A screenshot of a computer

Description automatically generated

- Cách scanf một ký tự char:

char d;

scanf("%c", &d); không như scanf mảng ký tự, scanf ký tự vẫn cần dấu &.

printf("%c\n", d);

- Khi scanf chuỗi kí tự a , lúc scanf ko cần &a do a là 1 con trỏ.

- trước lệnh scanf có phím enter thì lệnh scanf vẫn chạy như thường

- lệnh scanf gặp dấu cách sẽ tự ngắt NÊN khi muốn scanf “nguyen anh tuan” thì nó chỉ quét đến nguyen.

char a[100];

scanf("%s", a);

printf("%s\n", a);

}

- cách scanf đúng 10 kí tự đầu (chưa hết 10 kí tự mà gặp dấu cách thì vẫn sẽ ngắt do là lệnh scanf);

int main(){

char a[100];

scanf("%10s", a);

printf("%s", a);

}

- in từng kí tự trong xâu: dùng for và strlen(a)

int main(){

char a[100];

scanf("%s", a);

for(int i = 0; i < strlen(a); i++){

printf("%c", a[i]);

}

}

- lệnh scanf không quét được chuỗi có dấu cách, nên ta dùng gets(a)

int main(){

char a[100];

gets(a);

printf("%s", a);

}

- nếu trước câu lệnh gets(a) có phím enter thì lệnh gets sẽ không hoạt động.

int main(){

char a[100];

int x; scanf("%d", &x); //bấm enter ở đây.

gets(a); //không chạy do đằng trước có phím enter.

printf("%s", a);

}

output: số x vừa scanf (ko in ra a được do lệnh gets k chạy)

- cách giải quyết: thêm getchar() ngay trước lệnh gets(a) để nó ăn mất 1 phát gõ (cụ thể là phát bấm enter)

- lưu ý: chỉ dùng getchar() nếu trước gets(a) có phím enter, nếu không lệnh gets(a) hoặc lệnh scanf(“%s”) sẽ bị getchar() ăn mất kí tự đầu tiên.

int main(){

char a[100];

int x; scanf("%d", &x);

bấm enter;

getchar(); //ăn phím enter ta vừa bấm

gets(a);

printf("%s", a);

}

- có thể thay thế getchar() bằng scanf("\n"); lệnh này sẽ ăn toàn bộ phím enter trước nó.

int main(){

char a[100];

int x; scanf("%d", &x); bấm enter

scanf("\n"); ăn toàn bộ phím enter trước nó.

gets(a);

printf("%s", a);

}

- lưu ý: lệnh gets(a) cũng ăn luôn phím enter, nên nếu ngay sau gets(a) là lệnh gets(b) thì ta không cần getchar() hoặc scanf(“\n”);

int main(){

char a[100], b[100];

gets(a);

gets(b);

printf("%s %s", a, b);

}

- dùng fgets để lấy đúng số lượng kí tự mình muốn(vẫn cần ăn phím enter ở trước giống gets)

int main(){

char a[10];

fgets(a, 10, stdin); lấy tối đa 10 pt cho xâu a

printf("%s", a);

}

- chú ý: hàm fgets sẽ coi phím enter là phần tử cuối cùng của xâu kí tự vừa nhập

int main(){

char a[10];

fgets(a, 9, stdin); gõ nguyen, xong bấm enter

printf("%d\n", strlen(a)); nguyen chỉ có 6 kí tự, nhưng strlen(a) = 7 do fgets đã đọc cả phím enter

printf("%s", a);

}

- cách giải quyết: biến a[strlen(a) - 1] (kí tự cuối cùng, là vị trí của phím enter) thành null.

int main(){

char a[10];

fgets(a, 9, stdin);

a[strlen(a) - 1] = '\0';

printf("%d\n", strlen(a));

printf("%s", a);

}

- nếu đặt a[6](phần tử thứ 7 trong mảng) bằng null, mảng sẽ bị ngắt từ chỗ đó, tạo thành một mảng chỉ có 6 phần tử.(chạy từ a[0] đến a[5])

(null không được coi là một phần tử)

int main(){

char a[10];

fgets(a, 9, stdin);

a[6] = '\0';

printf("%d\n", strlen(a));

printf("%s", a);

}

- Có 4 kiểu trả về:

+ Kiểu 1: test bằng is: trả về 0 hoặc 1

+ Kiểu 2: tolower và toupper: trả về một char, không thay đổi giá trị của biến c nhập vào

+ Kiểu 3: atoi và atoll, trả về một int, không thay đổi giá trị của mảng c nhập vào.

