# BÀI THỰC HÀNH SỐ 1

## 1. MỤC TIÊU:

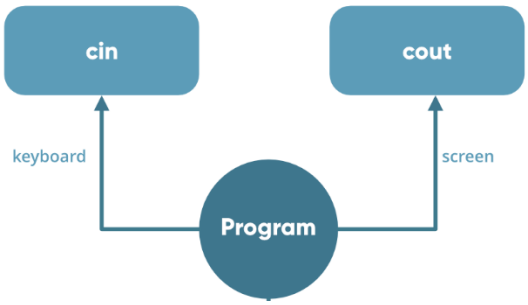
- Giúp sinh viên làm quen với visual studio 2015
- Làm quen với cấu trúc chung của một chương trình C++ đơn giản và các lệnh nhập xuất dữ liệu
- Làm việc với danh sách đặc
- Giúp sinh viên phát hiện được các lỗi cơ bản khi lập trình

## 2. LÝ THUYẾT CẦN GHI NHỚ

- Cấu trúc cơ bản 1 chương trình C++ gồm có

```
#include <iostream>   // gọi chỉ thị tiền xử lý
using namespace std;  //sử dụng standard namespace (namespace định nghĩa ác class, object, function...) (1)
int main()
{
    cout << "Hello World!\n";     // nếu không định nghĩa ở (1) thì ở dòng này std::cout << "Hello World!\n";
}
```

- Các lệnh xuất nhập cơ bản:




- Danh sách đặc: danh sách mà các phần tử trong danh sách có cùng kiểu dữ liệu, và được cấp phát liên tục trong bộ nhớ, số lượng phần tử tối đa phải được khai báo trước. Truy xuất các phần tử trong danh sách đặc thông qua chỉ số index (index bắt đầu từ 0 đến n-1 với n là số lượng phần tử trong danh sách)
- Khai báo danh sách đặc:
  - Cú pháp khai báo tường minh:

    **<kiểu cơ sở> <tên danh sách> [<số phần tử>]**

    - `int songuyen [10] ;`
    - `float float sothuc [15] ;`
  - Cú pháp khai báo không tường minh:
- **<Kiểu> <Tên danh sách> []= {Các giá trị cách nhau bởi dấu phẩy}**
    - Ví dụ: `float x[] = {12.1 , 7.23 , 5.0 , 27.6 , 87.9 , 9.31};`
    - Hoặc có thể khai báo danh sách là tham số hình thức của một hàm:
      - `void nhap (int a[], int n)`

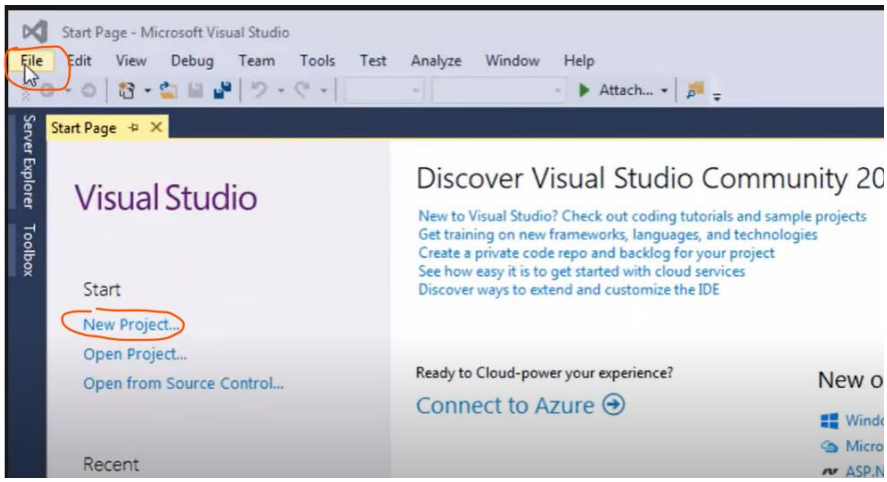
- Các thao tác cơ bản trên danh sách đặc:
  - Khai báo cấu trúc
  - Nhập n phần tử vào danh sách
  - Xuất danh sách
  - Tìm một phần tử trong danh sách
  - Xóa phần tử trong danh sách
  - Liệt kê các phần tử trong danh sách đặc theo điều kiện
- Các lỗi thường gặp:
  - Sai cú pháp ( thiếu dấu ; cuối dòng lệnh)
  - Truy xuất đến biến trong khi chưa gán giá trị
  - Chỉ số index < 0 hoặc vượt quá n

