

# Програмування GUI

На основі мови C++ та фреймворку Qt

## Лекція 3

# Клас QString



Даний клас дозволяє виконувати всі основні операції над рядками: вставка, видалення, пошук підрядка, об'єднання рядків.



Конструктори:

`QString (const char *txt)` – створення рядка на базі масиву символів (C-рядок)



`QString (QString &s)` – створення рядка на основі якого іншого рядка

`QString ()` – створення пустого рядка

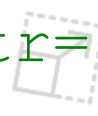


```
QString str ("Hello"); //str=Hello
```

```
QString str1="abc"; //str=abc
```


```
QString str2 (str1); //str=abc
```

```
QString str3; //str=""
```




# Методи та оператори класу QString

1. Метод `int length()` – повертає довжину рядка
2. Метод `bool isEmpty()` – повертає `true`, якщо рядок порожня, тобто не містить символів



```
QString str="abc";  
int n=str.length(); //n=3  
bool ok=str.isEmpty(); //ok=false  
str=""; ok=str.isEmpty(); //ok=true
```

3. Для об'єднання рядків використовуються оператори `+=`, `+`, а також метод `QString &append (const QString &str)`.



```
QString str="1";  
str.append("2"); //str="12«  
str+=3; //str="123«  
str=str+"4"; //str="1234"
```





A  
Z ↓



# Клас QString




- 
4. Для порівняння рядків можна використовувати оператори порівняння `==`, `!=`, `<`, `>`, `<=`, `>=`, А також статичний метод `int compare(const QString &s1, const QString &s2);`
- 

Метод `compare` повертає позитивне значення якщо `s1 > s2`, 0 - якщо `s1 = s2`, і негативне значення, якщо `s1 < s2`.





Значення, що повертається дорівнює різниці зі знаком між кодами перших несовпавших символів в рядках.



```
QString str1="abc", str2="abd";
bool ok=str1==str2; //ok=false
      ok=str1<str2; //ok=true;
      ok=str2>=str1; //ok=true
```

```
int n=QString::compare("abc","akc"); //n=-9 ('b'-'k'=-9)
    n= QString::compare("abc","abb"); //n=1 ('c'-'b'=1)
```



# Клас QString



5. Для пошуку підрядка в рядку можна використовувати наступний метод `int indexOf (QString str, int from=0, Qt::CaseSensitivity cs=Qt::CaseSensitive),`



де `str` – шуканий підрядок,

`from` – початкова позиція пошуку,

`cs` – дозволяє враховувати \ не враховувати регістри символів при пошуку



`Qt::CaseSensitive` – регістр враховується (за замовчуванням)

`Qt::CaseInsensitive` – не враховується



Якщо підрядок знайдено, метод `indexOf ()` повертає номер позиції першого символу підрядка в рядку, інакше метод повертає -1.



# Клас QString



Також існує метод `int lastIndexOf ()` з таким же списком параметрів, який виконує пошук, починаючи з останнього символу у напрямку до першого символу рядка.





Існують перевантаження методів `indexOf` и `lastIndexOf`

```
QString str="abcc";  
int n=str.indexOf('c'); //n=2  
n=str.lastIndexOf('c'); //n=3 так как поиск идет с конца строки  
str="abc_abc";  
n=str.indexOf("bc"); //n=1  
n=str.lastIndexOf("abc"); //n=4  
n=str.indexOf("123"); //n=-1
```





# Клас QString



- 
6. Методи `QString left (uint len)` та `QString right (uint len)` повертають відповідно ліву і праву частину рядка довжиною в `len`.
- 

```
QString str="abc 123";  
QString l=str.left(3); //l="abc"  
QString r=str.right(2); //r="23"
```

- 
7. Для вставки підрядка в рядок можна використовувати метод `QString insert (int position, const QString &str)`,
- 

де `position` – позиція вставки, `str` – рядок, що вставляється

```
QString str="1245";  
str.insert(2, '3'); //str="12345"  
str.insert(0, "12345"); //str="1234512345"
```



# Клас QString



8. Для видалення з рядка символів необхідно використовувати метод `remove`.

a. `QString &remove (int position, int n)` – видаляє `n` символів, починаючи з позиції `position`


b. `QString &remove (const QString &str, Qt::CaseSensitivity cs=CaseSensitive)` - видаляє з рядка всі вирази підстроки `str`.

