-chú ý: parament: tham số: truyền thẳng số vào hàm khi code. vd: max(100, 200)

arguments: đối số(tham số thực sự): ko truyền số vào hàm khi code mà truyền độc lập mỗi lần run.

- Nếu khai báo mảng m[] trong hàm thì mảng m sẽ không dùng ở bên ngoài hàm được

Giải pháp: khai báo mảng m trước khi gọi hàm, rồi nhập mảng m vào hàm.

\*xay dung ham:

kiểu trả về + tên hàm(danh sách tham số) {

câu lệnh;(2)

return + kết quả;(nếu kiểu trả về là int, lld,...

}

int main() {

tên hàm;

}

\* lưu ý: kiểu trả về là kiểu dữ liệu gì thì khi khai báo biến trong (2) phải dùng kiểu dữ liệu đó, vd: int, long long,...

- hàm void không có danh sách tham số, chỉ đơn giản là thực hiện tất cả câu lệnh có trong hàm

không cần return. Có thể code như bình thường rồi ném vào hàm void.

void xinchao() {

printf("xin chao\n");

}

int main() {

xinchao();

}

- nếu kiểu dữ liệu là int, có tham số n trong hàm () thì cần return về biến thể hiện đáp án của hàm, như dưới đây sẽ là return sum;

int tong(int n){

int sum = 0;

for(int i = 1; i <= n; i++){

sum += i;

}

return sum;

}

int main(){

int x = 50;

int kq = tong(x);

printf("%d", kq);

return 0;

}

(hàm int sau khi nhập tham số or đối số vào sẽ giống như 1 hằng số)

-chú ý: return phải nằm trong {}, ngay sau hàm có giá trị cần trả về; return có chức năng như break khi nằm trong hàm for.

- danh sách tham số gồm 2 biến (có 2 cách trả về là return a và return b)

(ko khai báo kiểu int a, b mà phải dùng int a, int b)

int max(int a, int b){

if(a > b) return a;

else return b;

}

int main(){

printf("%d", max(100, 200));

return 0;

}

- hàm sẽ không làm thay đổi giá trị của đối số nhập vào.

vd: cho gọi hàm với m = 5, sau khi gọi xong, m vẫn bằng 5.

Quá trình gọi chỉ đơn giản là quá trình ném giá trị (pass by value) của m vào chỗ thằng tham số n được khai báo khi code hàm ở trên.

void change(int n){

n += 10;

}

int main() {

int m = 5;

change(m);

printf("%d", m);

}

- void lồng void:

int tong(int n){

return n \* (n + 1) / 2;

}

void xinchao() {

printf("xin chao\n");

printf("%d", tong(100));

}

int main() {

xinchao(); }

- khi gọi hàm a trong hàm b, hàm a phải nằm ở trước hàm b. Nếu không máy sẽ báo hàm a chưa được khai báo.

void xinchao() {

printf("xin chao\n");

printf("%d", tong(100));

}

int tong(int n){

return n \* (n + 1) / 2;

}

int main() {

xinchao();

}

Giải pháp: sử dụng “Khai báo nguyên mẫu hàm” hay function propotype hay function declaration( thêm tên của tất cả các hàm lên trên cùng)

void xinchao();

int tong(int n);

void xinchao() {

printf("xin chao\n");

printf("%d", tong(100));

}

int tong(int n){

return n \* (n + 1) / 2;

}

int main() {

xinchao(); }

- kiểm tra 1 số có phải chính phương hay ko

\*lưu ý: khi kiểm tra tính đúng sai, nên dùng return 1 cho mệnh đề đúng, return 0 cho mệnh đề sai

trong hàm main thì kiểm tra như ví dụ sau:

int chinhphuong2(int n){

int can = sqrt(n);

if(can \* can == n) return 1;

else return 0;

}

int main() {

int n = 16;

if(chinhphuong2(n) == 1) printf("yes");

else printf("no");

return 0;

}

- liệt kê các số từ 1 đến n sao cho số đó có chứa chữ số chia hết cho 6, và tổng các chữ số chia hết cho 8

\* lưu ý: sử dụng 2 hàm int

hàm check1 là hàm while else mình kiếm bấy lâu nay

int check1(int n){

while(n){

if(n % 6 == 0) return 1;

n /= 10;

}

return 0;

}

int check2(int n){

int sum = 0;

while(n){

sum += n % 10;

n /= 10;

}

return sum % 8 == 0;

}

int main() {

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

for(int i = 1; i <= n; i++){

if(check1(i) == 1 && check2(i) == 1){

printf("%d ", i);

}

}

return 0;

}