# LECTURE 01

# DYNAMIC ARRAY & STRING

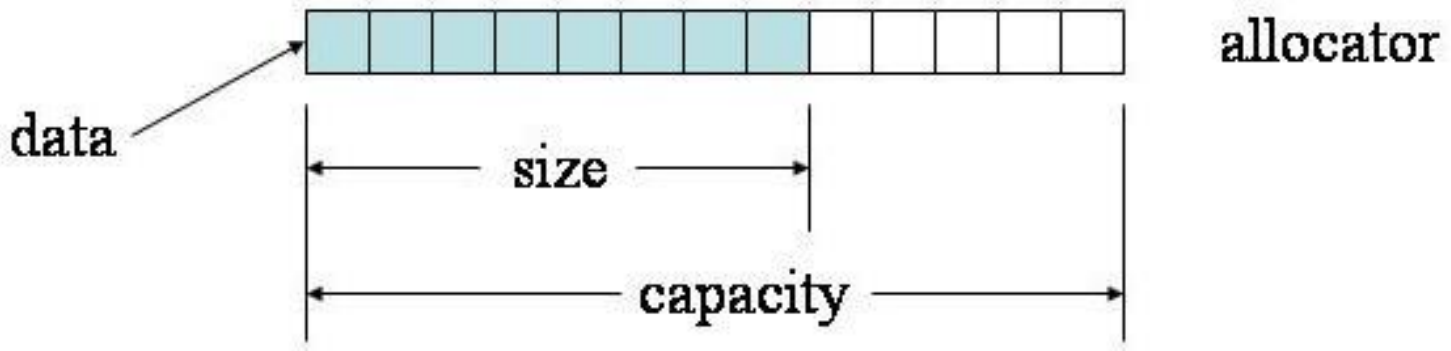



Phạm Nguyễn Sơn Tùng

Email: sontungtn@gmail.com

# Dynamic Array là gì?

**Dynamic Array** là một cấu trúc dữ liệu mảng động, người dùng không cần khai báo trước số lượng phần tử là bao nhiêu.




C++: `vector`

Java: `Arraylist`

Python: `list`

# Cách khai báo sử dụng Dynamic Array

C++

Thư viện:

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

Khai báo:

```
vector<data_type> name;
```

```
vector<int> v;
```

python

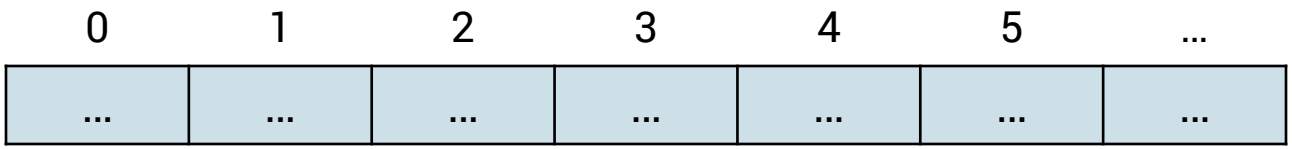
Thư viện:

```
NULL
```

Khai báo:

```
variable = [value1, value2, ..]
```

```
l = []
```



# Cách khai báo sử dụng ArrayList

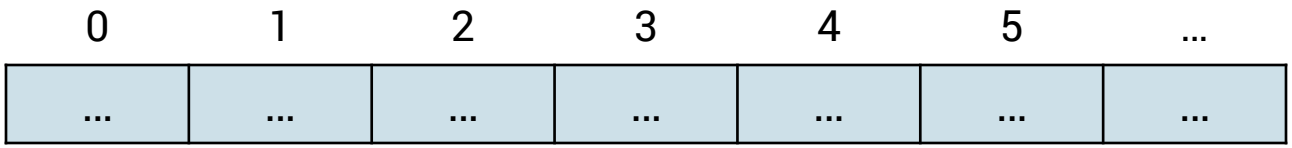

Thư viện:

```
import java.util.ArrayList;
```

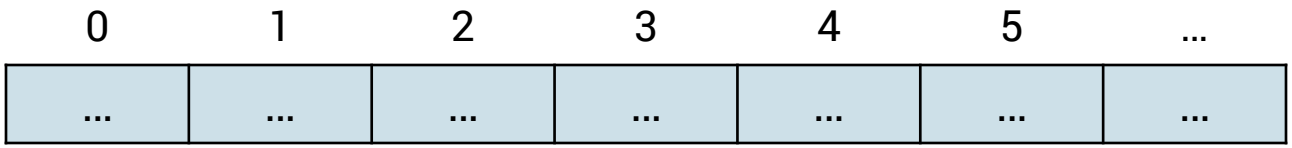
Khai báo:

```
ArrayList<DataType> variable = new ArrayList<DataType>();
```

```
ArrayList<String> a = new ArrayList<String>();
```



