

LECTURE 07

STRING



python™

Big-O Coding

Website: www.bigocoding.com

Kiểu kí tự (C++)

- Kí tự: A-Z, a-z, 0-9, các dấu, các kí tự điều khiển.
- C++ có kiểu dữ liệu **char**, dùng để lưu 1 kí tự.
- Kí tự được đặt trong cặp dấu “”

```
char c;  
  
cin >> c;  
cout << c << endl;  
  
c = '0';  
cout << c << endl;
```

Kiểu kí tự (Java)

- Tương tự, Java cũng có kiểu dữ liệu **char**, biểu diễn 1 kí tự.

```
char c;

Scanner sc = new Scanner(System.in);
c = sc.nextLine().charAt(0);
System.out.println(c);

c = '!';
System.out.println(c);
```

Kiểu kí tự (Python)

- Python **không có** kiểu char.
- Python coi 1 kí tự cũng là 1 chuỗi.

Kiểu chuỗi

- Chuỗi, bao gồm **nhiều kí tự**, dùng để lưu thông tin dạng text.
- Ví dụ:
 - Chuỗi lưu thông báo “Hello, World!!!”.
 - Chuỗi lưu họ tên “Ho Tuan Thanh”.
 - Chuỗi lưu địa chỉ “75 Huynh Tinh Cua”.
 - Chuỗi lưu tên khóa học “Big-O Green 7”.

1. Tạo chuỗi (C++)

- Trong C/C++, chuỗi được tạo thành bằng cách khai báo **mảng 1 chiều các kí tự** (char).
- Kí tự '**\0**' là kí tự kết thúc chuỗi.
- Dùng dấu "" cho chuỗi.

```
#include <string.h>

char name[100] = {'T', 'h', 'a', 'n', 'h'};
cout << name << endl;

name[4] = '\0';
cout << name << endl;
```

Thanh
Than

1. Tạo chuỗi (Java)

- Java cũng coi chuỗi là một mảng các kí tự.
- Java có kiểu dữ liệu **String** dùng cho chuỗi.

```
String name = "Thanh";  
System.out.println(name);
```

1. Tạo chuỗi (Python)

- Python cũng coi chuỗi là một mảng các kí tự.
- Python cũng có kiểu **str** để biểu diễn chuỗi.

```
name = "Thanh"  
print(name)
```

```
Thanh
```

2. Đọc chuỗi nhập vào (C++)

- Toán tử **>>**: đọc vào chuỗi không có khoảng trắng.

```
char name[100];
cin >> name;
cout << name << endl;
```

```
// Nhập vào: Ho Tuan Thanh
Ho
```

- Hàm **cin.getline()**: đọc vào chuỗi có/không có khoảng trắng.

```
char name[101];
cin.getline(name, 101);
cout << name << endl;
```

name: tối đa 100 kí tự

```
// Nhập vào: Ho Tuan Thanh
Ho Tuan Thanh
```

2. Đọc chuỗi nhập vào (Java)

- Hàm **Scanner.next()**: đọc chuỗi không có khoảng trắng.

```
String name = "";
Scanner sc = new Scanner(System.in);
name = sc.next();
System.out.println(name);
```

```
// Nhập vào: Ho Tuan Thanh
Ho
```

- Hàm **Scanner.nextLine()**: đọc chuỗi có / không có khoảng trắng.

```
String name = "";
Scanner sc = new Scanner(System.in);
name = sc.nextLine();
System.out.println(name);
```

```
// Nhập vào: Ho Tuan Thanh
Ho Tuan Thanh
```

2. Đọc chuỗi nhập vào (Python)

- Sử dụng hàm **input()**, như khi nhập giá trị cho các kiểu dữ liệu khác.

```
name = input()  
print(name)
```

```
// Nhập vào: Ho Tuan Thanh  
Ho Tuan Thanh
```

3. Lấy độ dài của chuỗi

- C++: không cần biến n đi kèm, sử dụng hàm **strlen(s)**.
- Java: sử dụng hàm **s.length()** (không phải thuộc tính s.length như mảng).
- Python: sử dụng hàm **len(s)** (giống như mảng).

```
char name[100] = {"Thanh"};
cout << strlen(name) << endl;
```

```
name[4] = '\0';
cout << strlen(name) << endl;
```

```
String name = "Thanh";
System.out.println(name.length());
```

```
name = "Thanh"
print(len(name))
```

4. Gán bằng 2 chuỗi

- C++: không hỗ trợ toán tử gán bằng.
 - Sử dụng hàm **strcpy(dest, src)**.
- Java, Python: hỗ trợ **toán tử gán bằng**.

```
char name[100] = {"Thanh"};  
char name2[100];  
name2 = name;
```

```
char name[100] = {"Thanh"};  
char name2[100];  
strcpy(name2, name);
```

```
String name = "Thanh";  
String name2 = name;
```

```
name = "Thanh"  
name2 = name
```

5. Truy xuất các kí tự trong chuỗi (C++)

- Chuỗi là một mảng các kí tự, nên ta sử dụng **[index]** để truy xuất từng kí tự trong chuỗi.

0	1	2	3	4	5	6	7	8	9	10	11	12
H	o		T	u	a	n		T	h	a	n	h

```
char s[100] = {"Ho Tuan Thanh"};
cout << s[0] << endl;
cout << s[8] << endl;
cout << s[12] << endl;

cout << s[-1];
cout << s[14];
```

H
T
h

// rác
// rác

5. Truy xuất các kí tự trong chuỗi (Java)

- Sử dụng hàm s.**charAt(index)**.

