

# Bài giảng trên lớp

Bài giảng trên lớp là tài liệu gợi ý và hỗ trợ giảng viên trong quá trình lên lớp, tuy nhiên giảng viên cần có sự chuẩn bị của riêng mình cho phù hợp với từng lớp học.

Nếu giảng viên thiết kế bài giảng tốt hơn tài liệu đã cung cấp, xin hãy chủ động làm và gửi lại cho chúng tôi.

Nếu giảng viên cần thay đổi tài liệu đã cung cấp, những thay đổi có liên quan đến cấu trúc, nội dung kiến thức và có tính đổi mới, xin hãy chủ động làm và gửi lại cho chúng tôi.

**Mọi ý kiến xin gửi cho Phòng NC-PTCT để được xem xét và ban hành chính thức và góp phần ngày càng hoàn thiện hơn học liệu của trường.**

Trân trọng cảm ơn.

*Liên hệ: Phòng NC-PTCT – FPT Polytechnic*

*Những ý tưởng đổi mới sẽ được xem xét và gửi qua Dự án CNGD để có sự hỗ trợ giảng viên làm nghiên cứu khoa học, hoặc hướng dẫn viết bài báo đăng trên tạp chí CNGD.*

# Lập trình C++

## Bài 4 Mảng và Xâu ký tự

# Hệ thống bài cũ

- Định nghĩa hàm, gọi hàm.
- Nguyên mẫu hàm.
- Các bước thiết kế hàm.
- Phạm vi.
- Hàm không trả trị.
- Một số hàm toán học.
- Xét chi tiết cách truyền tham số.
- Hàm nội tuyến - hàm inline.
- Nạp chồng hàm.
- Hàm `main()`, hàm nhị phân.

# MỤC TIÊU

- Hiểu về mảng và chuỗi, cách sử dụng nó

# Nội dung

- Khai báo, khởi tạo, và truy xuất mảng một chiều
- Xử lý mảng bằng vòng lặp for.
- Truyền mảng đến hàm.
- Cơ chế typedef.
- Một số phương pháp sắp xếp, tìm kiếm trên mảng.
- Mảng đa chiều.
- Xâu ký tự chuẩn.
- Tập (file).

## Vấn đề

**Ví dụ:** bạn cần cộng 2 số lại bạn làm thế nào?

Có phải : `int a, b,t; t= a + b;`

5 số thì sao: `int a,b,c,d,e,t; t=a+b+c+d+e;`

Vậy 10 số,...,20 số,...100 số, thật không dễ. Vậy giải pháp của chúng ta là gì? => **MẢNG**

Vậy ta sẽ ví dụ cộng 10 số lại được nhập từ bàn phím?



# Giải thuật

- Khai báo mảng **count** có 10 phần tử và khởi tạo các phần tử này bằng 0, biến **tong** kiểu số nguyên khởi tạo bằng 0.
- Nhắc người sử dụng nhập giá trị nguyên.
- Đọc giá trị nguyên vào biến **number**.
- Lặp lại khi  $0 \leq \text{number} \leq 10$ 
  - Tăng phần tử của **count** có chỉ số là **number** lên 1 đơn vị.
  - Nhắc người sử dụng nhập giá trị nguyên.
  - Đọc giá trị nguyên vào biến **count[**number**]**
- Với mỗi giá trị của **i** từ 0...10, hiển thị **count[**number**]** bằng cách lặp lại khi  $0 \leq \text{number} \leq 10$ ; mỗi lần tăng phần tử của **count** ta thực hiện tính tổng: **tong** = **tong** + **count[**number**]**
- Hiện kết quả tổng của 10 số nhập vào **tong**;