# Thêm phần tử vào cuối Dynamic Array




C++

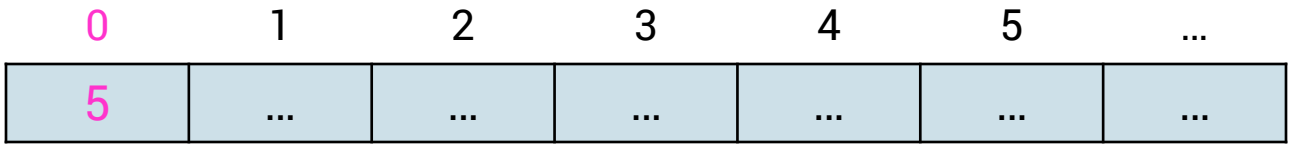
python

## push_back(value)

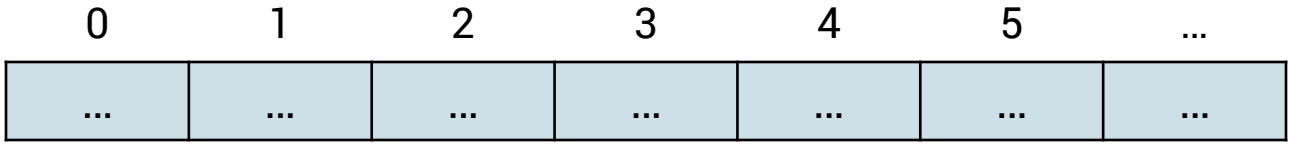
```
vector<int> v;
v.push_back(5);
```

## append(obj)

```
l = []
l.append(5)
```



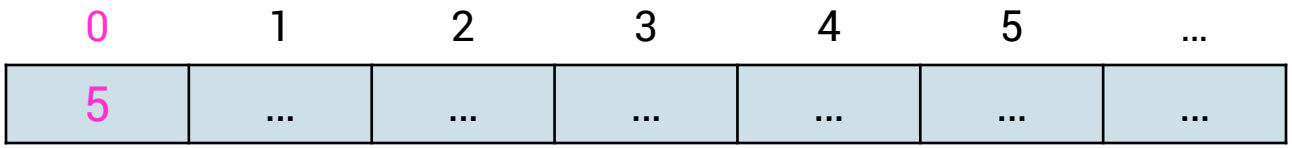
# Thêm phần tử vào cuối ArrayList



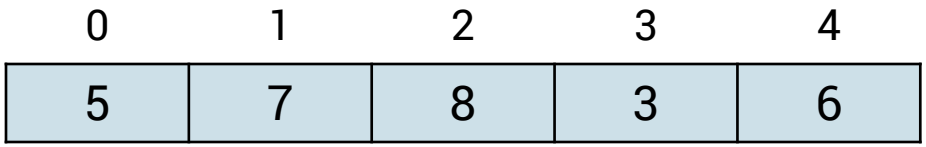


## add(value)

```
ArrayList<Integer> a = new ArrayList<Integer>();
a.add(5);
```



# Lấy phần tử đầu tiên Dynamic Array




C++

## front()

```
int result = v.front();
//hoặc result = v[0];
cout<<result;
```

python

**Dùng vị trí để trả về giá trị đầu tiên.**

```
result = l[0]
print(result)
```

**5**

# Lấy phần tử đầu tiên của ArrayList

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |


**get(index):** Để lấy giá trị phần tử đầu hay cuối của ArrayList, ta sử dụng hàm get để lấy giá trị. Lưu ý chương trình sẽ Throw exception nếu index không thỏa mãn giới hạn của ArrayList (index < 0 hoặc index >= size()).

```
a.get(0);
```

**5**

*Lưu ý: hàm get chỉ trả về giá trị, không thể dùng để cập nhật cho ArrayList.*

# Lấy phần tử cuối cùng Dynamic Array

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |

C++

**back()**

```
int result = v.back();
// result = v.[n-1];
cout<<result;
```

python

**Dùng vị trí để trả về giá trị cuối cùng.**

```
result = l[-1]
print(result)
```

**6**

# Lấy phần tử cuối cùng ArrayList

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |

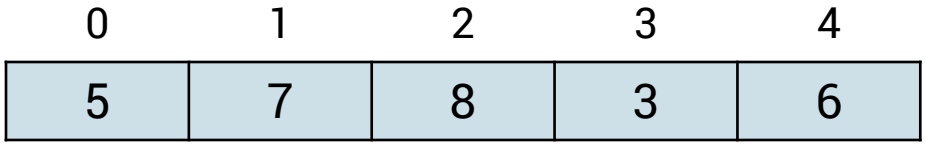

Tiếp tục dùng hàm **get(index)** để lấy phần tử cuối cùng.

```
a.get(a.size() - 1);
```

**6**

*Lưu ý: hàm size lấy số lượng phần tử của ArrayList.*

# Chèn một giá trị vào Dynamic Array




C++

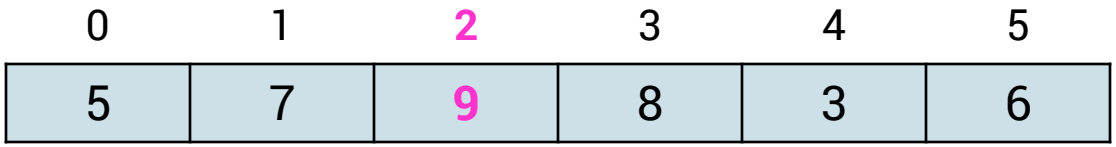
python™

insert(iterator, val): Chèn **một** giá trị.

```
vector<int>::iterator it;
it = v.begin() + 2;
v.insert(it, 9);
```

insert(pos, val): chèn giá trị val vào vị trí pos.

```
l.insert(2, 9)
```



# Chèn một giá trị vào ArrayList

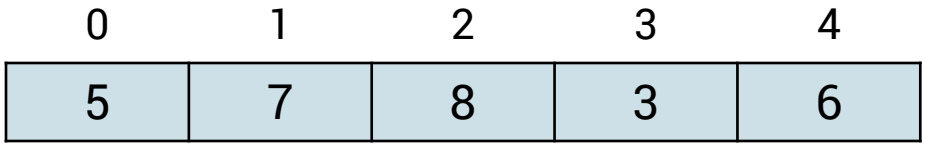
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |


**add(index, value):** Sử dụng lại hàm add có thêm một tham số thứ 2 để chèn một phần tử vào **vị trí index** của ArrayList. Throw exception nếu index không thỏa mãn giới hạn của ArrayList (index < 0 hoặc index > size()).

```
a.add(2, 9);
```

| 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 5 | 7 | 9 | 8 | 3 | 6 |

# Xóa phần tử cuối khỏi Dynamic Array




C++

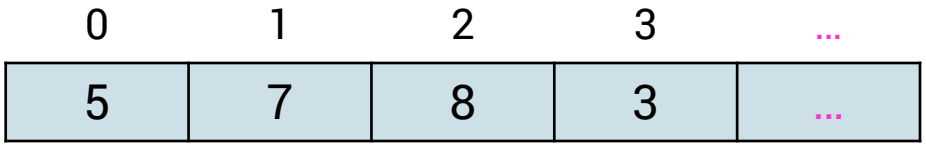
python

## pop_back()

```
v.pop_back();
```

## pop()

```
l.pop()
```



# Xóa phần tử cuối của ArrayList

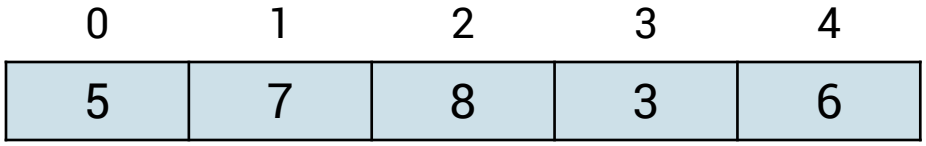
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |


**remove(index):** Để xóa phần tử cuối cùng của ArrayList, ta sử dụng hàm remove để xóa. Lưu ý chương trình sẽ Throw exception nếu index không thỏa mãn giới hạn của ArrayList (index < 0 hoặc index >= size()).

```
a.remove(a.size()-1);
```

| 0 | 1 | 2 | 3 | ... |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | ... |

# Xóa phần tử ở vị trí bất kỳ trong Dynamic Array




C++

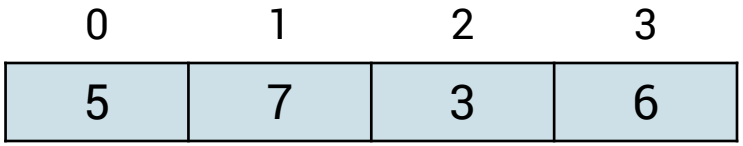
python

erase(iterator): Xóa **một** giá trị.

```
vector<int>::iterator it;
it = v.begin() + 2;
v.erase(it);
```

pop(pos): Xóa **giá trị** trong list ở vị trí bất kì và trả về giá trị bị xóa.

```
l.pop(2)
```



# Xóa phần tử ở vị trí bất kì trong ArrayList

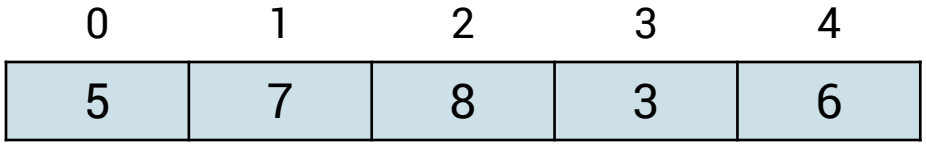
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |


Như đã trình bày ở trên, để xóa phần tử bất trong Array List thì ta sử dụng lại hàm **remove(index).**

```
a.remove(2);
```