0	1	2	3	4	5	6	7	8	9	10	11	12
H	o		T	u	a	n		T	h	a	n	h

```
String s = "Ho Tuan Thanh";
System.out.println(s.charAt(0));
System.out.println(s.charAt(8));
System.out.println(s.charAt(12));
```

```
System.out.println(s.charAt(-1));
System.out.println(s.charAt(14));
```

H
T
h

StringIndexOutOfBoundsException
StringIndexOutOfBoundsException

5. Truy xuất các kí tự trong chuỗi (Python)

- Tương tự C++, chuỗi là một mảng các kí tự, nên ta sử dụng **[index]** để truy xuất từng kí tự trong chuỗi.

0	1	2	3	4	5	6	7	8	9	10	11	12
H	o		T	u	a	n		T	h	a	n	h
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
s = "Ho Tuan Thanh"
print(s[0])
print(s[8])
print(s[12])

print(s[-1])
print(s[14])
```

```
H
T
h
h
IndexError: string index out of range
```

6. Thay đổi nội dung chuỗi (C++)

- C++ cho phép thay đổi nội dung chuỗi. (**mutable**)

```
char s[100] = {"Ho Tuan Thanh"};
s[5] = '@';
cout << s << endl;
```

```
Ho Tu@n Thanh
```

6. Thay đổi nội dung chuỗi (Java)

- Java không cho phép thay đổi nội dung chuỗi.
(immutable)

```
String s = "Ho Tuan Thanh";
s[5] = "@"; // compiled-time error
s.charAt(5) = "@"; // compiled-time error
```

6. Thay đổi nội dung chuỗi (Python)

- Giống như Java, Python cũng không được phép thay đổi nội dung chuỗi. (**immutable**)

```
s = "Ho Tuan Thanh"  
s[5] = "@"  
print(s)
```

```
TypeError: 'str' object does not support item assignment
```

7. Duyệt theo index

```
char s[100] = {"Ho Tuan Thanh"};
for(int i = 0; i < strlen(s); i++){
    cout << s[i] << " ";
}
```

```
String s = "Ho Tuan Thanh";
for(int i = 0; i < s.length(); i++){
    System.out.printf("%c ", s.charAt(i));
}
```

```
s = "Ho Tuan Thanh"
for i in range(len(s)):
    print(s[i], end=" ")
```

8. Duyệt theo kí tự

```
// C++ không hỗ trợ cách duyệt này
```

```
String s = "Ho Tuan Thanh";
for(char ch: s.toCharArray()){
    System.out.printf("%c ", ch);
}
```

s.toCharArray(): chuyển từ String sang char[]

```
s = "Ho Tuan Thanh"
for ch in s:
    print(ch, end=" ")
```

ASCII table

- Mỗi kí tự trong chuỗi được đại diện bằng 1 số nguyên trong bảng mã ASCII/Unicode.
- C++, Java:
 - **(int)ch**: trả về số nguyên đại diện kí tự ch.
 - **(char)x**: trả về kí tự tương ứng với số nguyên x.
- Python:
 - **ord(ch)**: trả về số nguyên đại diện kí tự ch.
 - **chr(x)**: trả về kí tự tương ứng với số nguyên x.

ASCII table

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

Ví dụ

```
cout << (int)'A' << endl;  
cout << (int)'a' << endl;  
cout << (int)'B' << endl;  
cout << (int)'5' << endl;
```

```
cout << (char)65 << endl;  
cout << (char)66 << endl;  
cout << (char)49 << endl;  
cout << (char)53 << endl;
```

```
print(ord("A"))  
print(ord("a"))  
print(ord("B"))  
print(ord("5"))
```

```
print(chr(65))  
print(chr(66))  
print(chr(49))  
print(chr(53))
```

65
97
66
53

A
B
1
5

```
System.out.println((int)'A');  
System.out.println((int)'a');  
System.out.println((int)'B');  
System.out.println((int)'5');
```

```
System.out.println((char)65);  
System.out.println((char)66);  
System.out.println((char)49);  
System.out.println((char)53);
```

9. So sánh chuỗi (C++)

- Không thể sử dụng các toán tử <, <=, ==, !=, >=, > để so sánh chuỗi.
- Sử dụng hàm **strcmp(s1, s2)**, trả về 3 giá trị:
 - <0: s1 < s2
 - ==0: s1 == s2
 - >0: s1 > s2
- So sánh có phân biệt chữ hoa, chữ thường.

```
char s1[100] = {"Big-O Coding"};
char s2[100] = {"Google"};
cout << strcmp(s1, s2) << endl;

char s3[100] = {"Thanh"};
char s4[100] = {"thanh"};
cout << strcmp(s3, s4) << endl;
```

9. So sánh chuỗi (Java)

- Sử dụng hàm `s1.compareTo(s2)` để so sánh.
- So sánh có phân biệt chữ hoa, chữ thường

```
String s1 = "Big-O Coding";
String s2 = "Google";
System.out.println(s1.compareTo(s2));
```

```
String s3 = "Thanh";
String s4 = "thanh";
System.out.println(s3.compareTo(s4));
```

9. So sánh chuỗi (Python)

- Sử dụng các toán tử so sánh `<, <=, ==, !=, >=, >` để so sánh chuỗi (so sánh theo thứ tự từ điển).
- So sánh có phân biệt chữ hoa, chữ thường.

