додавання і віднімання покажчиків з константою

різниця вказівників

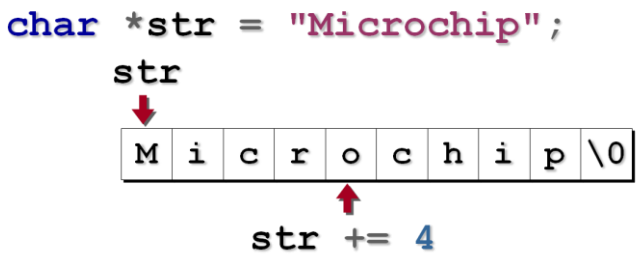
РІЗНИЦЯ ДВОХ ВКАЗІВНИКІВ

Різниця двох вказівників - це різниця їх значень, поділена на розмір типу в байтах.



ВКАЗІВНИКИ ТА РЯДКИ

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {

char s1[]="Finish";
char *ptr=&s[0];
for (int i=0; i<strlen(s1);i++, ptr++)
cout << *ptr << endl;

        return 0;
}
```



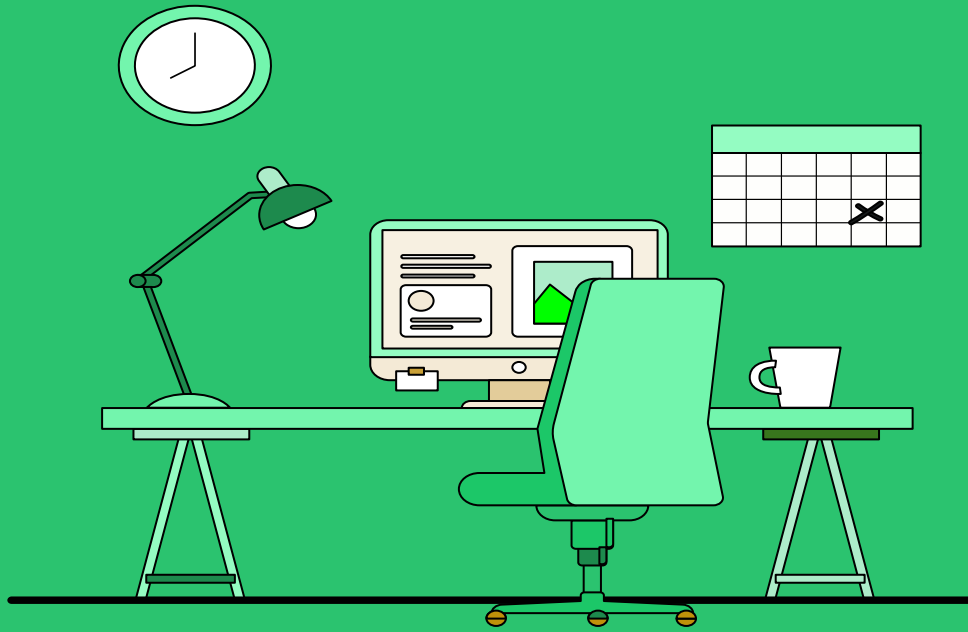
ПОСИЛАННЯ

```
#include <iostream>
using namespace std;
int main()
{
int t = 13;
int &r = t;
cout << "Було  t:" << t; r += 10;
cout<<"\n Стало t:" << t;
return 0;
}
```

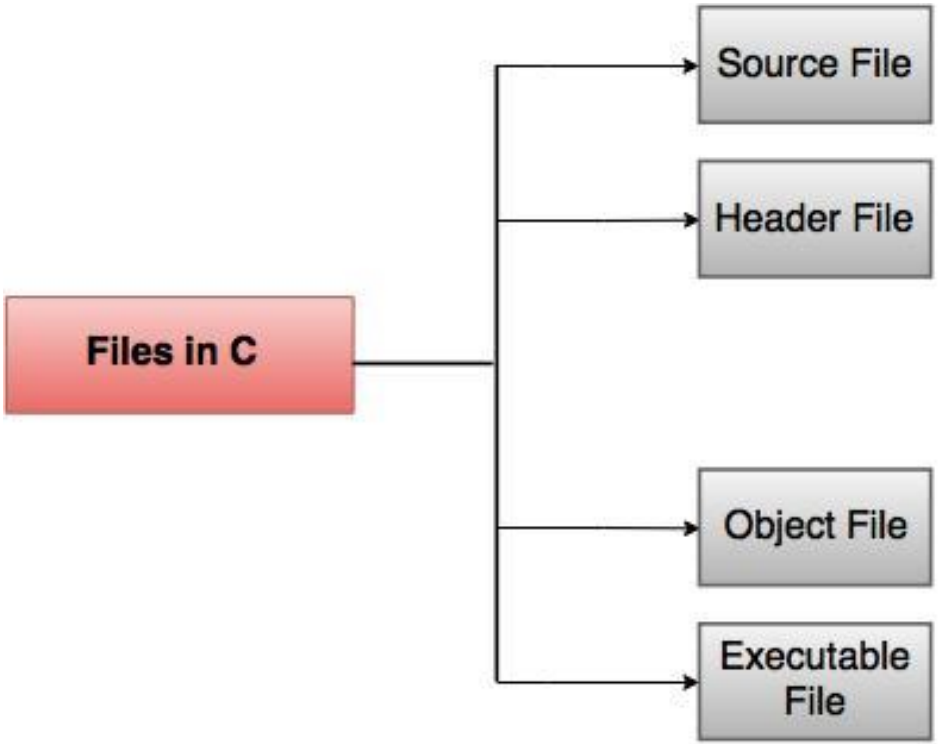
int &r

ФАЙЛИ

C++

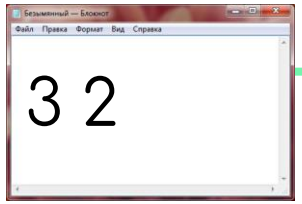


