 Algoritma

Algoritma, bir sorunu çözmek için gerçekleştirilmesi gereken adımların bir dizisidir.

 Algoritma

Mimarisi Algoritma genellikle şu bölümlerden oluşur: Giriş: Algoritmanın başlatılması için kullanılan veriler. İşleme: Sorunu çözmek için gerçekleştirilen eylemler. Çıkış: Algoritmanın sonuçları.

 Algoritma

Gereksinimleri Algoritma aşağıdaki gereksinimlere uygun olmalıdır: Tamlık: Algoritma, sorunu çözmek için gerekli olan tüm eylemleri gerçekleştirmelidir. Doğruluk: Algoritma doğru sonuçlar vermelidir. Optimalite: Algoritma, sorunu çözmek için minimum kaynakları kullanmalıdır.

 Değişkenler

Değişken, verileri depolamak için kullanılan bellek alanıdır. Değişkenler, algoritmanın ara sonuçlarını saklamak için kullanılır.

 Döngüler

Döngü, belirli bir koşul sağlandığı sürece tekrarlanan adımların bir dizisidir. Döngüler, algoritmada tekrarlanan eylemleri gerçekleştirmek için kullanılır.

örnekler

1.İlk n doğal sayısının toplamını hesaplama.

2.Doğal sayı n'ın faktöriyelini hesaplama.

3.Mantıksal ifade için doğruluk tablosunu yazdırma.

1)

#include <stdio.h>

int main() {

int n, sum = 0;

printf("Введите значение n: ");

scanf("%d", &n);

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

sum += i;

}

printf("İlk %d doğal sayısının toplamı: %d\n",n, sum);

return 0; }

2)

#include <stdio.h>

int main() {

int n, factorial = 1;

printf("n: ");

scanf("%d", &n);

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

factorial \*= i;

}

printf("factorial %d: %d\n", n, factorial);

return 0;

}

3)

#include <stdio.h>

int main() {

printf("Mantıksal İfade İçin Doğruluk Tablosu A AND B:\n");

printf("A | B | A AND B\n");

printf("--|---|--------\n");

for (int A = 0; A <= 1; ++A) {

for (int B = 0; B <= 1; ++B) {

printf("%d | %d | %d\n", A, B, A && B);

}

}

return 0;

}