```
s1 = "Big-O Coding"  
s2 = "Google"  
print(s1 < s2)
```

```
s3 = "Thanh"  
s4 = "thanh"  
print(s3 == s4)
```

10. So sánh không phân biệt hoa thường (C++)

- Sử dụng hàm **tolower(ch)** để chuyển các kí tự thành chữ thường.
- Sau đó, sử dụng hàm **strcmp(s1, s2)**.
- Hàm **strcmp(s1, s2)** có thể so sánh không phân biệt hoa thường.
 - Nhưng một vài compiler không hỗ trợ.

```
char s3[100] = {"Thanh"};
char s4[100] = {"thanh"};

tolower(s3);
tolower(s4);
cout << strcmp(s3, s4) << endl;
```

```
void tolower(char s[]){
    for(int i = 0; i < strlen(s); i++){
        s[i] = tolower(s[i]);
    }
}
```

10. So sánh không phân biệt hoa thường (Java)

- Sử dụng hàm `s1.compareToIgnoreCase(s2)`.

```
String s3 = "Thanh";
String s4 = "thanh";
System.out.println(s3.compareToIgnoreCase(s4));
```

10. So sánh không phân biệt hoa thường (Python)

- Sử dụng hàm `s.lower()` để chuyển các kí tự thành chữ thường.
- Sau đó, sử dụng các toán tử để so sánh.

```
s3 = "Thanh"  
s4 = "thanh"  
  
s3 = s3.lower()  
s4 = s4.lower()  
print(s3 == s4)
```

11. Nối chuỗi (C++)

- C++ không hỗ trợ toán tử cộng chuỗi.
- Sử dụng hàm **strcat**(dest, src).

```
char first[100] = {"Thanh"};
char last[100] = {"Ho"};
char name[100];
strcpy(name, first);
strcat(name, " ");
strcat(name, last);
```

11. Nối chuỗi (Java)

- Java hỗ trợ **toán tử cộng** chuỗi.

```
String first = "Thanh";
String last = "Ho";
String name = first + " " + last;
```

11. Nối chuỗi (Python)

- Python hỗ trợ **toán tử cộng** chuỗi.

```
first = "Thanh"  
last = "Ho"  
  
name = first + " " + last
```

12. Kiểm tra s2 có xuất hiện trong s1

- Sử dụng hàm **strstr**(s1, s2).
 - Nếu trả về NULL, s2 không xuất hiện trong s1.
 - Nếu trả về khác NULL, s2 có xuất hiện trong s1.

```
char s1[100] = {"Ho Tuan Thanh"};
char s2[100] = {"thanh"};
tolower(s1);
tolower(s2);

if(strstr(s1, s2)!= NULL){
    cout << "s2 in s1" << endl;
}
else{
    cout << "s2 not in s1" << endl;
}
```

12. Kiểm tra s2 có xuất hiện trong s1 (Java)

- Sử dụng hàm s1.**indexOf**(s2).
 - Nếu trả về -1, s2 không xuất hiện trong s1.
 - Ngược lại, trả về vị trí xuất hiện của s2 trong s1.

```
String s1 = "Ho Tuan Thanh";
String s2 = "Thanh";

s1 = s1.toLowerCase();
s2 = s2.toLowerCase();

if(s1.indexOf(s2) >= 0)
    System.out.println("s2 in s1");
else
    System.out.println("s2 not in s1");
```

12. Kiểm tra s2 có xuất hiện trong s1 (Python)

- Sử dụng hàm s1.**find**(s2).
 - Nếu trả về -1, s2 không xuất hiện trong s1.
 - Ngược lại, trả về vị trí xuất hiện của s2 trong s1.

```
s1 = "Ho Tuan Thanh"  
s2 = "Thanh"  
  
s1 = s1.lower()  
s2 = s2.lower()  
if(s1.find(s2) >= 0):  
    print("s2 in s1")  
else:  
    print("s2 not in s1")
```

13. Tạo chuỗi mới từ chuỗi ban đầu (C++)

- Tạo chuỗi chỉ chứa các chữ số từ chuỗi ban đầu.
 - VD: s = “Green04” → ans = “04”
1. Khai báo chuỗi kết quả ans.
 2. Khởi tạo j = 0.
 3. Duyệt qua từng kí tự trong chuỗi s.
 - a. Nếu s[i] là chữ số, **đưa s[i] vào vị trí ans[j]**: ans[j] = s[i].
 - b. Tăng biến j lên 1.
 4. Nhớ gán kí tự kết thúc chuỗi: **ans[j] = '\0'**.

13. Tạo chuỗi mới từ chuỗi ban đầu (C++)

```
char s[100] = {"Green04"};
char ans[100];
int j = 0;
for(int i = 0; i < strlen(s); i++){
    if(s[i] >= '0' && s[i] <= '9'){
        ans[j] = s[i];
        j++;
    }
}
ans[j] = '\0';
cout << ans;
```

13. Tạo chuỗi mới từ chuỗi ban đầu (Java)

- Tạo chuỗi chỉ chứa các chữ số từ chuỗi ban đầu.
 - VD: s = “Green04” → ans = “04”
- Trên Java, việc tạo chuỗi mới dễ hơn.
 1. Khởi tạo chuỗi kết quả ans rỗng.
 2. Duyệt qua từng kí tự trong chuỗi s.
 - Nếu s[i] là chữ số, **cộng dồn s[i] vào ans**.

13. Tạo chuỗi mới từ chuỗi ban đầu (Java)