ФАЙЛИ

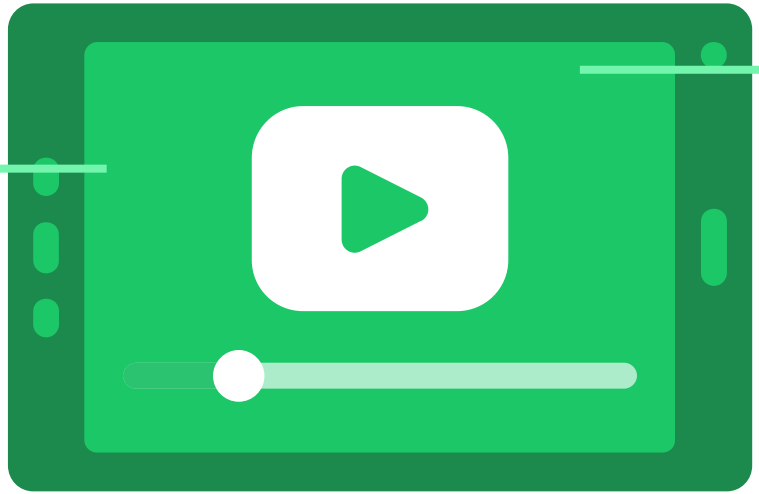


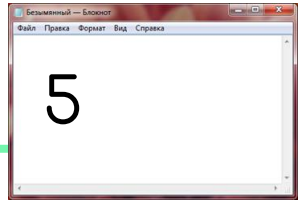
Важливою можливістю мови С ++ є роздільна компіляція.

ФАЙЛОВЕ
ВВЕДЕННЯ- ВИВЕДЕННЯ



INPUT.TXT





OUTPUT.TXT

a+b

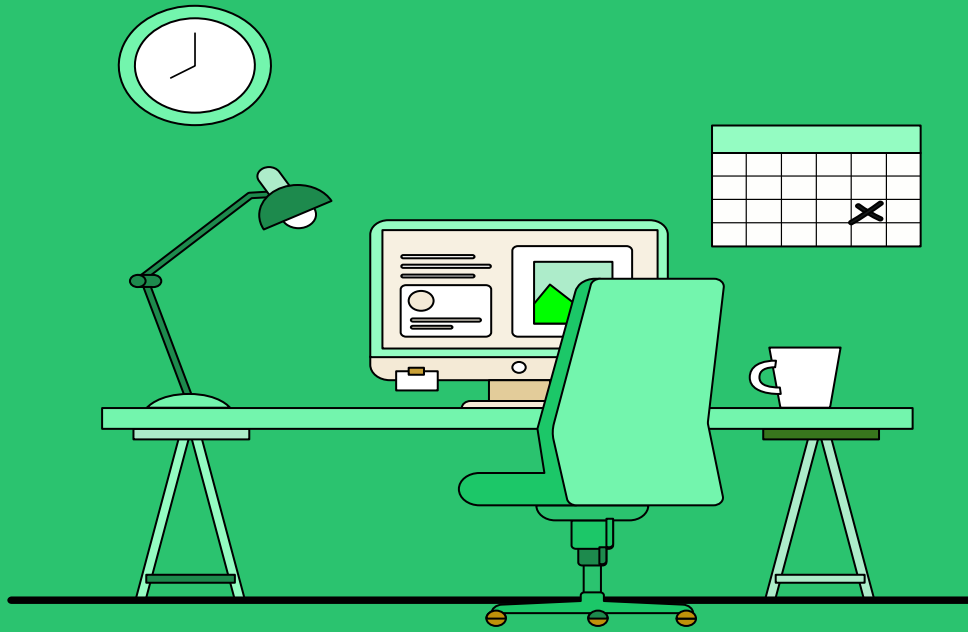
ДОДАЄМО ВВЕДЕННЯ

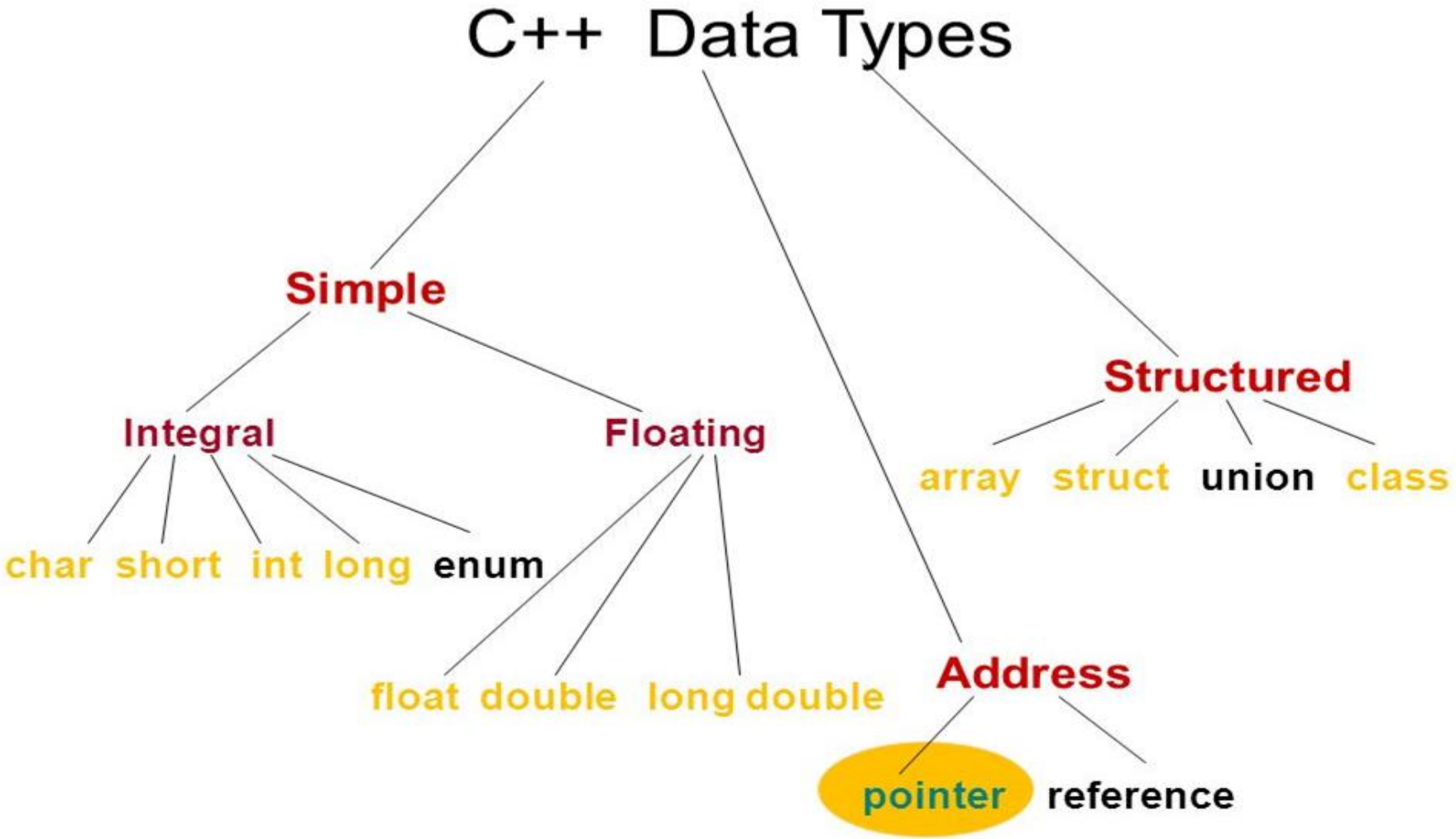
```
#include<fstream>
using namespace std;