# Viết mã và kiểm tra

MangCongNso.cpp

(Global Scope)

main()

```
void print(int[],int);
int sum (int a[], int n);
int main()
{
    const int MAXSIZE=100;
    int a[MAXSIZE]={0}, size;
    cout << " Nhap so phan tu mang size: " ;cin>>size;
    //nhap gia tri cho tung phan tu mang
    read(a,size);
    cout << " Mang co " << size << " phan tu."<<endl;
    int tong =sum(a,size);
    cout << " Tong gia tri cua cac phan tu trong mang: " << tong <<endl;
    system ("pause");
    return 0;
}
//chuong trinh tinh tong mot mang n phan tu
int sum (int a[], int n)
{ int sum=0;
  for (int i=0; i<n; i++) sum += a[i];
  return sum;
}
//nhap so phan tu mang
void read(int a[], int n)
{
    int i = 0;
    cout << " Nhap gia tri cho tung phan tu mang"<<endl;
    do
    { cout << " a[" << i << "]: ";cin >> a[i];
    } while (a[i++] != 0 && i < n);
    --n;    // khong tinh gia tri 0
}
```

C:\Users\hieunh3\Documents\Visual Stu...

```
Nhap so phan tu mang size: 10
Nhap gia tri cho tung phan tu mang
a[0]: 1
a[1]: 2
a[2]: 3
a[3]: 5
a[4]: 2
a[5]: 6
a[6]: 1
a[7]: 9
a[8]: 11
a[9]: 8
Mang co 10 phan tu.
Tong gia tri cua cac phan tu trong mang: 48
Press any key to continue . . .
```



## Kiểm tra, thực thi, phát hiện lỗi

```
Nhap so phan tu mang size: 10
Nhap gia tri cho tung phan tu mang
a[0]: 1
a[1]: 2
a[2]: 3
a[3]: 5
a[4]: 2
a[5]: 6
a[6]: 1
a[7]: 9
a[8]: 11
a[9]: 8
Mang co 10 phan tu.
Tong gia tri cua cac phan tu trong mang: 48
```

# Mảng

## Khái niệm Mảng

- Một mảng dữ liệu gồm một số hữu hạn phần tử có cùng kiểu cơ bản.
- Số phần tử của mảng được xác định ngay khi định nghĩa mảng.

## Khai báo

<Kiểu\_phần\_tử> <Tên\_mảng> <[số phần tử]>;

Ví dụ: `int x[10];` // mảng x gồm 10 phần tử

Các phần tử trong mảng được đánh số thứ tự lần lượt từ 0 đến (số phần tử - 1)

Để truy xuất đến các phần tử của mảng ta dùng cặp dấu [**chỉ số phần tử của mảng**]

Ví dụ: `x[3]`: phần tử thứ 2 trong mảng.

# Mảng

## Khởi tạo mảng

Ví dụ: `const int CAPACITY = 10;`

`int intArray [CAPACITY] = {9, 8, 7, 6, 5, 4, 3, 2, 1, 0};`

Có thể khởi tạo mảng trong khai báo:

Các giá trị phải đặt trong cặp dấu ngoặc móc `{}` và cách nhau bằng dấu phẩy.

Lưu ý là số giá trị khởi tạo trong ngoặc nhọn phải nhỏ hơn hay bằng dung lượng mảng

`intArray`

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

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----


# Mảng

## Khởi tạo mảng

Nếu số giá trị khởi tạo nhỏ hơn dung lượng mảng thì các phần tử còn lại được khởi tạo là 0

```
int intArray [CAPACITY] = { 9, 8, 7, 6, 5, 4 };
```

intArray



9	8	7	6	5	4	0	0	0	0
---	---	---	---	---	---	---	---	---	---

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

- Ứng dụng: `int count [6] = {0};` sẽ làm cho phần tử đầu tiên bằng 0 và các phần tử còn lại cũng bằng 0.
- Nếu số giá trị khởi tạo lớn hơn dung lượng mảng thì lỗi cú pháp.

Các phần tử của mảng có thể có bao nhiêu kiểu dữ liệu khác nhau?

1

0

3

2



**DEMO**

**DEMO Mảng**



# Mảng Ký Tự

- Thừa kế từ ngôn ngữ C, mảng ký tự trong C++ có thể được khởi tạo bằng hai cách:
- `char name[10] = {'H','o','a','i',' ','T','h','u'};`
- `char name[10] = "Hoai Thu";`
- Trình biên dịch tự động gán ký tự null ('\0') vào vị trí cuối cùng của mảng ký tự

name

H	o	a	i		T	h	u	\0	\0
---	---	---	---	--	---	---	---	----	----

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

# Xử Lý Mảng Bằng Vòng for

Vòng lặp **for** thường được sử dụng để xử lý các phần tử trong mảng thông qua chỉ số của chúng:

```
for (int i = 0; i < CAPACITY; i++) // ...thao tác với someArray[i]
```

Những tác vụ thường được thực hiện với các phần tử trong mảng:

Tính tổng giá trị của các phần tử

Khởi tạo giá trị cho các phần tử

Tăng, giảm, tính phần trăm, ...



