Kí tự và chuỗi kí tự

- Mục đích sử dụng
- Các đặc trưng
- Đọc vào cả dòng với gets, fgets và scanf
- Ví dụ minh họa

Mục đích sử dụng

- Dùng kiểu char để lưu trữ một kí tự đơn
- Dùng mảng/con trỏ char để lưu trữ một chuỗi kí tự
- Kí tự thường là kí tự hằng đặt trong dấu nháy đơn. Ví dụ ‘a’.
- Bản chất của 1 kí tự là 1 giá trị số nguyên đại diện cho kí tự đó trong bộ kí tự của máy tính. Bảng mã hay sử dụng để đối chiếu kí tự là ASCII
- Chuỗi kí tự là một tập các kí tự được sắp xếp theo một trật tự nào đó nhằm biểu đạt một ý nghĩa cụ thể nào đó và được coi là một đơn vị đơn nhất.

Đặc điểm của chuỗi kí tự

- Chuỗi kí tự hằng được bao bởi dấu nháy kép “”. Ví dụ “Tran Van A”
- Chuỗi kí tự có thể chứa chữ cái, chữ số, các kí tự đặc biệt và các kí tự khác
- Chuỗi kí tự thường kết thúc bởi kí tự null ‘\0’
- Độ dài chuỗi kí tự không giới hạn tuy nhiên ta không nên tạo chuỗi quá dài
- Ta thường dùng con trỏ char để thao tác với chuỗi kí tự

Gán chuỗi kí tự

- Ta có thể khởi tạo chuỗi kí tự bằng cách gán giá trị cho mảng char hoặc con trỏ char
- Ví dụ:

```
char name[] = "Nam";
const char* namePtr = "Nhung";
```

- Cú pháp trên tương đương với:

```
char name[] = {'N', 'a', 'm', '\0'};
```

- Luôn nhớ kết thúc chuỗi kí tự với kí tự null '\0' để khi in chuỗi kí tự với hàm printf sẽ không gặp lỗi
- Hàm printf khi in chuỗi kí tự, sẽ chỉ dừng việc in ra cho tới khi gặp kí tự kết thúc chuỗi, tức '\0'

Đọc vào kí tự và chuỗi kí tự với scanf

- Để đọc vào một kí tự đơn ta dùng định dạng %c
- Để đọc vào/in ra một chuỗi kí tự ta dùng %s
- Khi đọc chuỗi, hàm scanf sẽ chỉ dừng khi gặp khoảng trắng(dấu cách, tab, \n)
- Để tránh việc tràn bộ nhớ, tức đọc vào số phần tử quá khả năng lưu trữ(n) của mảng kí tự, ta giới hạn việc đọc vào chỉ n-1 kí tự
- Ví dụ:

```
char name[20];
printf("%s", "What your name? ");
scanf("%19s", name); // doc vao mot tu
// hien thi ten ra man hinh
printf("Hello %s!\n", name);
```

Đọc vào cả dòng

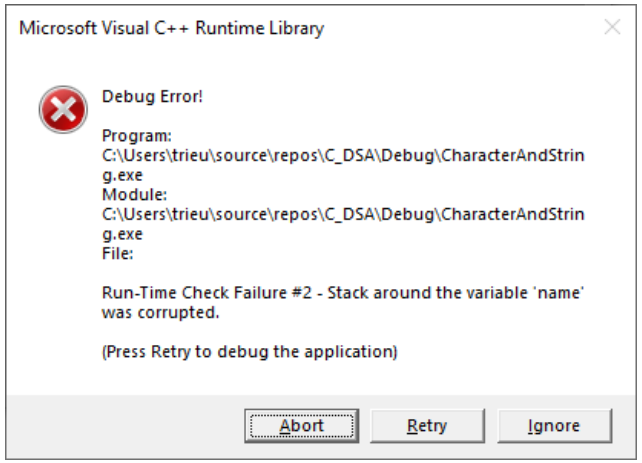
- Ta có thể đọc vào cả dòng sử dụng hàm scanf, gets và fgets
- Nếu dùng gets thì có thể xảy ra tràn mảng khi số lượng kí tự đọc vào lớn hơn khả năng lưu trữ của mảng
- Nếu dùng fgets ta không sợ tràn mảng vì đã có tham số kiểm soát lượng kí tự nhập vào tối đa. Nhược điểm của phương pháp này là nó sẽ đọc cả kí tự '\n'
- Nếu dùng scanf: có thể xảy ra tràn mảng nhưng có cách khắc phục là giới hạn số lượng kí tự tối đa có thể đọc vào. Hàm này sẽ dừng việc đọc vào khi gặp kí tự khoảng trắng(tab, dấu cách, \n)

