8.o. Algoritmusok fogalmak és meghatározásuk

1. **Algoritmus** (Algorithm): Egy lépésekből álló, a problémamegoldásra szolgáló eljárás, amely világosan meghatározza, hogyan kell egy feladatot elvégezni.
2. **Folyamatábra** (Flowchart): Az algoritmusok grafikus ábrázolása, amely különböző geometriai formák segítségével mutatja be az egyes lépéseket.
3. **Változó** (Variable): Olyan tárolóhely, amely értéket tartalmazhat, és amelyet programozás során változtathatunk.
4. **Ciklus** (Loop): Olyan vezérlési szerkezet, amely egy sor utasítást ismétel, amíg egy feltétel teljesül.
5. **Számlálós ciklus** (Counting loop): Olyan ciklus, amelynek az ismétlődése egy előre meghatározott számú alkalommal történik.
6. **Feltételes ciklus** (Conditional loop): Olyan ciklus, amely addig ismétlődik, amíg egy adott feltétel igaz.
7. **Végtelen ciklus** (Infinite loop): Olyan ciklus, amely soha nem ér véget, hacsak nem állítjuk le valamilyen módon.
8. **Elágazás** (Branching): Az algoritmus azon része, ahol a program különböző utasítások végrehajtására választhat a feltételtől függően.
9. **Egyszerű elágazás** (Simple branching): Egy olyan elágazás, ahol egy feltétel igazságától függően csak egy lehetőség van végrehajtva.
10. **Kétirányú elágazás** (Two-way branching): Egy elágazás, ahol két különböző lehetőség közül választunk a feltételtől függően.
11. **Logikai művelet** (Logical operation): Olyan műveletek, amelyek több feltételt összekapcsolnak, például ÉS, VAGY, NEM.
12. **Összehasonlító művelet** (Comparison operation): Olyan művelet, amely két értéket hasonlít össze, például egyenlőség vagy nagyobb mint.
13. **Változó értéke** (Variable value): Az a konkrét adat, amelyet egy változó tárol.
14. **Függvény** (Function): Olyan programrész, amely egy adott feladatot végez, és visszaad egy eredményt.
15. **Alprogram** (Subroutine): Egy önállóan hívható programrészlet, amely egyes feladatokat lát el.
16. **Paraméter** (Parameter): A függvényekhez átadott bemeneti adat, amely lehetővé teszi, hogy a függvény más-más adatokat dolgozzon fel.
17. **Visszatérési érték** (Return value): A függvény által végzett számítás vagy művelet eredménye, amelyet visszaad a függvény hívójának.
18. **Szintaxis** (Syntax): Az a szabályrendszer, amely meghatározza, hogy egy programnyelvben hogyan kell helyesen megírni az utasításokat.
19. **Hibaüzenet** (Error message): A program által generált üzenet, amely tájékoztat minket, ha a program végrehajtása során valami nem működik megfelelően.
20. **Függvényhívás** (Function call): Az a művelet, amikor egy programban meghívunk egy függvényt, hogy végrehajtsa annak feladatát.
21. **Változó deklarálás** (Variable declaration): A változó nevének és típusának meghatározása a programban, hogy azt később használni tudjuk.
22. **Adattípus** (Data type): A változók által tárolt adatok típusa, mint például számok, karakterek vagy logikai értékek.