# Xử Lý Mảng Bằng Vòng for

Các thao tác trên mảng

## Nhập mảng

Với n là số phần tử của mảng, đoạn chương trình sau thực hiện nhập mảng a.

```
for( int i = 0; i < n; i++){  
    cout<<"Nhap a[ "<<i<<" ] = ";  cin>>a[i];  
}
```

## Xuất mảng

Xuất mảng a sau khi đã được nhập ở phần trên:

```
for( i = 0; i < n; i++)  
    cout<< a[ i]<< "\\t";
```



**DEMO**

**DEMO Xử lý  
Mảng bằng lệnh for**



# Truyền Mảng vào một Hàm

**Ví dụ:** Truyền mảng vào một hàm và trả về tổng số

```
int arrSum ( int[], int );
```

```
int main ( )
```

```
{  
    int a[] = { 11, 33, 55, 77 };  
    int size = sizeof(a)/sizeof(int);  
    cout << "sum(a,size) = " << arrSum (a,size) << endl;  
}
```

```
int arrSum(int a[], int n)
```

```
{  
    int sum = 0;  
    for (int i= 0; i<n; i++)  
        sum += a[i];  
    return sum;  
}
```

Nên có tham số thứ hai để chỉ định kích thước mảng khi truyền mảng đến hàm

Sử dụng tên mảng (không có dấu []) như là đối số để truyền đến hàm

Tham số của hàm gồm kiểu, tên mảng, dấu [] và kích thước của mảng

## Cơ chế typedef

Định nghĩa tên mới cho kiểu dữ liệu đã có.

Ví dụ: Định nghĩa tên mới **IntegerArray** có kiểu dữ liệu là mảng

Sử dụng tên mới trong các khai báo khác.

```
const int CAPACITY = 6;  
typedef int IntegerArray [ CAPACITY ];
```



...

```
IntegerArray count = {0};
```

hay

```
void initArray ( IntegerArray list, int listSize);
```

Cú pháp: **typedef** *type* *alias*;

*type* : kiểu được đưa ra.

*alias* : tên mới.

# Lỗi tràn chỉ số mảng

Trong một số ngôn ngữ lập trình, một biến chỉ số sẽ không được phép vượt quá một giá trị biên được thiết lập từ định nghĩa của mảng. Nếu tham chiếu vượt qua giá trị biên đó sẽ làm ngắt chương trình.

Trong C++ không tồn tại cơ chế an toàn này.

Ví dụ:

```
int main ()
{ const int SIZE=4;
float a[SIZE] = { 33.3, 44.4, 55.5, 66.6 };
for (int i=0; i<7; i++) // LỖI: chỉ mục vượt quá giới hạn!
cout << "\ta[" << i << "] = " << a[i] << endl;
}
```

Lỗi tràn chỉ số mảng gây ra tác dụng phụ không mong muốn. Đây là lỗi chạy chương trình nguy hiểm nhất. Do vậy lập trình viên phải chú ý trong quá trình sử dụng mảng.



**DEMO**

**DEMO truyền Mảng  
vào một hàm**



# Tìm kiếm Tuyến Tính

Tìm lần lượt các phần tử trong mảng:

- Bắt đầu từ phần tử đầu tiên.
- Tiếp tục cho đến khi tìm thấy hoặc không đến cuối mảng.

arr\_a

22	13	45	11	67	55	32	80	41	75
----	----	----	----	----	----	----	----	----	----

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

Tìm kiếm lần lượt các phần tử trong danh sách. Lưu ý **Giá trị cần tìm** có thể không có.

15



**DEMO**

**DEMO tìm kiếm  
tuyển tính**

A white hand cursor icon, resembling a computer mouse pointer, is positioned to the right of the text. The index finger is pointing upwards towards the 'O' in the word 'DEMO'.