## 3. BÀI TẬP THỰC HÀNH CƠ BẢN

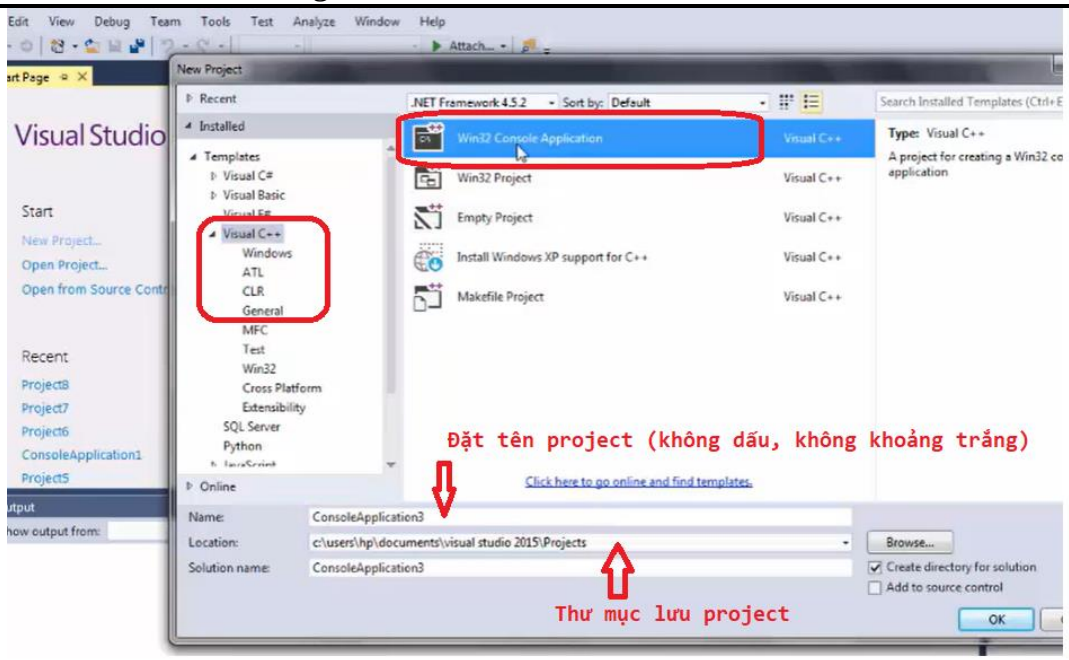
**Hướng dẫn:**

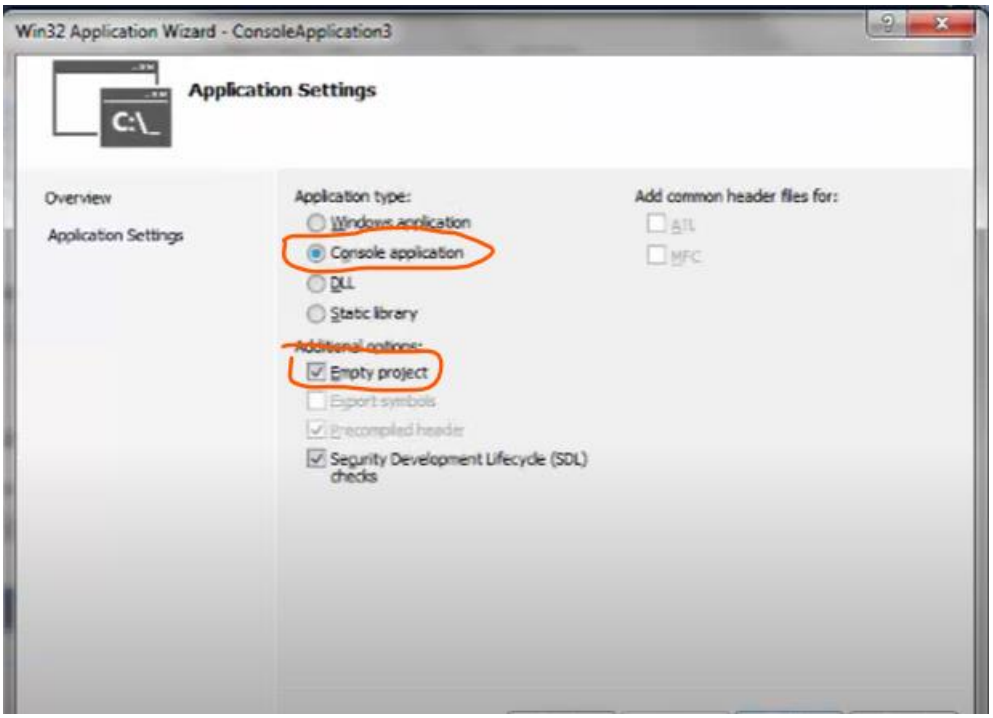
1. **Tạo project**

- Mở visual studio 2015, chọn "Create a new project" như hình hoặc vào File ➔ Create New Project