```
String s = "Green04";
String ans = "";
for(int i = 0; i < s.length(); i++){
    if(s.charAt(i) >= '0' && s.charAt(i) <= '9'){
        ans += s.charAt(i);
    }
}
System.out.println(ans);
```

13. Tạo chuỗi mới từ chuỗi ban đầu (Python)

- Tạo chuỗi chỉ chứa các chữ số từ chuỗi ban đầu.
 - VD: s = “Green04” → ans = “04”
- Trên Python, việc tạo chuỗi mới cũng rất dễ dàng, tương tự trên Java.
 1. Khởi tạo chuỗi kết quả ans rỗng.
 2. Duyệt qua từng kí tự trong chuỗi s.
 - Nếu s[i] là chữ số, **cộng dồn s[i] vào ans**.

13. Tạo chuỗi mới từ chuỗi ban đầu (Python)

```
s = "Green04"  
ans = ""  
for i in range(len(s)):  
    if(ord(s[i]) >= ord('0') and ord(s[i]) <= ord('9')):  
        ans += s[i]  
print(ans)
```

14. Tách chuỗi dựa vào kí tự phân cách (C++)

```
int main() {
    char s[] = "nstnguyen;dtphuyen;nvaminh;nqchuong";
    const char delimiter[2] = ";";
    char *token;

    token = strtok(s, delimiter);

    while( token != NULL ) {
        cout << token << endl;
        token = strtok(NULL, delimiter);
    }
    return 0;
}
```

14. Tách chuỗi dựa vào kí tự phân cách (Java)

```
String s = "nstnguyen;dtphuyen;nvaminh;nqchuong";
String []a = s.split(";");
for(String ans: a){
    System.out.println(ans);
}
```

14. Tách chuỗi dựa vào kí tự phân cách (Python)

