**new\_words**

|  |  |
| --- | --- |
| **Từ** | **Nghĩa** |
| occurrence | sự xuất hiện |
| duplicated | được lặp lại |
| normalize | bình thường hóa ( thường là khiến cho các từ cách nhau bằng một dấu cách duy nhất) |
| prime | số nguyên tố |
| even | số chẵn |
| odd | số lẻ |
| exponent | số mũ |
| power of n | n^x |
| representation of | biểu diễn dưới dạng |
| average | trung bình cộng |
| square | bình phương |
| palindrome | xuôi ngược đều giống nhau |
| synmetric | đối xứng |
| factorial | giai thừa |
| uppercase | in hoa |
| lowercase | viết thường |
| reversed | đảo ngược |
| adjacent | liền kề |
| frequency | tần suất |
| product | tích |
| forming | hình thành |
| obeys | tuân theo |
| divisible | chia hết |
| toggle | chuyển đổi |
| separated | phân cách |
| equivalent | tương đương |

**stdio.h**

**getchar()**: đọc một kí tự được nhập vào bàn phím.

    char ch;

    printf("Nhập một ký tự: ");

    ch = getchar();

    //Nhập từ bàn phím 1 kí tự và nó sẽ được lưu vào biến ch.

**gets()**: nhập chuỗi từ bàn phím.

    char name[50];

    gets(name);

    //Lưu ý: xài gets có thể gây tràn bộ nhớ vì nó ko kiểm tra độ dài của chuỗi.

**fgets()**: nhập chuỗi từ bàn phím hoặc một luồng dữ liệu.

    char a[101];

    //fgets sẽ để lại một kí tự NULL('\n'), ở cuối chuỗi vì vậy nó chỉ đọc nhập được 100 kí tự còn 1 kí tự là cho '\n'.

    // Vì vậy nếu muốn nhập n kí tự, ta phải khai báo n+1 kí tự.

    fgets (a, sizeof(a), stdin);

**%#**: in mã hexa hoặc octal có bao gồm tiền tố 0x

    int number = 10;

    printf ("%#x", number);

    //Lưu ý: Tùy theo x viết thường hay X viết hoa thì mã hexa sẽ in thường hoặc in hoa theo nó.

    //OUTPUT: 0xa

**string.h**

**strlen()**: Trả về độ dài của chuỗi.

    size\_t length = strlen("Help");

    // Output: 4

**strcpy()**: Sao chép từ chuỗi này sang chuỗi khác.

    char a[6];

    strcpy(a, "Stupid");

    printf("%s", a);

    // Output: Stupid

**strstr()**: Kiểm tra xem chuỗi con có xuất hiện trong chuỗi lớn hơn không.

    char a[3] = "ngu";

    char b[9] = "nguđầnngu";

    strstr(a, b);

    // strstr sẽ trả về giá trị NULL nếu có xuất hiện chuỗi a trong b.

**strcat()**: thêm một chuối vào cuối chuỗi khác.

    char a[50] = "Vờ ";

    char b[] = "cờ lờ!";

    strcat(a, b);

    printf("%s\n", a); // In ra "Vờ cờ lờ!"

**stdlib.h**

**malloc()**: được sử dụng để cấp phát bộ nhớ động (heap memory).

    int \*ptr = (int \*)malloc(sizeof(int));

    \*ptr = 10;

    free(ptr);

**calloc()**: cấp phát và khởi tạo bộ nhớ động với tất cả các phần tử được thiết lập thành 0.

    int \*ptr = (int \*)calloc(5, sizeof(int));

    free(ptr);

**realloc()**: được sử dụng để thay đổi kích thước của bộ nhớ đã cấp phát trước đó.

    int \*ptr = (int \*)malloc(5 \* sizeof(int));

    ptr = (int \*)realloc(ptr, 10 \* sizeof(int));

    free(ptr);

**free()**: được sử dụng để giải phóng bộ nhớ đã cấp phát trước đó.

    int \*ptr = (int \*)malloc(sizeof(int));

    \*ptr = 10;

    free(ptr);

**math.h**

**abs()**: Hàm tính giá trị tuyệt đối của một số nguyên.

    int x = -5;

    int absolute = abs(x);

    printf("%d", absolute);

    //OUTPUT: 5

**sqrt()**: Hàm tính căn bậc hai của một số thực.

    double number = 16.0;

    double squareRoot = sqrt(number);

    printf("%d",squareRoot);

    //OUTPUT: 4

**pow()**: Hàm tính lũy thừa của một số thực.

    double base = 2.0;

    double exponent = 3.0;

    double result = pow(base, exponent);

    printf("%.1lf" result);

    //OUTPUT: 8.0

**floor()**: Hàm làm tròn số thực xuống (lấy phần nguyên).

    double number = 4.9;

    double roundedDown = floor(number);

    printf("%.1lf"roundedDown);

    //OUTPUT: 4.0

**ceil()**: Hàm làm tròn số thực lên (lấy phần nguyên gần nhất lớn hơn).

    double number = 4.1;

    double roundedUp = ceil(number);

    printf("%.1lf"roundedUp);

    //OUTPUT: 5.0

**fabs()**: Hàm tính giá trị tuyệt đối của một số thực.

    double x = -7.3;

    double absolute = fabs(x);

    printf("%.1lf" absolute);

    //OUTPUT: 7.3