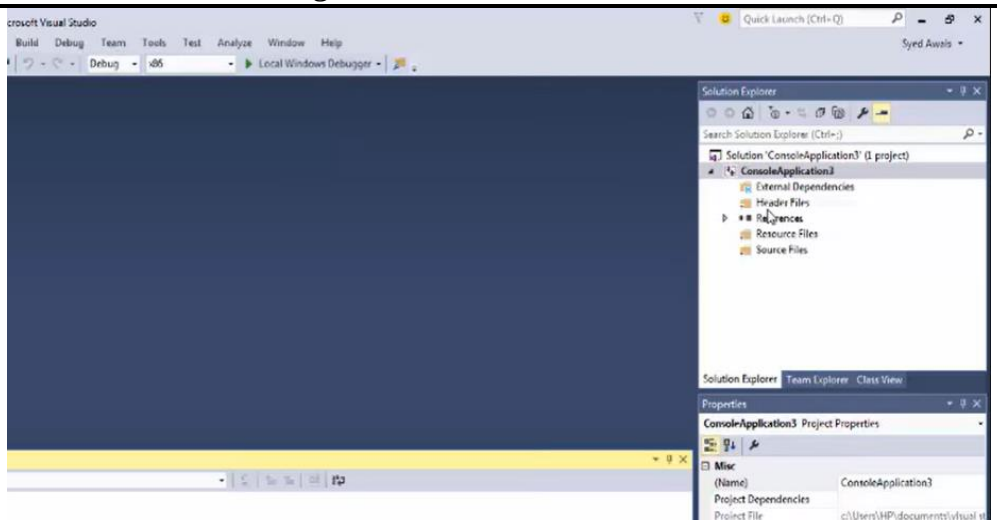



Chọn Win32 Console Application,

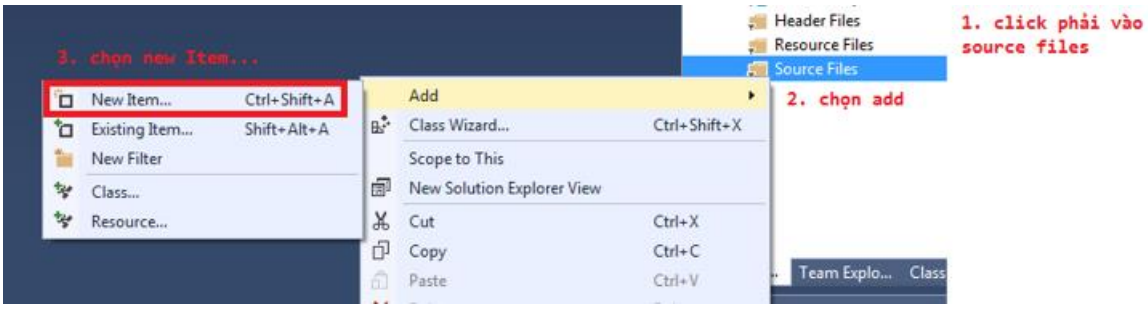


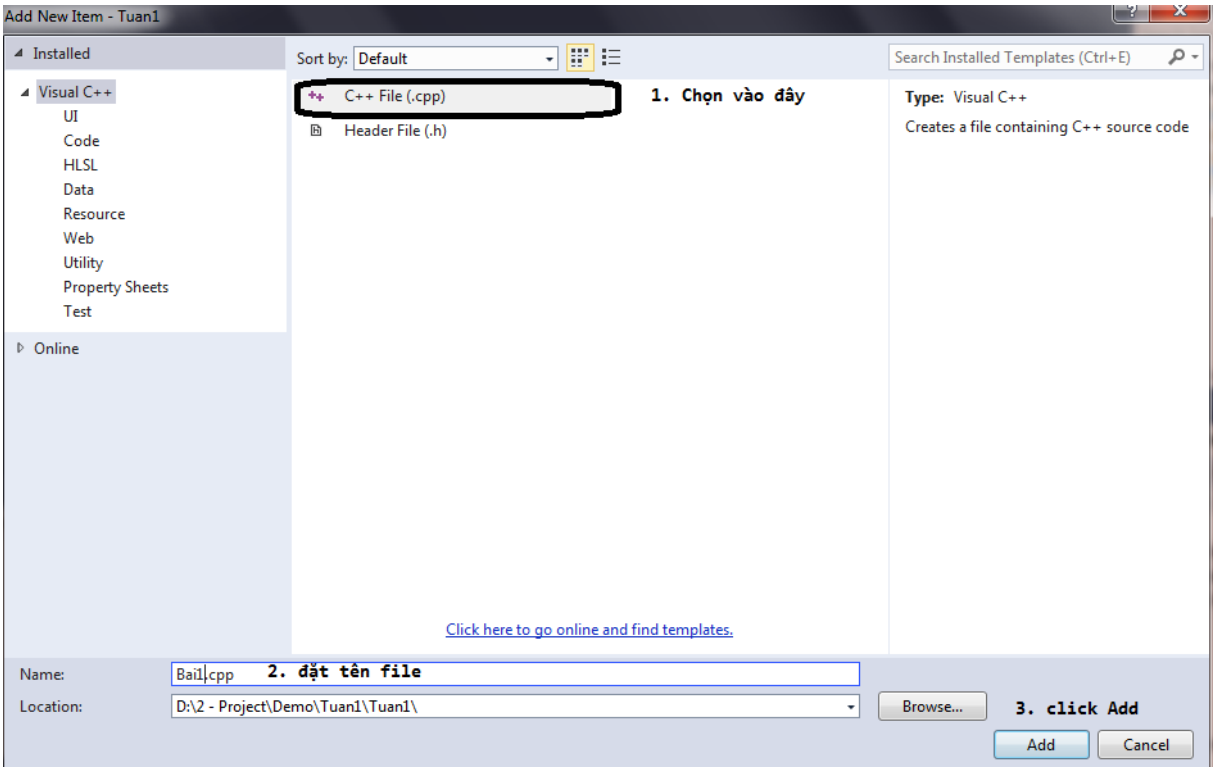

- Project sau khi được tạo



- Tạo file .cpp



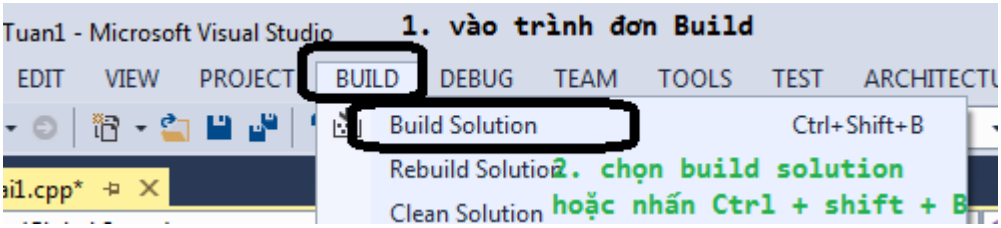

- Trong cửa sổ Create New Item




- Trong file vừa mới tạo, nhập đoạn code sau:

```
#include <iostream>
using namespace std;
int main()
{
    cout<<"Hello World";
}
```