23. **Logikai érték** (Logical value): Olyan érték, amely csak két lehetséges állapotot vehet fel: igaz vagy hamis.
24. **Végtelen ciklus** (Infinite loop): Egy olyan ciklus, amely nem ér véget, hacsak valamilyen külső esemény nem állítja meg a végrehajtást.
25. **Függvény paramétere** (Function parameter): Az a bemeneti adat, amelyet egy függvény végrehajtása során használunk.
26. **Visszatérési érték** (Return value): Az érték, amit a függvény visszaad, miután elvégezte a rá bízott műveleteket.
27. **Sorrend** (Order): A programban lévő utasítások végrehajtási sorrendje.
28. **Töréspont** (Breakpoint): Az a hely a programban, ahol a végrehajtás megállítható, hogy részletesen vizsgálhassuk a program állapotát.
29. **Ciklusváltozó** (Loop variable): Az a változó, amely a ciklusok számát vagy iterációit követi.
30. **Feltételes utasítás** (Conditional statement): Az utasítás, amely alapján a program két vagy több különböző módon hajtódik végre a feltétel igazságától függően.
31. **Paraméter** (Parameter): A függvény bemeneti értéke, amelyet a függvény végrehajtása során használunk.
32. **Változó típus** (Variable type): A változó által tárolt adat típusa, például egész szám, valós szám, vagy karakterlánc.
33. **Programozási nyelv** (Programming language): A számítógép számára készült utasítások nyelve, amely lehetővé teszi a számítógép számára a feladatok végrehajtását.
34. **Függvényhívás** (Function call): Az a művelet, amikor egy programban meghívunk egy függvényt.
35. **Típusalgoritmus** (Type algorithm): Olyan algoritmus, amely adott típusú problémák megoldására szolgál.
36. **Műveleti sorrend** (Operation order): Az a szabály, hogy a műveletek milyen sorrendben hajtódnak végre a programban.
37. **Szemantika** (Semantics): A programnyelv azon aspektusa, amely meghatározza, hogy a szintaktikailag helyes utasítások mit jelenthetnek a programban.
38. **Visszatérési érték** (Return value): A függvény eredménye, amelyet visszaküld a hívó programnak.
39. **Deklarálás** (Declaration): Az a folyamat, amikor a változó nevét és típusát először meghatározzuk a programban.
40. **Hiba elhárítása** (Error handling): A folyamat, amely a program futása közben felmerülő hibák kezelésére szolgál.
41. **Változó értéke** (Variable value): Az adott változóhoz rendelt adat.
42. **Vezérlési szerkezet** (Control structure): A program végrehajtásának irányítását biztosító szerkezetek, mint a ciklusok és elágazások.
43. **Végtelen ciklus** (Endless loop): Olyan ciklus, amely soha nem ér véget, hacsak valamilyen külső tényező nem állítja meg.
44. **Műveleti kifejezés** (Operational expression): Olyan kifejezés, amely különböző műveletek eredményét adja vissza.
45. **Deklarációs nyelv** (Declarative language): Olyan programozási nyelv, amely a kívánt eredmény meghatározására összpontosít.

46. **Összefűzés** (Concatenation): Különböző karakterláncok összekapcsolása egy új karakterlánc létrehozásával.
47. **Típusellenőrzés** (Type checking): Az a folyamat, amely biztosítja, hogy a változókat a megfelelő típusú adatokkal használjuk.
48. **Matematikai művelet** (Mathematical operation): Olyan műveletek, amelyek számértékekkel dolgoznak, például összeadás, kivonás.
49. **Forráskód** (Source code): A program szövege, amelyet a programozó ír.
50. **Eredmény** (Result): Az algoritmus vagy művelet kimenete.
51. **Adatfeldolgozás** (Data processing): A beérkezett adatok elemzése és manipulálása.
52. **Bemeneti adat** (Input data): Az adatok, amelyeket a program vagy algoritmus feldolgoz.
53. **Kimeneti adat** (Output data): Az eredmények, amelyeket a program ad vissza a feldolgozott bemeneti adatok alapján.
54. **Értékadás** (Assignment): Az a művelet, amely során egy változóhoz értéket rendelünk.
55. **Bemenet kérése** (Input request): A folyamat, amelyben adatokat kérünk a felhasználótól.
56. **Hiba elhárítás** (Debugging): A program hibáinak keresése és javítása.
57. **Feltétel vizsgálata** (Condition testing): A folyamat, amelyben megvizsgáljuk, hogy egy adott feltétel igaz vagy hamis.
58. **Funkcionális programozás** (Functional programming): Olyan programozási paradigma, amely a függvényekre összpontosít.
59. **Deklaráció** (Declaration): A változó vagy függvény létrehozása a programban.
60. **Algoritmus tervezés** (Algorithm design): Az a folyamat, amely során meghatározzuk, hogyan oldjunk meg egy problémát lépésekben.
61. **Bemenet** (Input): Az adat, amit a programnak átadunk a feldolgozáshoz.
62. **Kimenet** (Output): Az eredmény, amit a program ad vissza a feldolgozott bemenet alapján.
63. **Műveleti sorrend** (Order of operations): A műveletek végrehajtásának meghatározott sorrendje, amelyet a szintaxis határoz meg.
64. **Függvény paramétere** (Function parameter): Az a bemeneti adat, amit egy függvény hívásakor átadunk neki.
65. **Visszatérési érték** (Return value): Az a szám vagy adat, amit egy függvény visszaad miután elvégezte a feladatát.
66. **Eljárás** (Procedure): Olyan programrészlet, amely nem ad vissza értéket, de egy adott feladatot elvégez.
67. **Bemeneti változó** (Input variable): Az a változó, amely adatokat tárol, amit a program végrehajtása során felhasználunk.
68. **Függvényhívás** (Function call): Az a művelet, amikor meghívunk egy függvényt, hogy elvégezze a feladatát.
69. **Programozási paradigmák** (Programming paradigms): Különböző megközelítések a programok megírására, mint például az objektumorientált vagy funkcionális programozás.
70. **Függvény visszatérési típus** (Function return type): A típus, amelyet a függvény visszaad, például egész szám vagy szöveg.
71. **Nyelvi struktúra** (Language structure): A programozási nyelv szerkezete, amely meghatározza, hogyan kell felépíteni a program utasításait.
72. **Ciklus vezérlő változó** (Loop control variable): Az a változó, amely a ciklus működését szabályozza, például egy számláló.