```
s = "nstnguyen;dtphuyen;nvaminh;nqchuong"  
a = s.split(";")  
print(a)
```

Bài tập

BT1 – KIỂM TRA KÍ TỰ XUẤT HIỆN

- Viết chương trình đọc vào 1 chuỗi, kiểm tra chuỗi đó có xuất hiện một trong các ký tự 'B' 'I' 'G' hoặc 'O' hay không? (Không phân biệt hoa thường) Nếu có in ra YES, ngược lại in ra NO.

Kỹ thuật đặt cờ hiệu

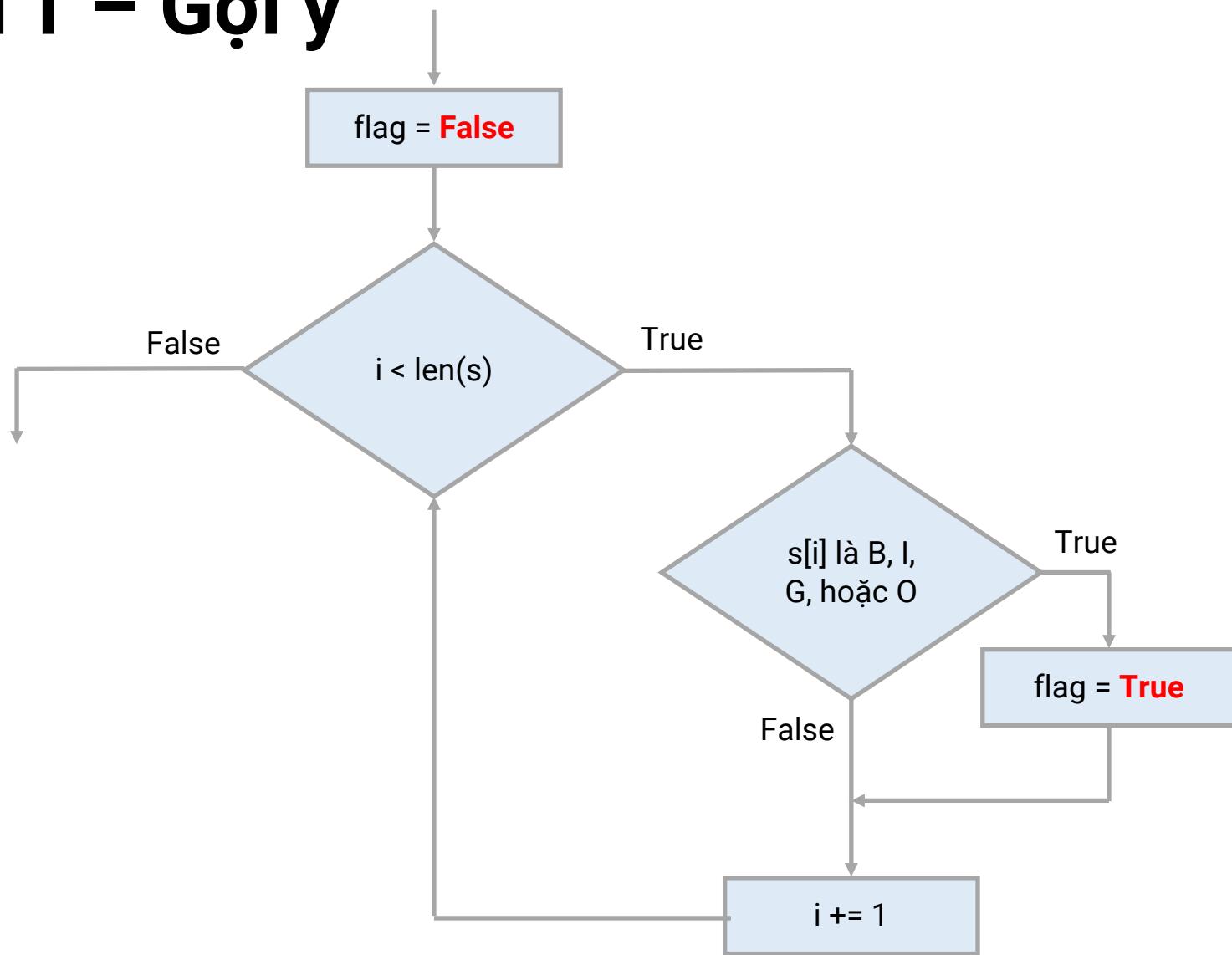


BT1 – Gợi ý

- a. Đọc vào chuỗi s.
- b. Khởi tạo cờ hiệu, flag = False.
- c. Duyệt qua từng kí tự trong chuỗi s.
 - Nếu kí tự s[i] là B, I, G, O, b, i, g hoặc o thì gán lại flag = True.
- d. In kết quả.



BT1 – Gợi ý



BT1 – Minh họa – s = “Google”

flag = False

i = 0

n = 6 (strlen/len = 6)

0 < 6: True

s[0] = 'G' == B, I, G, O, b, i, g hoặc o: True, flag = True

1 < 6: True

s[1] = 'o' == B, I, G, O, b, i, g hoặc o: True, flag = True

2 < 6: True

s[2] = 'o' == B, I, G, O, b, i, g hoặc o: True, flag = True

3 < 6: True

s[3] = 'g' == B, I, G, O, b, i, g hoặc o: True, flag = True

4 < 6: True