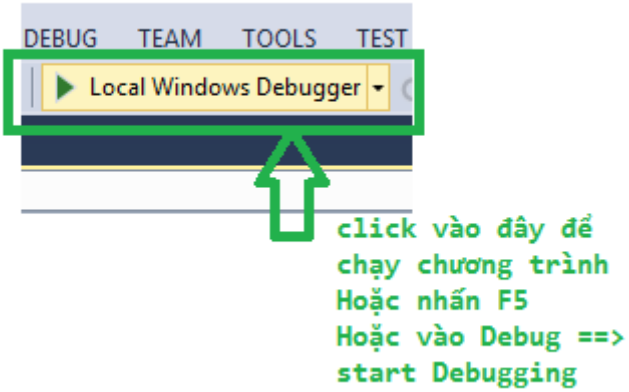
- Build chương trình:



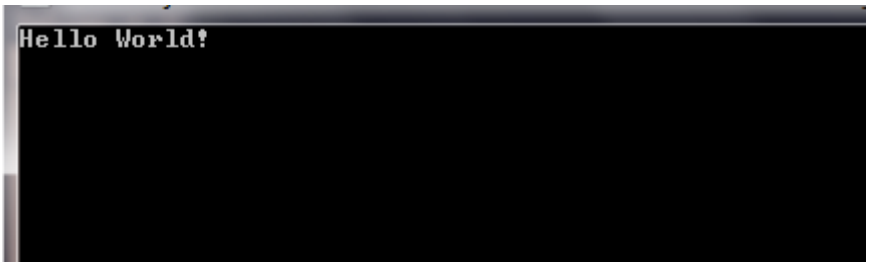

- Kết quả sau khi build chương trình

```
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

- Chạy chương trình:




- Kết quả hiển thị lên màn hình:




## Bài 1:

Quản lý một danh sách có tối đa 100 phần tử, mỗi phần tử trong danh sách có kiểu int. (Danh sách không có thứ tự)

(SV có thể khai báo nguyên mẫu hàm, sau đó viết thủ tục)

1.1. Khai báo cấu trúc danh sách

```
#define MAX 100
int a[MAX];
int n;
```

- Viết thủ tục nhập n, với n<100 (SV có thể viết hàm trả về n thay vì hàm void như hướng dẫn)

```
void nhapn(int& n) {
     do {
       // SV tự code
     } while (biểu thức điều kiện);
}
```

1.2. Viết thủ tục nhập danh sách

- Sử dụng vòng lặp for (while) để nhập từng phần tử vào danh sách. Lưu ý số lượng phần tử của danh sách là n

```
void input(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        // SV tự viết code
    }
}
```

1.3. Viết thủ tục xuất danh sách ra màn hình

Sử dụng vòng lặp for(hoặc while) để duyệt từng phần tử trong danh sách và in ra màn hình (số lượng phần tử trong danh sách là n)

** Lưu ý: SV xử lý cho trường hợp in ra màn hình các phần tử trong danh sách trong 2 trường hợp:

- In trên 1 dòng
- In trên nhiều dòng

```
void output (int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        // SV tự viết code
    }
}
```

1.4. Viết thủ tục tìm một phần tử trong danh sách. Tính độ phức tạp của thuật toán.

- Sử dụng vòng lặp for hoặc while để duyệt qua các phần tử trong danh sách
- Sử dụng if để kiểm tra phần tử x có bằng với bất kỳ phần tử nào trong danh sách không

```
int search (int a[], int n, int x)
{
        // hàm trả về -1 nếu không tìm thấy, trả về vị trí i
nếu tìm thấy ➔ sinh viên tự viết code
}
```

1.5. Viết thủ tục thêm một phần tử vào cuối danh sách.

- Nếu n (số phần tử hiện tại của danh sách) <MAX thì gán giá trị của x vào vị trí này, sau đó tăng giá trị của n lên 1

```
int InsertEnd(int a[], int& n, int x) // x là pt thêm vào
{
    if ( n < MAX)
    {
        // SV tu viet code
        return 1; // them thanh cong
    }
    return 0; // them that bai
}
```

1.6. Viết thủ tục xóa phần tử cuối danh sách.

Giảm số lượng phần tử trong danh sách xuống 1 đơn vị