# Sắp xếp nổi bọt

Giả sử cần sắp theo thứ tự tăng dần.

Duyệt nhiều lần qua các phần tử trong mảng:

- Trong bước thứ  $i$  ( $i = 1, \dots, n-1$ ) đổi chỗ những phần tử tử lớn nhất tiếp theo vào đúng vị trí của nó.
- Nghĩa là so sánh một cặp phần tử liền kề và di chuyển phần tử lớn hơn lên trước,  $a[i-1] > a[i]$ .

Thể hiện bằng chương trình như sau:

```
void sort (float a[], int n)
{ // sắp xếp nổi bọt
    for (int i=1; i<n; i++)
        // làm nổi giá trị max{a[0..n-i]}
        for (int j=0; j<n-i; j++)
            if (a[j] > a[j+1]) swap (a[j],a[j+1]);
}
```

Ví dụ 1 mảng các dãy số:      22, 13, 45, 11, 67, 55, 32, 80, 41, 75  
Kết quả sau khi sắp xếp:      11, 13, 22, 32, 41, 45, 55, 67, 75, 80



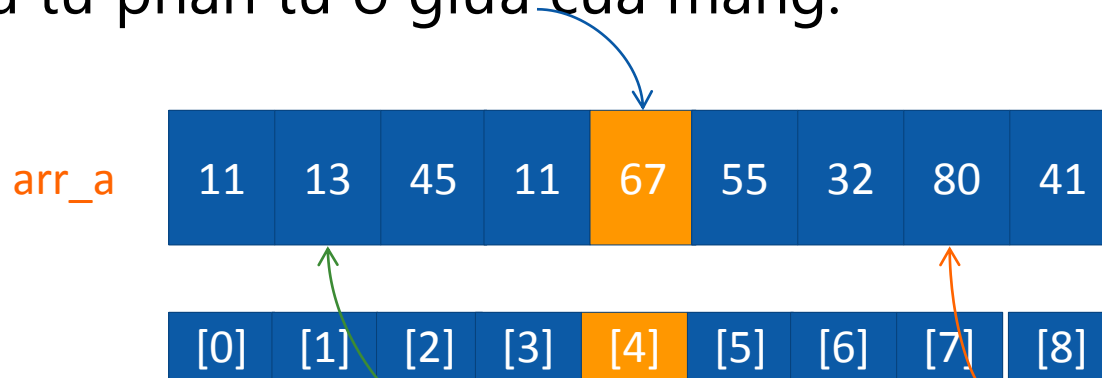
**DEMO**

**DEMO sắp xếp  
nổi bật**



# Tìm kiếm nhị phân

- Khi mảng có thứ tự, tìm kiếm nhị phân hiệu quả hơn.
- Tìm kiếm nhị phân sử dụng chiến lược “chia để trị”.
- Nó lặp lại việc chia mảng thành hai phần và sau đó tìm trong các phần có khả năng chứa giá trị cần tìm.
- Bắt đầu từ phần tử ở giữa của mảng.



Lặp lại quá trình tìm kiếm trong danh sách con nếu cần.  
Lưu ý **Giá trị cần tìm** có thể không tìm thấy

55

- Nếu giá trị của phần tử ở giữa nhỏ hơn giá trị cần tìm kiếm, việc tìm kiếm thực hiện trên **danh sách con bên phải**.
- Nếu giá trị của phần tử ở giữa lớn hơn giá trị cần tìm kiếm, việc tìm kiếm thực hiện trên **danh sách con bên trái**.



**DEMO**

**DEMO tìm kiếm  
nhị phân**

A white hand cursor icon, resembling a computer mouse pointer, is positioned below the word 'DEMO' and points upwards towards the letter 'O'.