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| 5 | 7 | 3 | 6 |

# Xóa toàn bộ các phần tử trong Dynamic Array




C++

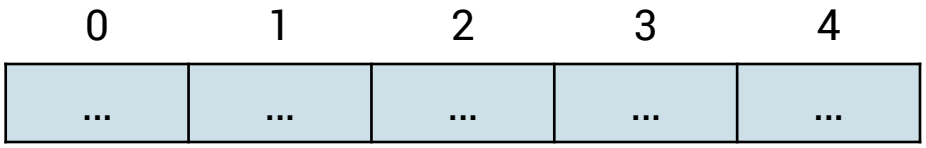
**clear()**

```
v.clear();
```

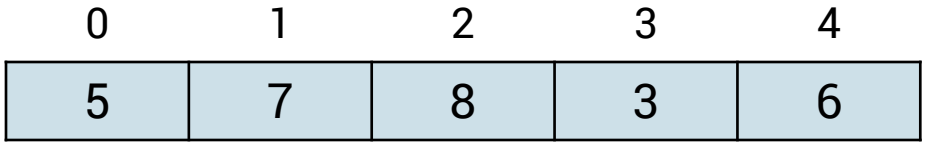
python™

**clear()**

```
l.clear()
```



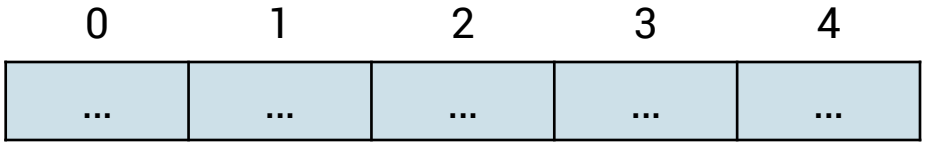
# Xóa toàn bộ ArrayList



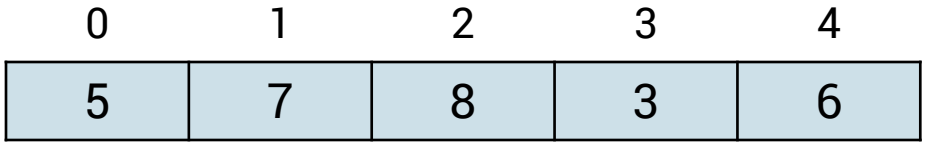


**clear()**: Xóa tất cả phần tử trong ArrayList.

```
a.clear();
```



# Lấy kích thước của Dynamic Array




C++

**size()**

```
int n = v.size();
cout<<n;
```

python™

**len(obj)**

```
n = len(l)
print(n)
```

**5**

# Lấy kích thước của ArrayList

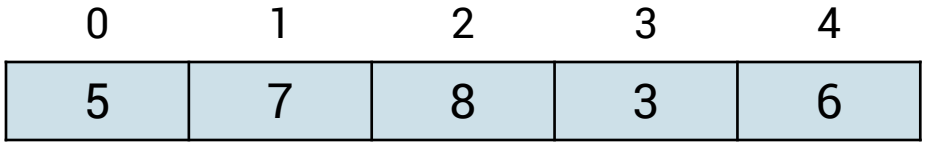
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |


**size()**: Lấy kích thước / số lượng phần tử trong ArrayList.

```
a.size();
```

**5**

# Thay đổi kích thước Dynamic Array lớn thêm




C++

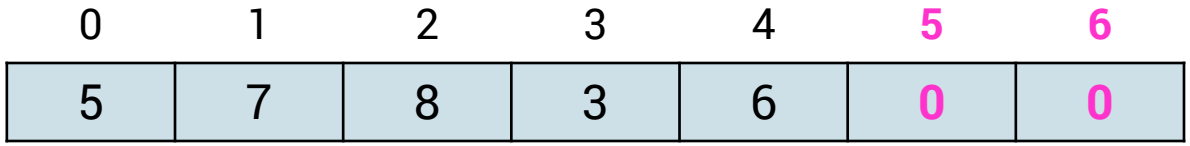
resize(size_type): Thay đổi và gán giá trị bằng giá trị mặc định của kiểu dữ liệu **nếu có thể**.

```
v.resize(7);
```

python™

extend(list): nối 2 list lại với nhau.

```
l.extend(2*[0])
```



# Thay đổi kích thước ArrayList lớn thêm

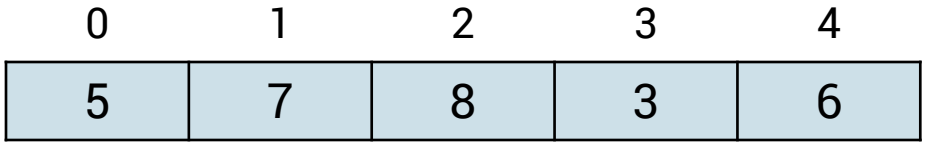
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |


Java không có hàm resize mà phải tự cài đặt bằng tay.

```
for (int i = 5; i < 7; i++) {
    a.add(0);
}
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 | 0 | 0 |

# Thay đổi kích thước Dynamic Array nhỏ lại

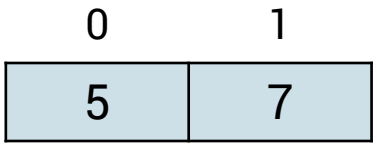



C++

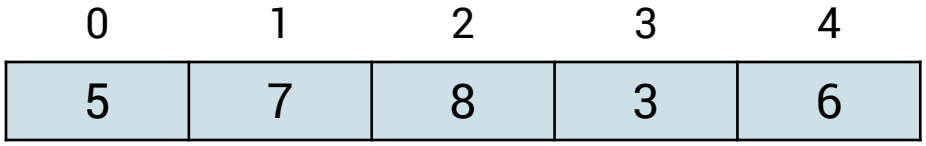
```
v.resize(2);
```

python

```
l = l[0:2]
```



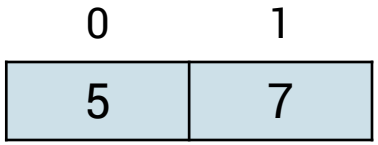
# Thay đổi kích thước ArrayList nhỏ lại



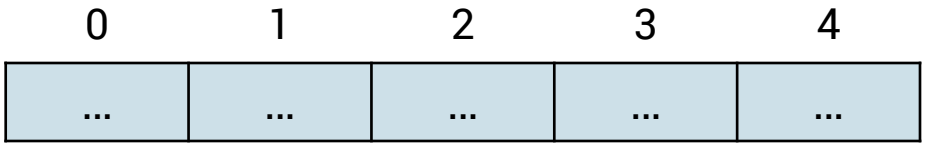


Java không có hàm resize mà phải tự cài đặt bằng tay.