```
int DeleteEnd(int a[], int& n)
{
    if (n >0)
    {
// SV tự viết code
        return 1;
    }
    return 0;
}
```

1.7. Viết thủ tục xóa phần tử tại vị trí thứ i. Tính độ phức tạp của thuật toán.

- Dời các phần tử phía sau I (bắt đầu từ i+1) lên phía trước 1 bước
- Giảm n xuống 1 đơn vị

```
int Delete(int a[], int& n, int i)
{
    if (i >= 0 && i < n)
    {
        // viết dòng for để dời các phần tử về phía trước
        n--; // giảm n xuống
        return 1;
    }
    return 0;
}
```

1.8. Tìm một phần tử trong danh sách. Nếu tìm thấy, xóa phần tử đó. (Tính độ phức tạp của thuật toán) (*)

- Bài tập này là sự kết hợp của bài 1.4 và 1.7. SV tự viết code

## Bài 2:

Quản lý một danh sách có thứ tự tối đa 100 phần tử, mỗi phần tử trong danh sách có kiểu int. (Danh sách đặc)

2.1. Khai báo cấu trúc danh sách

- SV thực hiện tương tự như câu 1.1

- SV viết hàm nhập vào một danh sách tăng dần (số nhập sau lớn hơn số nhập trước )

2.2. Viết thủ tục thêm một phần tử x vào danh sách (thêm một phần tử vào vị trí phù hợp trong danh sách đã có thứ tự). Lưu ý: Không xếp thứ tự danh sách.

- Duyệt từ cuối danh sách về đầu danh sách
  - Nếu các phần từ này > số cần thêm vào thì dời các phần tử này về phía sau 1 vị trí
  - Gán giá trị x vào vị trí phù hợp (vị trí [mà a[i] bắt đầu nhỏ hơn x ]+ 1)
  - Tăng giá trị n lên 1

```
int insertSorted(int arr[], int &n, int x)
{

    if (n >= MAX)
        return -1; // insert khong thanh cong

    int i;
     // SV viết vòng for để dời phần tử và gán giá trị mới

    n++;
    return 1; // insert thanh cong

}
```

## 4. BÀI TẬP NÂNG CAO

### Bài 1:

- Viết thủ tục tạo danh sách đặc A gồm n phần tử (n>0), với yêu cầu giá trị của các phần tử của danh sách đặc thỏa điều kiện đều là số dương
    - Mở rộng cho trường hợp số âm, số chẵn, số lẻ, số vừa âm vừa lẻ …
    - Trường hợp các các phần tử trong danh sách chỉ nằm trong khoảng từ 10 – 50

### Bài 2:

- Viết thủ tục tạo danh sách đặc A gồm n phần tử (n>0), với yêu cầu những vị trí (hay chỉ số của danh sách) là số lẻ chỉ nhận giá trị nhập vào là số lẻ, và những vị trí (hay chỉ số của danh sách) là số chẵn chỉ nhận giá trị nhập vào là số chẵn

### Bài 3:

- Viết thủ tục tạo danh sách đặc A gồm n phần tử (n>0), với yêu cầu chỉ cho người dùng nhập giá trị của các phần tử của danh sách là số nguyên tố
    - *Mở rộng* cho các trường hợp số hoàn thiện, số chính phương, …

### Bài 4:

- Viết thủ tục liệt kê các phần tử trong danh sách có giá trị lẻ.
    - *Mở rộng*: giá trị chẵn, giá trị âm, giá trị dương, …

## 5. BÀI TẬP VỀ NHÀ

### Bài 1:

- Viết thủ tục nhận tham số là số x, và in ra tất cả các số có vị trí lớn hơn x trong danh sách.
    - *Mở rộng*: khác X, nhỏ hơn hay bằng X, bội số của X, ước số của X, …).

### Bài 2:

- Viết thủ tục liệt kê các phần tử trong danh sách có vị trí là số nguyên tố.
    - *Mở rộng* cho các trường hợp số đối xứng, số may mắn, số chính phương, số hoàn thiện, …