# Mảng đa chiều

- Một mảng của mảng được gọi là **mảng đa chiều**. Hay mảng đa chiều là mảng có từ **2 chiều trở lên**.
- Người ta thường sử dụng mảng đa chiều để lưu các ma trận, tọa độ 2 chiều, 3 chiều...
- Khai báo mảng:  
<**kiểu**> <**tên mảng**> [**số phần tử**][**số phần tử**]...[**số phần tử**];
- Truy xuất từng phần tử của mảng:  
<**tên mảng**> [**chỉ số phần tử**][**chỉ số phần tử**]...[**chỉ số phần tử**];

Ví dụ:

Khai báo mảng 3 chiều như sau: **double** a[32][10][4];

Truy xuất vào phần tử của mảng: a[1][2][0] = 355;



**DEMO**

**DEMO mảng  
đa chiều**

A white hand cursor icon, resembling a computer mouse pointer, is positioned below the word "DEMO". The index finger is pointing upwards towards the letter "O" in "DEMO".

# Xâu ký tự

Theo định nghĩa trong C thì xâu:

- Là một mảng các ký tự
- Tất cả các xâu ký tự đều kết thúc với ký tự **null** (' \0 ')
- Cú pháp khai báo: **char** <tên chuỗi>[độ dài tối đa]
- Ví dụ khai báo:

- **char** string1[255] = "hello";  
Ký tự **null** được tự động bổ sung.  
**string1** có 6 ký tự
- **char** string1[] =  
{ 'h', 'e', 'l', 'l', 'o', '\0' };

- Truy cập các phần tử của xâu:

Có thể truy cập vào từng ký tự của xâu với tên biến và chỉ số đặt trong ngoặc vuông như truy cập đến từng phần tử mảng. Chỉ số xâu chạy từ 0 đến độ dài tối đa của xâu ký tự.

# Kiểu dữ liệu `string`

- `string` là một lớp chuẩn trong C++
- Các ký tự trong `string` được đánh từ 0
- Khởi tạo một biến kiểu `string` như sau:
  - `string s1 ("Man");`
  - `string s2="Beast";`
  - `string s3;`
- Ta có thể sử dụng các toán tử toán học, logic ... trên đối tượng `string`
  - `s3 = s1;`
  - `s3 = "Neither" + s1 + "nor";`
  - `s3 += s2;`



## string và toán tử >>

Hàm `getline` (`cin`, `string str`): cho phép nhập toàn bộ xâu ký tự `str` kể cả khoảng trắng.

Ví dụ:

```
string full_name;  
cout << "Enter your fullname:";  
getline (cin, full_name);  
// full_name: lưu thông tin mà người sử dụng nhập từ bàn phím
```

Phải `#include <string>` để sử dụng hàm `getline`

# Các hàm xử lý chuỗi

## Nhập chuỗi

Khi khai báo một biến mảng `hoten`: `char hoten[30];`  
có hai cách nhập chuỗi.

`cin >> hoten;` // gặp ký tự trắng thì kết thúc việc nhập

`gets (hoten);` // máy vẫn hiểu khi gặp ký tự trắng. Kết thúc việc nhập bằng phím Enter

## Xuất chuỗi

Dùng lệnh `cout << [tham số];`



**DEMO**

**DEMO** xâu ký tự



# TỆP (FILE)

- Việc xử lý tệp trong C++ là rất giống với tương tác đầu vào và đầu ra thông thường vì sử dụng cùng đối tượng **stream**.
- Các thao tác trên tệp:
  - Đọc từ tệp là khi chương trình lấy input.
  - Ghi vào tệp là khi chương trình truyền output ra.
- Để đọc một tệp được quản lý bởi đối tượng **ifstream**.
- Để ghi một tệp được quản lý bởi đối tượng **ofstream**.
- Phải `#include` thư viện **<fstream>**.

## Sử dụng ofstream trong thư viện fstream để ghi file

Ví dụ tạo ra một file .txt rồi ghi vào đó một đoạn text tùy ý.