int main()
{
    int a,b;

    ifstream cin("input.txt");
    ofstream cout("output.txt");

    cin>> a >> b;
   cout << a+b;
       cout.close();
return 0;
}
```

СТРУКТУРИ ДАНИХ

C++

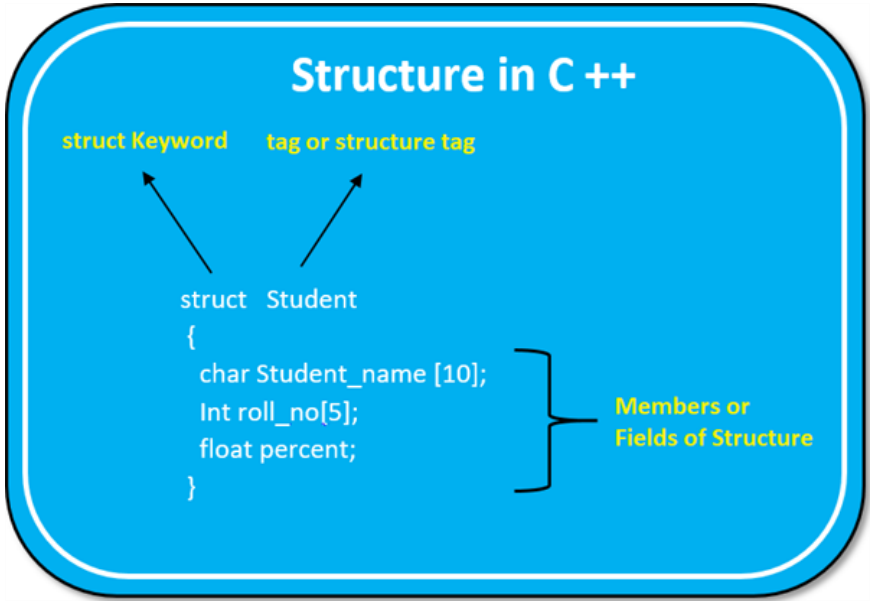
37




C++ Data Types
Simple
Integral
Floating
char
short
int
long
enum
float
double
long double
Structured
array
struct
union
class
Address
pointer
reference

Структури

Структура - це сукупність змінних різних типів даних під єдиною назвою.




```
struct Person
{
     char name[50];
     int age;
   };
```

```
int main()
{
     Person p1;
     cin.get(p1.name, 50);
     cin>>p1.age;
}
```

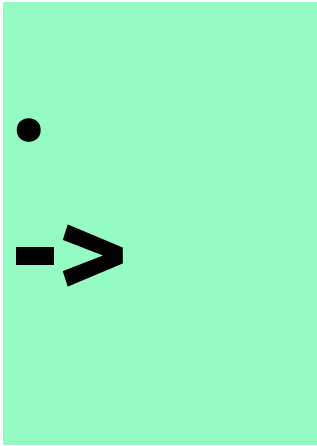
Доступ до елементів структур

Доступ до елементів структур може здійснювати двомісні операції: операцією точки (.) або операцією стрілки (->).

Якщо доступ здійснюється через об'єкт, то використовується точка
cout<<p.age;

Якщо доступ здійснюється за допомогою вказівника на об'єкті, використовується операційна стрілка:

```
p->age = 32;
```



Масив структур

```
#include <iostream>
using namespace std;
struct Person
{
    char name[50];
    int age;
   };

int main()
{
    Person p1[3];
    for(int i=0;i<3;i++)
    {
    cin.get(p1[i].name, 50);
    cin >> p1[i].age;
    cin.get();
    }
 for(int i=0;i<3;i++)

{cout << "Name: " << p1[i].name << endl;
    cout <<"Age: " << p1[i].age << endl;
}
    return 0;
}
```

Структури і функції

Перевагою використання структур є можливість передати всю структуру в функцію, яка повинна працювати з її елементами

```
#include <iostream>
 struct Employee
{short id;
    int age;
    double salary;
};
void printInformation(Employee employee)
{
    std::cout << "ID: " << employee.id;
    std::cout << "Age: " << employee.age;
    std::cout << "Salary: " << employee.salary;
}
 int main()
{    Employee john = { 21, 27, 28.45 };
    Employee james = { 22, 29, 19.29 };
     printInformation(john);
     printInformation(james);
     return 0;
}
```

Вказівники на структури

Як і на будь-який інший тип, на структури можна вказати за власним типом вказівників:

```
struct movies_t {
    заголовок рядка;
    int рік;
};
movies_t amovie;
movies_t * pmovie;
```

Вказівники на структури

Використовуючи вказівник на структуру, можна отримати доступ до її елементів.

Два способи

застосування опису операцій пошуку:

- **(*вказівник _на_структуру).ім'я_елемента**

використання операцій -> (операція стрілка):

- **вказівник _на_структуру->ім'я_елемента**

```
struct person kate = {31, "Kate"};
    struct person * p_kate = &kate;

    char * name = p_kate->name;
    int age = (*p_kate).age;

    // змінимо елемент age
   p_kate->age = 32;
```

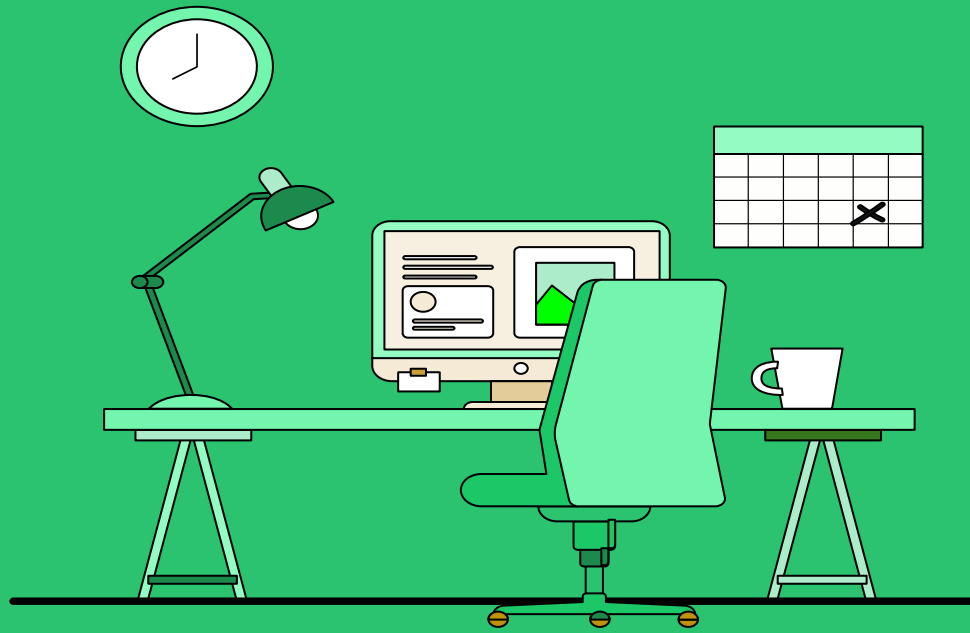
Приклад

Приклад, який поєднує вказівники та структури та служить для введення нового оператора: оператора стрілки (->):

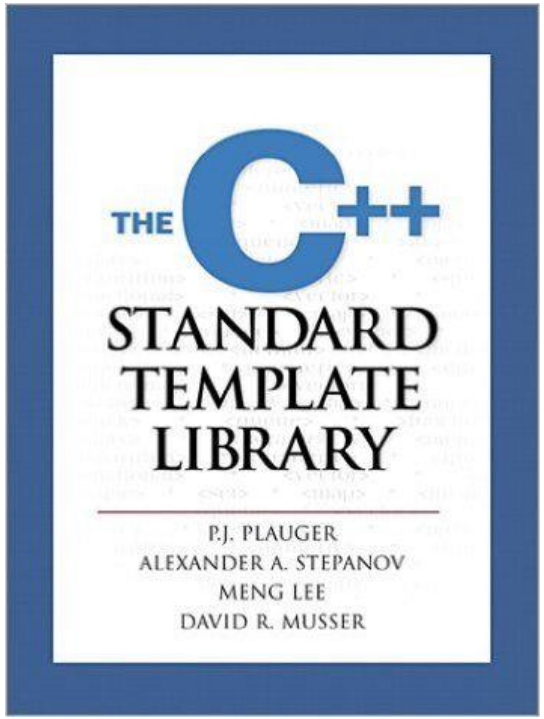
```
string mystr;
 movies_t amovie;
 movies_t * pmovie;
 pmovie = &amovie;