Є такий же метод і для видалення символів.


```
QString str="abcd";  
str.remove(1,2); //str="ad"  
str="ababab";  
str.remove('a');//str="bbb";  
str="ababab";  
str.remove("ba"); //str="ab"
```




# Клас QString




9. Для заміни в рядку однієї підрядка інший можна використовувати метод `QString &replace` (`const QString &before`, `const QString &after`, `Qt::CaseSensitivity cs=CaseSensitive`),



де `before` – підрядок, що замінюється  
`after` – підрядок, що замінює




Є аналогічний метод для заміни символів.



```
QString str="1 and 2";  
str.replace("1","2"); //str="2 and 2"
```




# Клас QString



10. Для обчислення числа входжень символу або підрядка в рядок потрібно використовувати метод count



```
int count (const QString &str,  
Qt::CaseSensitivity cs=CaseSensitive)
```





```
QString str="ababab";  
int n=str.count('a'); //n=3  
n=str.count("ba"); // b=2
```





# Клас QString



- 
11. Методи `QString toUpper()`, `QString toLower()` повертають копію рядка, що містить всі символи початкового рядка в верхньому або нижньому регістрі відповідно.
- 

```
QString str="aBcDe";  
QString str1=str.toUpperCase(); //str1="ABCDE"  
str1=str1.toLowerCase(); //str1="abcde"
```

- 
12. Метод `std::string toString()` – призначений для перетворення рядка класу `QString` в об'єкт рядка класу `string` стандартної бібліотеки `C++`.
- 

```
QString str="abcd";  
string str2=str.toString(); //str2="abcd"
```



# Клас QString



## Перетворення рядка в число

1. `int toInt (bool *ok=0, int base=10)` – в число типу `int`



2. `uint toUInt (bool *ok=0, int base=10)` – в число типу `uint`

3. `double toDouble (bool *ok=0)` – в число типу `double`



Параметр `ok` вказує на змінну булевого типу, в яку зазначені методи помістять значення `true` при успішному виконанні перетворення, `false` - при невдалому; параметр `base` - задає систему числення числа, що міститься в рядку.




Існують методи перетворення і для інших числових типів.




# Клас QString



## Перетворення рядка в число



```
bool ok;  
QString str="10.1";  
double d=str.toDouble(&ok); //d=10.1, ok=true  
int i=str.toInt(&ok); //i=0, ok=false
```



```
str="-10";  
i=str.toInt(&ok); //i=-10, ok=true  
int j=str.toUInt(&ok); // j=0, ok=false
```



# Клас QString



## Перетворення числа в рядок

З цією метою, як правило, використовують статичний метод `number`.



1. `QString number (int n, int base=10),`  
де `n` – число, яке підлягає перетворенню, `base` - система числення.

```
int n=-255;  
QString str=QString::number(n); //str="-255"
```



2. `QString number (double n, char format='g', int prec=6),`



- де `n` – число, що підлягає перетворенню в рядок,
- `format` – задає формат перетворення.



# Клас QString



## Перетворення числа в рядок

`QString number (double n, char format='g', int prec=6),`



де `n` – число, що підлягає перетворенню в рядок,

`format` – задає формат перетворення.

`'e'` или `'E'` – експонентний (науковий) формат(-3999 -3.99e)

`'f'` или `'F'` – формат з фіксованою точкою (коми)956.078

`'g'` или `'G'` – загальний формат

`prec` – задає число знаків після коми (точність).

У разі формату `'g'` (`'G'`) `prec` задає загальну кількість чисел мантиси числа (ціла і дробова частини).

```
double n=100.0/3.0;
QString str=QString::number(n,'e',2); //str="3.33e+01"
str=QString::number(n,'F',2); //str="33.33"
str=QString::number(n,'g'); //str="33.3333"
```



# Клас QString



Для форматування виведених повідомлень в бібліотеці Qt існує шаблонний символ %n, де n - число від 1 до 99, і метод QString::arg(), який всі шаблонні символи %n, що зустрічаються в рядку з найменшим номером замінює строковим поданням свого аргументу і повертає отриманий рядок, до якої потім можна повторно застосувати метод arg(), щоб зробити підстановку для шаблонних символів з наступним номером. Приклад:

```
QString str1 = tr("Меня зовут %1, мне %2  
лет, я из  
%3").arg(name).arg(age).arg(city);  
QString str2 = tr("Из %1 > %2 следует, что  
%2 < %1").arg(x).arg(y);
```



# Клас QString



Після виведеного значення в методі `arg()` можна вказати необов'язкові параметри: ширину поля, формат (E, G, f) і точність (число знаків після десяткового дробу).



Найкращий метод роботи з символами національних алфавітів реалізує спеціальна функція перекладу `tr()`, за допомогою якої здійснюється інтернаціоналізація додатків.



Домовимося все строкові значення, зазначені в тексті програми, передавати в якості параметра функції `tr()`. Ця статична функція є членом всіх класів бібліотеки Qt, породжених від базового класу `QObject`.



# Клас QString



Для завдання кодування, що використовується функцією перекладу, потрібно створити відповідний кодек і передати його в якості аргументу методу `setCodecForTr ()`.