```
a.subList(2, 5).clear();
```



# Kiểm tra xem Dynamic Array có rỗng không




## empty()

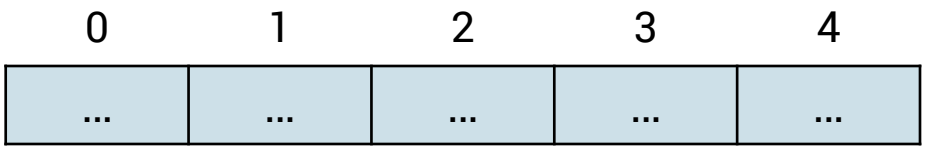
```cpp
vector<int> v;
if(v.empty() == true)
    cout<<"DA is empty!";
else
    cout<<"DA is not empty!";
```

## Sử dụng lại hàm len

```python
l = []
if len(l) == 0:
    print("DA is empty")
else:
    print("DA is not empty")
```

**DA is empty!**

# Kiểm tra ArrayList có rỗng không





**isEmpty()**: Kiểm tra ArrayList có rỗng hay không.

```
ArrayList<Integer> a = new ArrayList<Integer>();
if (a.isEmpty() == true)
    System.out.println("ArrayList is empty!");
else
    System.out.println("ArrayList is not empty!");
```

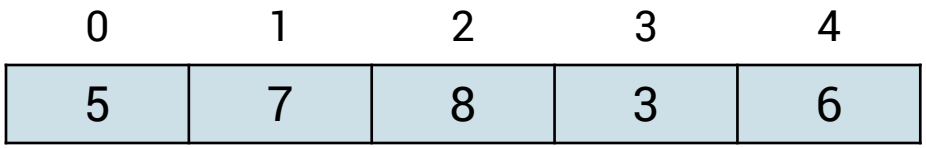
**ArrayList is empty!**

# Một số hàm thành viên khác:


**contains(value):** Trả về true nếu giá trị đó có trong ArrayList, ngược lại trả về false.

**indexOf(value):** Trả về vị trí phần tử nếu có phần tử trong ArrayList bằng với giá trị “value” trong ArrayList, ngược lại trả về -1.

# Duyệt xuôi trong Dynamic Array




C++

```
for(int i=0; i<v.size(); i++)
{
    cout<<v[i]<<", ";
}
```

```
vector<int>::iterator it;
for(it=v.begin(); it!=v.end(); it++)
{
    cout<<*it<<", ";
}
```

python

```
for i in range(0, len(l)):
    print(l[i],end=', ')
```

**5, 7, 8, 3, 6**

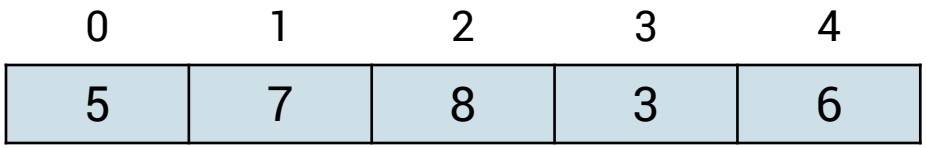
# Duyệt xuôi trong ArrayList

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |


```
for (int i = 0; i < a.size(); i++) {
    System.out.print(a.get(i) + ", ");
}
```

**5, 7, 8, 3, 6**

# Duyệt ngược trong Dynamic Array




C++

```
for(int i=v.size(); i>=0; i--){
    cout<<v[i]<<", ";
}
```

```
ector<int>::reverse_iterator it;
for(it=v.rbegin(); it!=v.rend(); it++) {
    cout<<*it<<", ";
}
```

python

```
for i in range(len(l)-1, -1, -1):
    print(l[i],end=', ')
```

**6, 3, 8, 7, 5**

# Duyệt ngược trong ArrayList

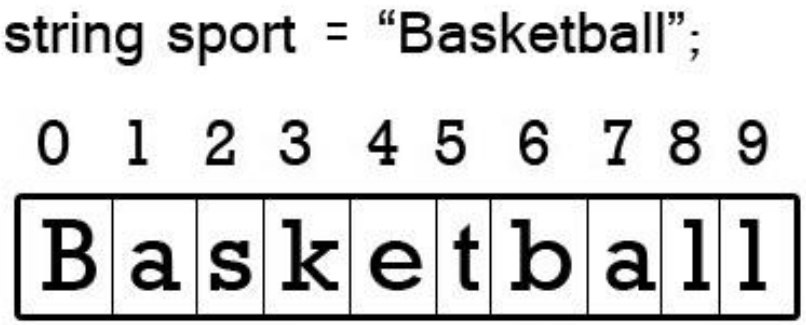
| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 5 | 7 | 8 | 3 | 6 |


```
for (int i = a.size() - 1; i >= 0; i--) {
    System.out.print(a.get(i) + ", ");
}
```

**6, 3, 8, 7, 5**

# String là gì?

**String** là một dãy gồm nhiều ký tự (character) liên tục nhau, các kí tự ở đây rất đa dạng có thể là chữ cái, số, dấu cách, hay các kí hiệu...




**C++:** string

**Python:** variable_name =

**Java:** String

# Cách khai báo và sử dụng

Thư viện:

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

Khai báo:

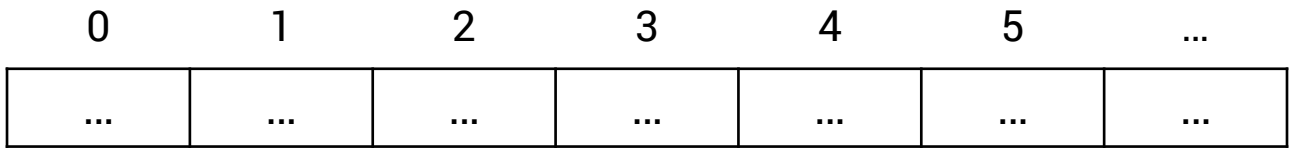
```
string variable_name;
```

```
string s;
```

Khai báo: các biến trong python có thể không cần khai báo trước, tuy nhiên nên khai báo trước một chuỗi rỗng.