  getline (cin, pmovie->title);
  getline (cin, mystr);
 (stringstream) mystr >> pmovie-
>year;
   cout << pmovie->title;
```

STL бібліотека

C++



Стандартна бібліотека шаблонів (STL) - це набір класів шаблонів C ++ для забезпечення загальних структур даних та функцій програмування, таких як списки, стеки, масиви тощо. Це бібліотека класів контейнерів, алгоритмів та ітераторів.



Компоненти STL:

- Алгоритми
- Контейнери
- Функтори
- Ітератори

Колекції

Контейнер - це об’єкт, що зберігає колекцію інших об’єктів (його елементів).

Контейнер керує місцем для зберігання своїх елементів і надає функції-члени для доступу до них, безпосередньо або через ітератори (посилальні об'єкти з властивостями, подібними до вказівників).

Для використання колекції в коді використовується директива #include <T>,

де T - назва колекції

vector - колекція елементів, збережених в масиві, що змінюється в міру необхідності розміру

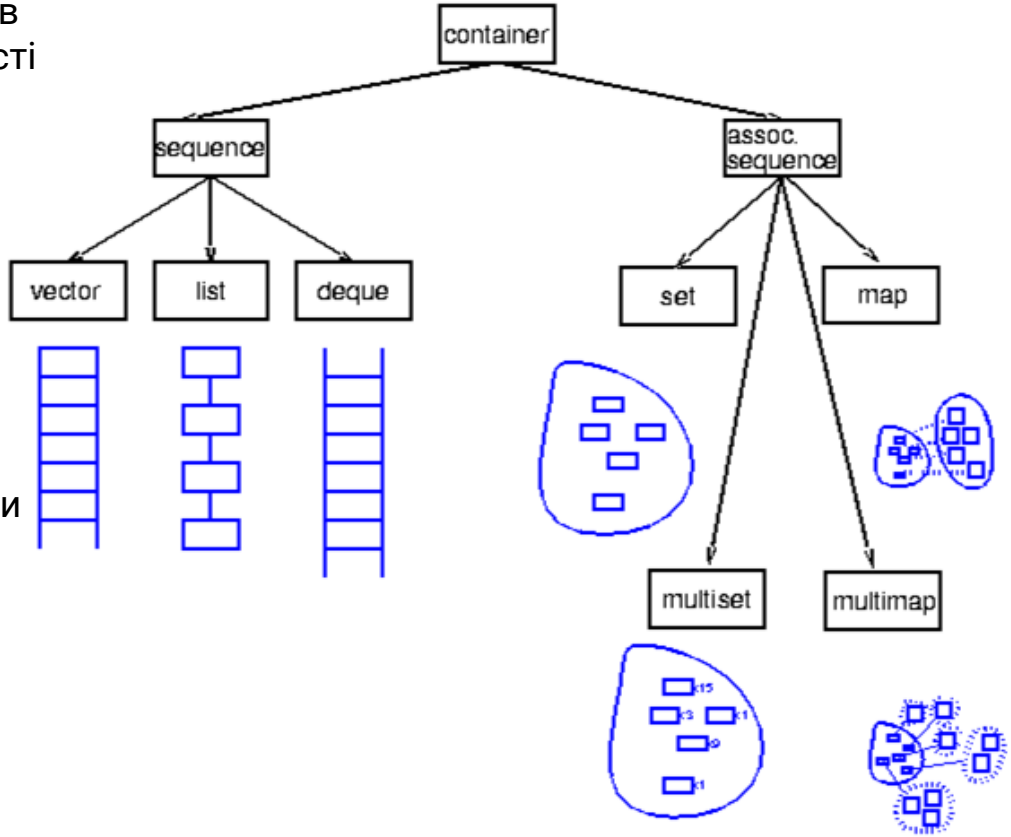
list - зберігає елементи в вигляді зв'язаного списку;

map - зберігає пари виду <const Key, T>, тобто кожен елемент - це пара виду <ключ, значення>

set - відсортована колекція одних тільки значень,д при цьому унікальних - кожен ключ може зустрітися тільки один раз

multimap - map, в якому відсутня умова унікальності ключа

multiset - set з відсутністю умови унікальності ключа.



Алгоритми

Алгоритм заголовка визначає набір функцій, спеціально розроблених для використання на діапазонах елементів. Вони діють на контейнери та забезпечують засоби для різних операцій щодо вмісту контейнерів.

Алгоритми

- Сортування
- Пошук
- Важливі алгоритми STL
- Корисні алгоритми масиву
- Операції розділів
- Числовий
- клас valarray