s[4] = 'l' == B, I, G, O, b, i, g hoặc o: False

5 < 6: True

s[5] = 'e' == B, I, G, O, b, i, g hoặc o: False

6 < 6: False

BT2 – KIỂM TRA KÝ TỰ SỐ

- Viết chương trình đếm xem chuỗi đã cho có bao nhiêu ký tự là số.

Kĩ thuật đếm

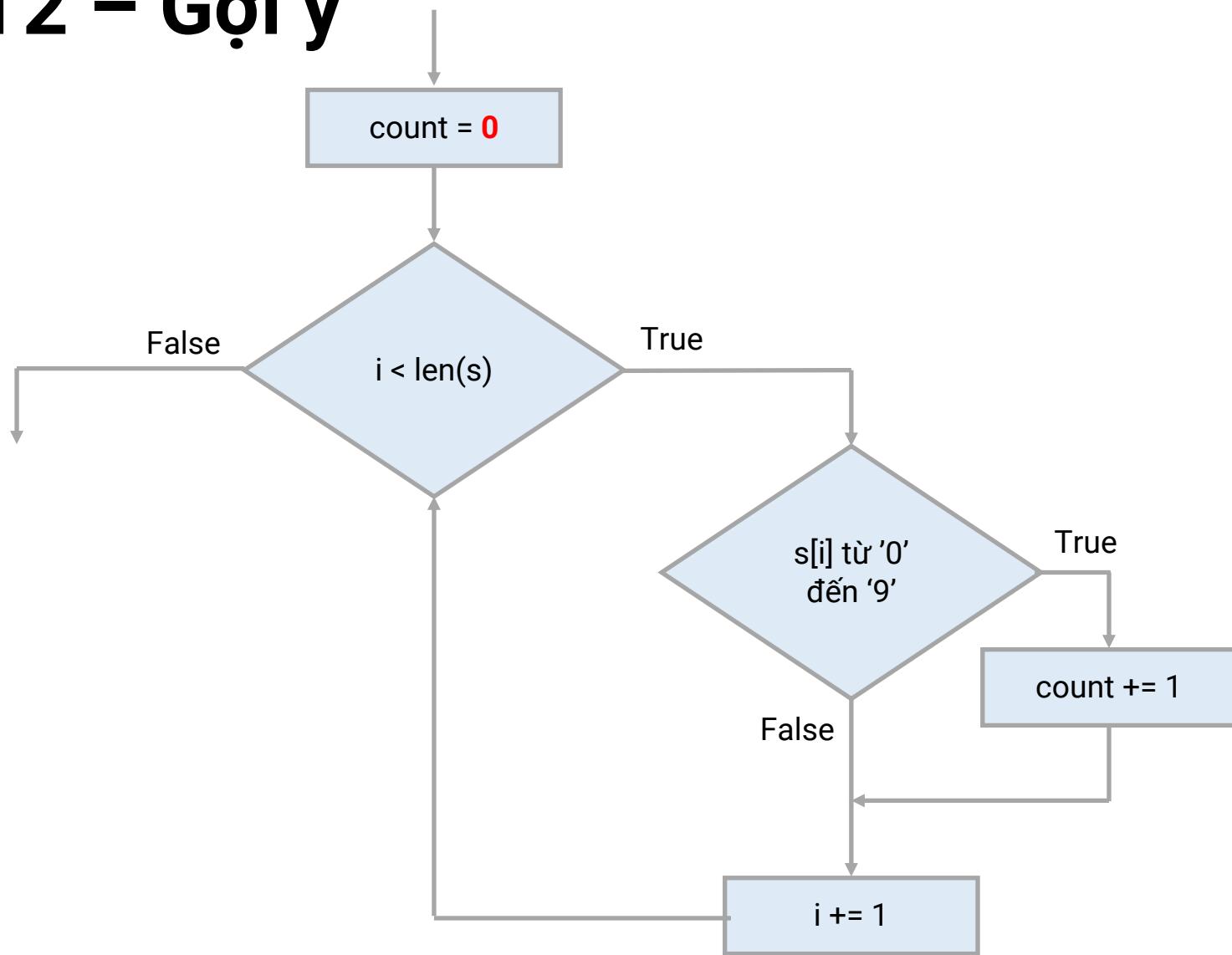


BT2 – Gợi ý

1. Khởi tạo đếm, **count = 0**.
2. Duyệt qua từng kí tự trong chuỗi s.
 - a. Nếu kí tự $s[i]$ là số, ' 0 ' \leq $s[i]$ \leq ' 9 ', **tăng count lên 1**.



BT2 – Gợi ý



BT2 – Minh họa

s = "Green04"

count = 0

i = 0

n = 7

0 < 7: True

s[0] = 'G'

1 < 7: True

s[1] = 'r'

2 < 7: True

s[2] = 'e'

3 < 7: True

s[3] = 'e'

4 < 7: True

s[4] = 'n'

5 < 7: True

s[5] = '0', count = 0 + 1 = 1

6 < 7: True

s[6] = '4', count = 1 + 1 = 2

7 < 7: False

BT3 – CHUẨN HÓA CHUỖI

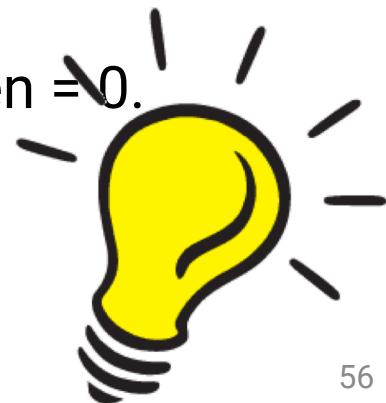
- Xóa các khoảng trắng ở đầu, cuối chuỗi. Giữa 2 từ chỉ giữ lại 1 khoảng trắng.

Kĩ thuật tạo chuỗi mới từ chuỗi ban đầu



BT3 – Gợi ý

- Duyệt qua từng kí tự trong chuỗi s