+ Kiểu 4: có dấu \*, bọn này sẽ thay đổi giá trị của mảng c nhập vào.

- Các hàm thường dùng:

int islower(char c); các hàm này trả về 0 hoặc 1;

int isupper(char c);

int isalpha(char c);

int isdigit(char c);

int strlen(char c[]); số phần tử của xâu c.

- char tolower(char c); (2 hàm này không thay đổi kí tự c nhập vào)

char toupper(char c);

-int strcmp(char c[], char d[]); so sánh 2 xâu c và d

c>d: 1

c = d: 0

c < d: -1

- Các lệnh sau dùng ngoài printf(“%s”, c) hay trong printf đều được. Chúng thay đổi giá trị của xâu c.

char\* strrev(char c[]): đảo ngược xâu c

char\* strlwr(char c[]): viết thường xâu c

char\* strupr(char c[]): viết hoa xâu c

char\* strcpy(char c[], char d[]); xóa các phần tử của c và copy thành d

char\* strcat(char c[], char d[]); nối xâu d vào đít xâu c

VD: int main(){

char c[10] = "alo ";

char d[10] = "nguyen t";

strupr(c);

printf("%s", c);

}

hoặc

int main(){

char c[10] = "alo ";

char d[10] = "nguyen t";

printf("%s", strupr(c));

}

-char\* strstr(char a[], char b[]); kiểm tra xâu b có phải xâu con của a hay không; Nếu xâu b không phải là con của xâu a sẽ trả về NULL. Nếu là con sẽ trả về kí tự khác NULL.

VD:

int main(){

char a[1001], b[1001];

gets(a);

gets(b);

if(strstr(a, b) == NULL){

printf("NO");

}

else printf("YES");

}

- cách dùng atoi(chuyển xâu thành int) và atoll(chuyển xâu thành long long)

int atoi(char a[]); output là 1 số int, hàm k thay đổi xâu.

long long atoll(char a[]) giống atoi nhưng output là lld

int main(){

char c[10] = "12344";

atoi(c);

printf("%d", c); như lày không được do lệnh atoi chỉ trả về 1 số int chứ không đụng đến mảng c;

}

///

int main(){

char c[10] = "12344";

printf("%d", atoi(c)); như lày mới được. atoi trả về int. (atoll thì phải printf kiểu %lld)

}

- char a[10] = “nguyen” là một mảng có 6 kí tự, có strlen = 6, các phần tử chạy từ a[0] đến a[5], nhưng có bộ nhớ tối đa là 10 phần tử.

int main(){

char a[100] = "nguyen anh tuan"; cách khai báo chuỗi kí tự

char b[100] = {'a', 'b', 'c', 'd'};

printf("%s %s", a, b); cách in chuỗi kí tự a, b

printf("%d", strlen(a)); cách tìm độ dài chuỗi kí tự a

}

}

- Không in được a[5] ở dạng %s, nhưng có thể in được a + 5 dưới dạng %s.

int main(){

char a[] = "nguyen anh tuan";

printf("%s", a + 5); // in toàn bộ những phần tử từ a[5] đến đít.

}

output: n anh tuan

- có thể in được a[5] dưới dạng %c, nhưng không in được

a + 5 dưới dạng %c.

int main(){

char a[] = "nguyen anh tuan";

printf("%c", a[5]);

}

- muốn in a + 5 dưới dạng %c thì phải có \* để giải tham chiếu.

int main(){

char a[] = "nguyen anh tuan";

printf("%c", \*(a + 5));

}

- đối với mảng int, khi in %d a + 1 sẽ in ra ô nhớ của a[1]

nhưng khi in %d \*(a + 1) sẽ in ra phần tử a[1].

int main(){

int a[5] = {1, 2, 3, 4, 5};

printf("%d", \*(a + 1));

}

- Đối với phần tử a[i][j], con trỏ quản lí nó là (\*(a+i)+j)

giá trị của a[i][j] là \*(\*(a+i)+j)

.