Đọc vào cả dòng với gets

- Để đọc cả dòng với hàm gets() ta đưa tên mảng sẽ chứa dữ liệu vào làm tham số của hàm:

```
char name[20];
printf("%s", "What your name? ");
gets(name); // doc vao ca dong
```

- Hàm này chỉ đọc và gán cho 1 tham số tại 1 lần gọi hàm
- Nếu ta nhập vào quá số phần tử mảng có thể chứa, lỗi sẽ xảy ra, ví dụ trên visual studio:



Đọc vào cả dòng với fgets

- Hàm fgets() nhận vào ba tham số: mảng đích, số kí tự tối đa được phép đọc vào và chuẩn đầu vào
- Ví dụ:

```
char name[20];
printf("%s", "What your name? ");
fgets(name, 19, stdin); // doc vao ca dong
```

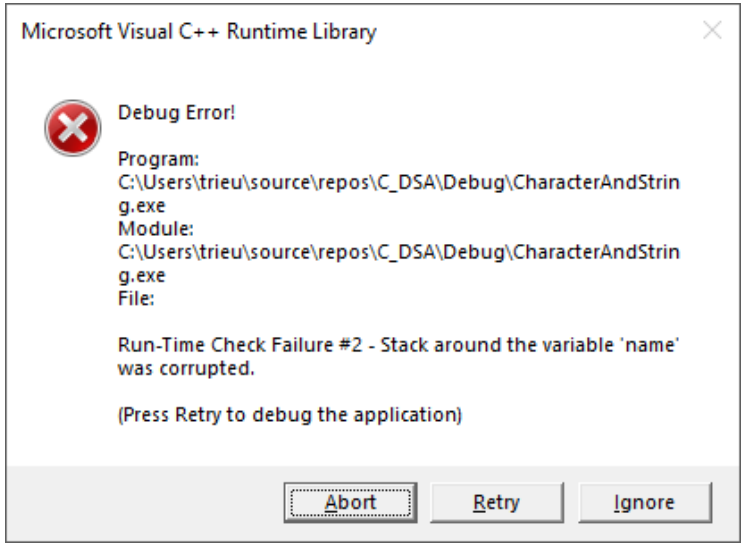
- Hàm này cũng chỉ đọc và gán cho 1 tham số tại 1 lần gọi hàm
- Nếu ta nhập vào quá số phần tử mảng sẽ không lo bị lỗi
- Vấn đề duy nhất gặp phải khi dùng hàm này là kí tự '\n' ở cuối chuỗi. Ta gán nó thành '\0' là xong

Đọc vào cả dòng với scanf

- Để đọc cả dòng với hàm scanf() ta kết hợp giữa giới hạn số kí tự cần đọc và [^\n] như sau:

```
char name[20];
printf("%s", "What your name? ");
scanf("%19[^\n]", name); // doc vao ca dong
```

- Nếu không có giới hạn số kí tự tối đa thì khi nhập quá khả năng lưu trữ có thể bị lỗi, ví dụ: scanf("%[^\n]", name);



Tiếp theo

Khử trôi lệnh khi đọc cả dòng