```
variable_name = ""
```

```
s = ""
```



# CÁC HÀM KIỂM TRA KÝ TỰ

# Bảng mã ASCII

Big-O Blue

| Dec | Hex | Oct | Chr | Dec | Hex | Oct | HTML | Chr | Dec | Hex | Oct | HTML | Chr | Dec | Hex | Oct | HTML | Chr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 000 | NULL | 32 | 20 | 040 | &#032; | Space | 64 | 40 | 100 | &#064; | @ | 96 | 60 | 140 | &#096; | ` |
| 1 | 1 | 001 | Start of Header | 33 | 21 | 041 | &#033; | ! | 65 | 41 | 101 | &#065; | A | 97 | 61 | 141 | &#097; | a |
| 2 | 2 | 002 | Start of Text | 34 | 22 | 042 | &#034; | " | 66 | 42 | 102 | &#066; | B | 98 | 62 | 142 | &#098; | b |
| 3 | 3 | 003 | End of Text | 35 | 23 | 043 | &#035; | # | 67 | 43 | 103 | &#067; | C | 99 | 63 | 143 | &#099; | c |
| 4 | 4 | 004 | End of Transmission | 36 | 24 | 044 | &#036; | $ | 68 | 44 | 104 | &#068; | D | 100 | 64 | 144 | &#100; | d |
| 5 | 5 | 005 | Enquiry | 37 | 25 | 045 | &#037; | % | 69 | 45 | 105 | &#069; | E | 101 | 65 | 145 | &#101; | e |
| 6 | 6 | 006 | Acknowledgment | 38 | 26 | 046 | &#038; | & | 70 | 46 | 106 | &#070; | F | 102 | 66 | 146 | &#102; | f |
| 7 | 7 | 007 | Bell | 39 | 27 | 047 | &#039; | ' | 71 | 47 | 107 | &#071; | G | 103 | 67 | 147 | &#103; | g |
| 8 | 8 | 010 | Backspace | 40 | 28 | 050 | &#040; | ( | 72 | 48 | 110 | &#072; | H | 104 | 68 | 150 | &#104; | h |
| 9 | 9 | 011 | Horizontal Tab | 41 | 29 | 051 | &#041; | ) | 73 | 49 | 111 | &#073; | I | 105 | 69 | 151 | &#105; | i |
| 10 | A | 012 | Line feed | 42 | 2A | 052 | &#042; | * | 74 | 4A | 112 | &#074; | J | 106 | 6A | 152 | &#106; | j |
| 11 | B | 013 | Vertical Tab | 43 | 2B | 053 | &#043; | + | 75 | 4B | 113 | &#075; | K | 107 | 6B | 153 | &#107; | k |
| 12 | C | 014 | Form feed | 44 | 2C | 054 | &#044; | , | 76 | 4C | 114 | &#076; | L | 108 | 6C | 154 | &#108; | l |
| 13 | D | 015 | Carriage return | 45 | 2D | 055 | &#045; | - | 77 | 4D | 115 | &#077; | M | 109 | 6D | 155 | &#109; | m |
| 14 | E | 016 | Shift Out | 46 | 2E | 056 | &#046; | . | 78 | 4E | 116 | &#078; | N | 110 | 6E | 156 | &#110; | n |
| 15 | F | 017 | Shift In | 47 | 2F | 057 | &#047; | / | 79 | 4F | 117 | &#079; | O | 111 | 6F | 157 | &#111; | o |
| 16 | 10 | 020 | Data Link Escape | 48 | 30 | 060 | &#048; | 0 | 80 | 50 | 120 | &#080; | P | 112 | 70 | 160 | &#112; | p |
| 17 | 11 | 021 | Device Control 1 | 49 | 31 | 061 | &#049; | 1 | 81 | 51 | 121 | &#081; | Q | 113 | 71 | 161 | &#113; | q |
| 18 | 12 | 022 | Device Control 2 | 50 | 32 | 062 | &#050; | 2 | 82 | 52 | 122 | &#082; | R | 114 | 72 | 162 | &#114; | r |
| 19 | 13 | 023 | Device Control 3 | 51 | 33 | 063 | &#051; | 3 | 83 | 53 | 123 | &#083; | S | 115 | 73 | 163 | &#115; | s |
| 20 | 14 | 024 | Device Control 4 | 52 | 34 | 064 | &#052; | 4 | 84 | 54 | 124 | &#084; | T | 116 | 74 | 164 | &#116; | t |
| 21 | 15 | 025 | Negative Ack. | 53 | 35 | 065 | &#053; | 5 | 85 | 55 | 125 | &#085; | U | 117 | 75 | 165 | &#117; | u |
| 22 | 16 | 026 | Synchronous idle | 54 | 36 | 066 | &#054; | 6 | 86 | 56 | 126 | &#086; | V | 118 | 76 | 166 | &#118; | v |
| 23 | 17 | 027 | End of Trans. Block | 55 | 37 | 067 | &#055; | 7 | 87 | 57 | 127 | &#087; | W | 119 | 77 | 167 | &#119; | w |
| 24 | 18 | 030 | Cancel | 56 | 38 | 070 | &#056; | 8 | 88 | 58 | 130 | &#088; | X | 120 | 78 | 170 | &#120; | x |
| 25 | 19 | 031 | End of Medium | 57 | 39 | 071 | &#057; | 9 | 89 | 59 | 131 | &#089; | Y | 121 | 79 | 171 | &#121; | y |
| 26 | 1A | 032 | Substitute | 58 | 3A | 072 | &#058; | : | 90 | 5A | 132 | &#090; | Z | 122 | 7A | 172 | &#122; | z |
| 27 | 1B | 033 | Escape | 59 | 3B | 073 | &#059; | ; | 91 | 5B | 133 | &#091; | [ | 123 | 7B | 173 | &#123; | { |
| 28 | 1C | 034 | File Separator | 60 | 3C | 074 | &#060; | < | 92 | 5C | 134 | &#092; | \ | 124 | 7C | 174 | &#124; | \| |
| 29 | 1D | 035 | Group Separator | 61 | 3D | 075 | &#061; | = | 93 | 5D | 135 | &#093; | ] | 125 | 7D | 175 | &#125; | } |
| 30 | 1E | 036 | Record Separator | 62 | 3E | 076 | &#062; | > | 94 | 5E | 136 | &#094; | ^ | 126 | 7E | 176 | &#126; | ~ |
| 31 | 1F | 037 | Unit Separator | 63 | 3F | 077 | &#063; | ? | 95 | 5F | 137 | &#095; | _ | 127 | 7F | 177 | &#127; | Del |

asciichars.com

# Kiểm tra ký tự



Dùng mã ASCII kiểm tra