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main ( )
```

```
{
```

```
    ofstream FileDemo ("File_Demo.txt");
```

```
    FileDemo << "File demo su dung ofstream de ghi file. ";
```

```
    FileDemo.close();
```

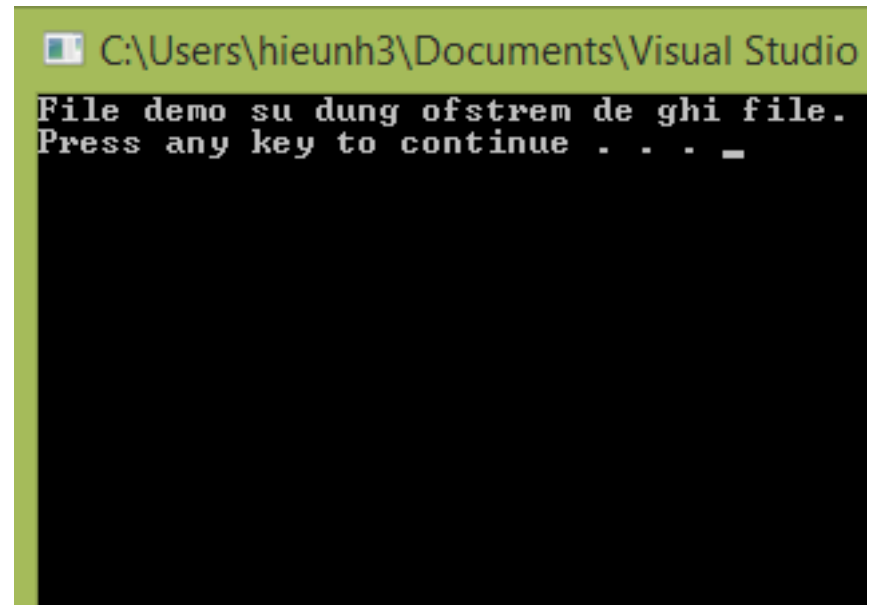
```
    return 0;
```

```
}
```

# Sử dụng ifstream trong thư viện fstream để đọc file

Ví dụ: Ở trên ta ghi ra một file "File Demo.txt" rồi, giờ ta sẽ đọc file đó bằng ifstream;

```
#include <iostream>
#include <fstream>
#include <iostream>
using namespace std;
int main ( ) {
    char a[50];
    ifstream FileDemo ("File_Demo.txt");
    if(! FileDemo.is_open()) {
        cout<<"Khong the mo file.\n";
        return 0;
    } else {
        while(!FileDemo.eof()){
            FileDemo >> a;  cout << a<< " ";
        }
    }
    FileDemo.close(); cout <<endl;
    system("pause");
    return 0;
}
```



```
C:\Users\hieunh3\Documents\Visual Studio
File demo su dung ofstrem de ghi file.
Press any key to continue . . . _
```

# Một số hàm xử lý chung

Tên phương thức	Chức năng
open	Mở tệp tin
close	Đóng tệp tin
is_open	Kiểm tra tệp tin có mở hay không
flush	Làm sạch vùng đệm tệp tin đang mở
good	Kiểm tra trạng thái của tệp tin có tốt để đọc/ghi không?
eof	Kiểm tra xem đã đọc đến cuối tệp tin chưa

# Một số hàm xử lý cho tệp tin văn bản


Tên phương thức	Chức năng
operator >>	Đọc dữ liệu từ tệp tin
operator <<	Ghi dữ liệu từ tệp tin
getline	Đọc một dòng dữ liệu từ tệp tin
get	Đọc 1 ký tự hoặc 1 chuỗi từ tệp tin, tùy thuộc vào tham số của phương thức
put	Ghi một ký tự xuống tệp tin





**DEMO**

**DEMO tệp (file)**

A white hand cursor icon, resembling a computer mouse pointer, is positioned below the word 'DEMO' and points upwards towards the letter 'O'.

# Kết luận

- Khai báo, khởi tạo, và truy xuất mảng một chiều.
- Xử lý mảng bằng vòng lặp for.
- Truyền mảng đến hàm.
- Cơ chế typedef.
- Một số phương pháp sắp xếp, tìm kiếm trên mảng.
- Mảng đa chiều.
- Xâu ký tự chuẩn.
- Tập (file).

## Chuẩn bị bài sau

Sinh viên đọc sách và slide trước bài học kế tiếp về Con trỏ và tham chiếu gồm:

- Con trỏ và tham chiếu.
- Con trỏ hằng.
- Con trỏ hàm.
- Mảng và con trỏ.



**FPT POLYTECHNIC**

THANK YOU!

[www.poly.edu.vn](http://www.poly.edu.vn)