 - Nếu $s[i]$ là khoảng trắng, đánh dấu need-a-space.
 - Nếu $s[i]$ không phải là khoảng trắng,
 - Nếu need-a-space, đưa kí tự khoảng trắng và kí tự $s[i]$ vào chuỗi ans.
 - Nếu không need-a-space, chỉ cần đưa kí tự $s[i]$ vào chuỗi ans.
1. Khởi tạo chuỗi rỗng: $res = ""$
 2. Khởi tạo biến đếm số kí tự hiện tại của res: $resLen = 0$.
 3. Đánh dấu $needASpace = False$.



BT3 – Gợi ý

4. Duyệt qua các kí tự $s[i]$ trong chuỗi s :

a. Nếu $s[i]$ là khoảng trắng,

i. Nếu độ dài chuỗi res: $resLen > 0$,

- $needASpace = True$



b. Ngược lại ($s[i]$ ko phải khoảng trắng),

i. Nếu $needASpace = True$,

- Copy vào chuỗi res: 1 khoảng trắng và kí tự $s[i]$
- Tăng biến $resLen$ lên 2
- Reset: $needASpace = False$

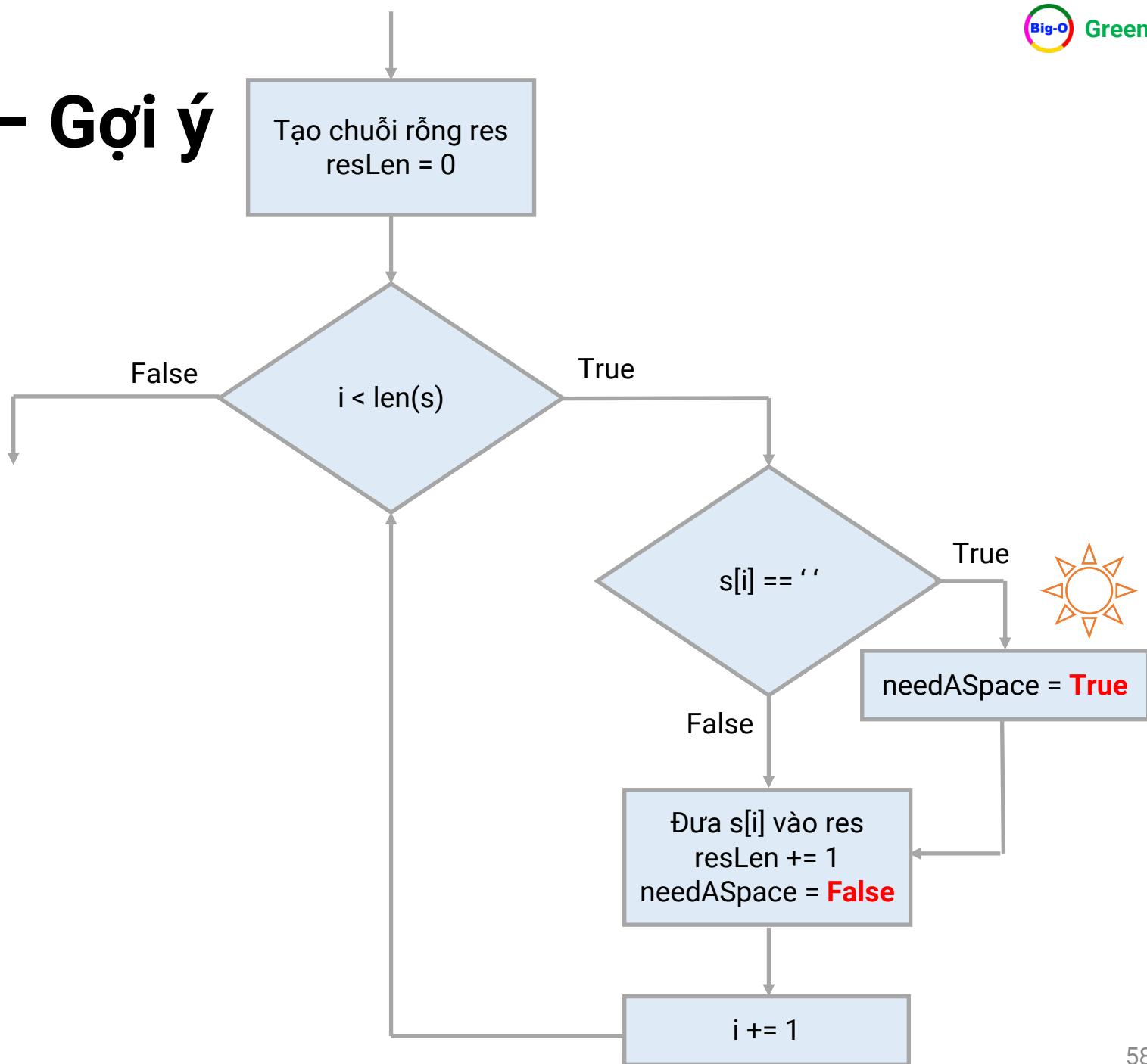


ii. Ngược lại (nếu $needASpace = False$),

- Chỉ cần copy vào chuỗi res kí tự $s[i]$
- Tăng biến $resLen$ lên 1



BT3 – Gợi ý



BT3 – Minh họa

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	B	i	g	o				C	o	d	i	n	g		