```cpp
#include <iostream>
using namespace std;
int main() {
    string s("123abc");
    if (s[0] > 48 && s[0] < 57)
        cout << "number";
    return 0;
}
```

Trong Python sử dụng hàm ord() để lấy mã của chuỗi có 1 kí tự.

Để chuyển mã ASCII thành chuỗi, sử dụng hàm chr().

```python
s = "123abc"
if ord(s[0]) > 48 and
ord(s[0]) < 57:
    print("number")
```

**number**

# Kiểm tra ký tự


Dùng mã ASCII kiểm tra

```
String s = "123abc";
if (s.charAt(0) >= 48 && s.charAt(0) < 58)
    System.out.println(“number");
```

**number**

# Một số hàm kiểm tra ký tự

- isalpha
- isdigit
- islower
- isupper

***Lưu ý:*** *python không chỉ kiểm tra từng ký tự mà có thể kiểm tra toàn bộ chuỗi.*

*Một số hàm kiểm tra ký tự, C++ sử dụng thư viện <ctype.h>*

# Kiểm tra ký tự có phải chữ cái không

## isalpha

Cú pháp: int result = **isalpha**(char c)

**Kết quả trả về:**

- 0: ký tự đó không phải chữ cái.
- Khác 0: ký tự đó là chữ cái.

```
string s = "Ky Thuat Lap Trinh";
int result = isalpha(s[1]);
cout<<result;
```

**2**

## isalpha

Cú pháp: result = s.**isalpha**()

**Kết quả trả về:**

- False: chuỗi rỗng hoặc ký tự đó không phải chữ cái.
- True: ký tự đó là ký tự chữ cái.

```
s = "Ky Thuat Lap Trinh"
result = s[1].isalpha()
print(result)
```

**True**

# Kiểm tra ký tự có phải chữ cái không


## isLetter

**Cú pháp:** isLetter(character c). Thư viện: java.lang.Character

**Kết quả trả về:**

- false: ký tự đó không phải chữ cái.
- true: ký tự đó là chữ cái.

```
System.out.println(Character.isLetter('c'));
System.out.println(Character.isLetter('5'));
```

**true**
**false**

# Kiểm tra ký tự có phải số không



## isdigit

Cú pháp: int result = **isdigit**(char c)

**Kết quả trả về:**

- 0: ký tự đó không phải số.
- Khác 0: ký tự đó là số.

```
string s = "Thu 6";

int result = isdigit(s[4]);

cout << result;
```

**4**

## isdigit

Cú pháp: result = s.**isdigit**()

**Kết quả trả về:**

- False: chuỗi rỗng hoặc ký tự đó không phải là số.
- True: ký tự đó là số.

```
s = "Thu 6"

result = s[4].isdigit()

print(result)
```

**True**

# Kiểm tra ký tự có phải số không


## isDigit

**Cú pháp:** isDigit(character c)

**Kết quả trả về:**

- false: ký tự đó không phải số.
- true: ký tự đó là số.

```
System.out.println(Character.isDigit('c'));
System.out.println(Character.isDigit('5'));
```

```
false
true
```

# Kiểm tra ký tự có phải viết thường không


## islower

Cú pháp: int result = **islower**(char c)

**Kết quả trả về:**

- 0: ký tự đó không phải viết thường.
- Khác 0: ký tự đó viết thường.

```
string s = "Thu 6";
int result = islower(s[1]);
cout << result;
```

**2**


## islower

Cú pháp: result = s.**islower**()

**Kết quả trả về:**

- False: chuỗi rỗng hoặc ký tự đó không phải viết thường.
- True: ký tự đó viết thường.