- cấp phát động trong C

int main(){

int \*ptr = (int\*)malloc(1000000 \* sizeof(int)); //cap phat 4m byte, tạo 1 mảng tên ptr có 1000000 pt

int n = 5;

for(int i = 0; i < n; i++){

scanf("%d", &ptr[i]);

}

for(int i = 0; i < n; i++){

printf("%d ", ptr[i]);

}

free(ptr);

}

- Trong 3 hàm sau: hàm đầu tiên trả về 1 kí tự nên dùng kiểu dữ liệu char

hàm thứ 2 trả về số nên dùng kiểu dữ liệu int

hàm thứ 3 trả về xâu kí tự c nên dùng kiểu dữ liệu char\*

VD1: char to\_upper(char c){ //in thuong thanh in hoa

if(c >= 'a' && c <= 'z'){

c -= 32;

}

return c;

}

VD2: int strlen(char c[]){

int cnt = 0;

while(c[cnt] != '\0'){

cnt++;

}

return cnt;

}

VD3: char\* strlwr(char c[]){

for(int i = 0; i < strlen(c); i++){

if(c[i] >= 'A' && c[i] <= 'Z'){

c[i] += 32;

}

}

return c;

}

- cách chuyển từ kí tự chữ số về số.

so = c[i] – ‘0’;

- cách in ra kí tự a-z từ mã ascii.

a: 97

z: 122

=> tổng có 26 chữ cái, ta dùng một mảng cnt[26] có 26 phần tử để đếm số lần xuất hiện của từng chữ cái.

int main(){

char c[1001];

gets(c);

int cnt[26] = {0};

for(int i = 0; i < strlen(c); i++){

cnt[c[i] - 'a']++;

}

for(int i = 0; i < 26; i++){

if(cnt[i])

printf("%c %d\n", (i + 'a'), cnt[i]);

}

}

- ép kiểu từ mã ascii thành char.

char m;

int i;

m = (char)i;

- tách từ bằng strtok(c, “dlim”); câu lệnh này sẽ tách ra một token bắt đầu từ c[0] cho đến dlim.

strtok(NULL, “dlim”): câu lệnh này sẽ tách ra một token chạy từ NULL đến dlim. Nếu mảng c đã hết từ thì hàm này sẽ trả về NULL.

PP: tạo một con trỏ tên token trỏ vào token vừa tách

int main(){

char c[1001];

gets(c);

char \*token = strtok(c, " "); câu lệnh này sẽ tạo ra một con trỏ token trỏ về từ đầu tiên (nguyen), đồng thời hàm c sẽ bị cắt ra chỉ còn từ đầu tiên đó(nguyen), và một hàm mới tên NULL được tạo ra bao gồm phần còn lại của xâu c (anh tuan).

while(token != NULL){

printf("%s\n", token);

token = strtok(NULL, " ");

}

}

- Cách sử dụng hàm qsort;

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

int tcs(int n){

int sum = 0;

while(n != 0){

sum += (n%10);

n /= 10;

}

return sum;

}

int cmp(const void \*a, const void \*b){

// -1 ben trai, 1 ben phai

//khi muon a dung truoc thi return -1 (mot gia tri am)

//khi muon b dung truoc thi return 1 (mot gia tri duong)

int \*x = (int\*)a;

int \*y = (int\*)b;

if(\*x%2==0 && \*y%2==1) return -1;

if(\*x%2==1 && \*y%2==0) return 1;

if(\*x%2==0 && \*y%2==0) return \*y - \*x;

return \*x - \*y;

}

int main(){

int n; scanf("%d", &n);

int a[n];

for(int i = 0; i < n; i++){

scanf("%d", &a[i]);

}

qsort(a, n, sizeof(int), cmp);

for(int i = 0; i < n; i++){

printf("%d ", a[i]);

}

}