```
res = "", resLen = 0, needASpace = False
```

```
s[0] = ''
```

```
s[1] = ''
```

```
s[2] = 'B', needASpace = False
```

```
res = "B", resLen = 1
```

```
s[3] = 'i', needASpace = False
```

```
res = "Bi", resLen = 2
```

```
s[4] = 'g', needASpace = False
```

```
res = "Big", resLen = 3
```

```
s[5] = 'O', needASpace = False
```

```
res = "BigO", resLen = 4
```

BT3 – Minh họa

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	B	i	g	o				C	o	d	i	n	g		

$s[6] = ''$, resLen = 4 > 0

needASpace = True

$s[7] = ''$, resLen = 4 > 0

needASpace = True

$s[8] = 'C'$, needASpace = True

res = "BigO C", resLen = 6, needASpace = False

BT3 – Minh họa

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	B	i	g	o				C	o	d	i	n	g		

s[9] = 'o', needASpace = False

res = "BigO Co", resLen = 7

s[10] = 'd', needASpace = False

res = "BigO Cod", resLen = 8

s[11] = 'i', needASpace = False

res = "BigO Codi", resLen = 9

s[12] = 'n', needASpace = False

res = "BigO Codin", resLen = 10

s[13] = 'g', needASpace = False

res = "BigO Coding", resLen = 11

BT3 – Minh họa

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	B	i	g	o				C	o	d	i	n	g		

s[14] = '', resLen = 11 > 0

needASpace = True

s[15] = '', resLen = 11 > 0

needASpace = True

Kết quả: res = “BigO Coding”

BT4 – IN HOA KÝ TỰ

- Viết chương trình in hoa ký tự đầu tiên của mỗi từ, các kí tự còn lại viết thường.

Kĩ thuật tạo chuỗi mới từ chuỗi ban đầu

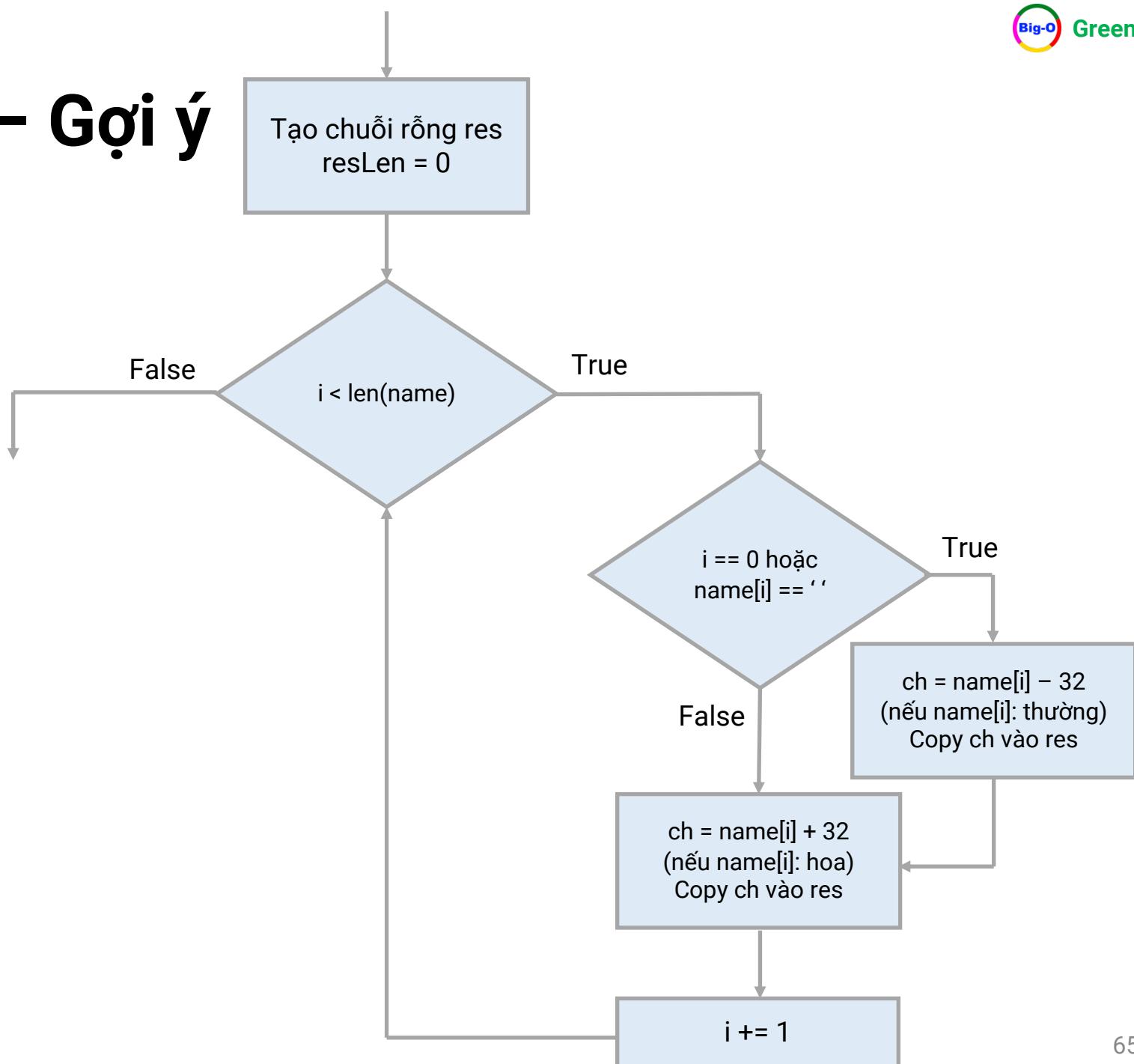


BT4 – Gợi ý

1. Đọc chuỗi name.
2. Duyệt qua từng kí tự name[i] trong chuỗi name.
 - a. Nếu đây là kí tự đầu tiên của mỗi từ, $i == 0$ hoặc $\text{name}[i-1] = ''$,
 - i. Nếu $\text{name}[i]$ **kí tự thường**, ' $'a'$ \leq $\text{name}[i]$ \leq ' $'z'$ ',
 - Chuyển $\text{name}[i]$ thành **kí tự hoa (name[i] - 32)** và copy vào res.
 - ii. Ngược lại, chỉ cần copy $\text{name}[i]$ vào res.
 - b. Ngược lại, nếu $\text{name}[i]$ ko phải là kí tự đầu tiên của từ
 - i. Nếu $\text{name}[i]$ là **kí tự hoa**, ' $'A'$ \leq $\text{name}[i]$ \leq ' $'Z'$ ',
 - Chuyển $\text{name}[i]$ thành **kí tự thường (name[i] + 32)** và copy vào res.
 - ii. Ngược lại, chỉ cần copy $\text{name}[i]$ vào res.
3. In res.



BT4 – Gợi ý



BT4 – Minh họa

0	1	2	3	4	5	6	7	8	9
p	a	u	L		P	o	b	g	A

```
res = ""
i = 0
name[0] = 'p'
i == 0 or name[-1] == ' ': True, res = "P"
```

```
i = 1
name[1] = 'a'
i == 0 or name[0] == ' ': False, res = "Pa"
```

```
i = 2
name[2] = 'u'
i == 0 or name[1] == ' ': False, res = "Pau"
```

```
i = 3
name[3] = 'L'
i == 0 or name[2] == ' ': False, res = "Paul"
```

BT4 – Minh họa

0	1	2	3	4	5	6	7	8	9
p	a	u	L		P	o	b	g	A

i = 4

name[4] = ''

i = 0 or name[3] == ' ': False, res = "Paul "

i = 5

name[5] = 'P'

i == 0 or name[4] == ' ': True, res = "Paul P"

i = 6

name[6] = 'o'

i == 0 or name[5] == ' ': False, res = "Paul Po"

i = 7

name[7] = 'b'

i == 0 or name[6] == ' ': False, res = "Paul Pob"

BT4 – Minh họa

0	1	2	3	4	5	6	7	8	9
p	a	u	L		P	o	b	g	A

i = 8

name[8] = 'g'

i = 0 or name[7] == ' ': False, res = "Paul Pobg"

i = 9

name[9] = 'A'

i == 0 or name[8] == ' ': False, res = "Paul Pobga"

Hỏi đáp