```
s = "Thu 6"
result = s[1].islower()
print(result)
```

**True**

# Kiểm tra ký tự có phải viết thường không


**isLowerCase**

**Cú pháp:** isLowerCase(character c)

**Kết quả trả về:**

- false: Ký tự đó không phải viết thường.
- true: Ký tự đó viết thường.

```
System.out.println(Character.isLowerCase('c'));
System.out.println(Character.isLowerCase('C'));
```

**true**
**false**

# Kiểm tra ký tự có phải viết hoa không

## isupper

Cú pháp: int result = **isupper**(char c)

**Kết quả trả về:**

- 0: ký tự đó không phải viết hoa.
- Khác 0: ký tự đó viết hoa.

```
string s = "Thu 6";
int result = isupper(s[0]);
cout << result;
```

**1**

## isupper

Cú pháp: result = s. **isupper**()

**Kết quả trả về:**

- False: chuỗi rỗng hoặc ký tự đó không phải viết hoa.
- True: ký tự đó viết hoa.

```
s = "Thu 6"
result = s[0].isupper()
print(result)
```

**True**

# Kiểm tra ký tự có phải viết hoa không


**isUpperCase**

Cú pháp: isUpperCase(character c)

**Kết quả trả về:**

- false: Ký tự đó không phải viết hoa.
- true: Ký tự đó viết hoa.

```
System.out.println(Character.isUpperCase('c'));
System.out.println(Character.isUpperCase('C'));
```

**false**
**true**

# CÁC HÀM CHUẨN HÓA TRONG STRING

# Chuyển chuỗi thành kiểu số



atoi(char *str): Chuyển chuỗi thành số nguyên.

atof(char *str): Chuyển chuỗi thành số thực.

```
string s("12");
int number = atoi(s.c_str());
cout << number;
```

Chuyển chuỗi thành số nguyên:

int(string)

Chuyển chuỗi thành số thực:

float(string)

```
s = "12"
number = int(s)
print(number)
```

**12**

# Chuyển chuỗi thành kiểu số


Integer.parseInt (string, base = 10)

```
String s = "12";
int number = Integer.parseInt(s);
// int number = Integer.parseInt(s, 10);
System.out.println(number);
```

**12**

# Chuyển số thành chuỗi



to_string(value)

```cpp
string s;
int number = 15789;
s = to_string(number);
cout << s;
```

str(value)

```python
number = 15789
s = str(number)
print(s)
```

**15789**

# Chuyển số thành chuỗi


Integer.toString(value)

```
int number = 15789;
String s = Integer.toString(number);
System.out.println(s);
```

**15789**

# In hoa và in thường ký tự - dùng hàm

toupper(char c): Chuyển ký tự thành ký tự in hoa.

tolower(char c): Chuyển ký tự thành ký tự in thường.

```
string s(“algorithm”);

char c = toupper(s[2]);

cout << c;
```

upper(): Chuyển ký tự thành ký tự in hoa.

lower(): Chuyển ký tự thành ký tự in thường.

Lưu ý: trong Python 2 hàm này có thể chuyển nguyên chuỗi thành hoa/thường.

```
s = "algorithm"

c = s[2].upper()

print(c)
```

**G**

# In hoa và in thường ký tự - dùng hàm


toUpperCase(): Chuyển ký tự thành ký tự in hoa.

toLowerCase(): Chuyển ký tự thành ký tự in thường.

```
String s = "algorithm";
char c = Character.toUpperCase(s.charAt(2));
System.out.print(c);
```

**G**

# In hoa và in thường ký tự - ASCII



In hoa và In thường ký tự (**dùng mã ASCII**)

- ký tự - 32: Chuyển ký tự thành ký tự in hoa.
- Ký tự + 32: Chuyển ký tự thành ký tự in thường.

```
string s("algorithm");
s[2] = s[2] - 32;
cout << s[2];
```

**G**

***Lưu ý:*** *In hoa và In thường ký tự (**dùng mã ASCII**). String trong Python là immutable, không thể cập nhật trực tiếp vào thành phần của string để gán.*

# In hoa và in thường ký tự - ASCII


In hoa và In thường ký tự (**dùng mã ASCII**)

- ký tự - 32: Chuyển ký tự thành ký tự in hoa.
- Ký tự + 32: Chuyển ký tự thành ký tự in thường.

```
String s = "algorithm";
char c = s.charAt(2);
c -= 32;
System.out.print(c);
```

**G**

# MỘT SỐ LƯU Ý KHI SỬ DỤNG STRING

# Đọc từng từ và nguyên dòng

nothing is impossible


```
string s0;
cin >> s0;
cout << s0;
```

nothing

```
string s1;
getline(cin, s1);
cout << s1;
```

nothing is impossible

# Đọc từng từ và nguyên dòng

**nothing is impossible**


Trong Python mặc định sẽ đọc theo từng dòng nên không có sự phần biệt giữa đọc từng từ và đọc nguyên dòng. Để phần tách lấy từng từ thì ta có thể dùng hàm split sau khi đọc.

```
line = input().split()
s0 = line[0]
print(s0)
```

**nothing**

# Đọc từng từ và nguyên dòng

nothing is impossible


```
Scanner sc = new Scanner(System.in);
String s0 = sc.next();
System.out.print(s0);
```

nothing

```
Scanner sc = new Scanner(System.in);
String s1 = sc.nextLine();
System.out.print(s1);
```

nothing is impossible

# Hỏi đáp