73. **Összefűzés** (Concatenation): Különböző szöveges adatokat egyesítünk egy új szöveggé.
74. **Konstrukció** (Construct): A programozásban egy adott feladatot végrehajtó utasítások kombinációja.
75. **Bemeneti paraméter** (Input parameter): Olyan paraméter, amely az adatokat tartalmazza, amiket egy programnak átadunk.
76. **Típusellenőrzés** (Type checking): A programozási nyelvekben végzett művelet, amely biztosítja, hogy a változókat a megfelelő típusú adatokkal használjuk.
77. **Programozási utasítás** (Programming statement): Olyan kifejezés, amely egy adott műveletet végez a programban.
78. **Szemantikai hiba** (Semantic error): Olyan hiba, amely akkor fordul elő, ha a program szintaxisa helyes, de a logikai működése nem megfelelő.
79. **Változó típusok** (Variable types): Az adat típusai, amelyek lehetnek egész számok, karakterláncok, logikai értékek stb.
80. **Moduláris programozás** (Modular programming): Olyan programozási technika, ahol a programot kisebb részekre (modulokra) bontjuk.
81. **Függvények és eljárások** (Functions and procedures): A programozás alapvető építőelemei, amelyek egy adott feladatot végeznek el.
82. **Program strukturálás** (Program structuring): A program logikai felépítése, amely segíti a kód olvashatóságát és karbantartását.
83. **Feltételes művelet** (Conditional operation): Olyan művelet, amely csak akkor hajtódik végre, ha egy adott feltétel teljesül.
84. **Programozási nyelv** (Programming language): A számítógép számára készült utasítások nyelve, amely lehetővé teszi a feladatok végrehajtását.
85. **Interaktív programozás** (Interactive programming): A programok írása és tesztelése közvetlenül a felhasználóval való interakció során.
86. **Cikluskezelő utasítás** (Loop control statement): Az a parancs, amely vezérli, hogy a ciklus mikor induljon el, mikor álljon le, vagy hogyan lépjen át.
87. **Verziókövetés** (Version control): Olyan eszközök, amelyek segítségével követhetjük a program módosításait és verzióit.
88. **Deklarálás és inicializálás** (Declaration and initialization): A változó típusának meghatározása és kezdő értékének hozzárendelése.
89. **Logikai kifejezés** (Logical expression): Olyan kifejezés, amely logikai értéket ad vissza, mint igaz vagy hamis.
90. **Algoritmus tervezés** (Algorithm design): Az a folyamat, amely során meghatározzuk, hogyan oldjunk meg egy problémát lépésekben.
91. **Törés** (Break): A ciklusból való kilépés utasítása.
92. **Funkcionális programozás** (Functional programming): Programozási paradigmák, amelyek a függvények és azok alkalmazására építenek.
93. **Állapotgép** (State machine): Olyan programozási megoldás, amely különböző állapotok között váltogat, és minden állapotnak más-más viselkedése van.
94. **Függvényhívás** (Function invocation): Olyan folyamat, amely során egy már létező függvényt hívunk meg, hogy elvégezzen egy feladatot.
95. **Program hiba** (Program bug): Olyan hiba, amely a program működésében nem kívánt viselkedést okoz.

96. **Hibakezelés** (Error handling): A folyamat, amely segít kezelni a program futtatása közben felmerülő hibákat.
97. **Változók életciklusa** (Variable lifecycle): A változó létezésének időtartama a programban, azaz a deklarálástól a törléséig.
98. **Bemeneti adat feldolgozás** (Input data processing): Az adatok begyűjtése és azok feldolgozása egy algoritmus által.
99. **Szöveges adat** (Text data): Olyan adatok, amelyek karakterekből állnak.
100. **Numerikus adat** (Numeric data): Olyan adatok, amelyek számokból állnak.
101. **Szintaktikai hiba** (Syntax error): Olyan hiba, amely akkor fordul elő, amikor a program nem felel meg a programozási nyelv szintaktikai szabályainak.
102. **Tárolóhely** (Storage location): A memóriában található hely, ahol adatokat tárolunk.
103. **Globális változó** (Global variable): Olyan változó, amely az egész programban elérhető és módosítható.
104. **Lokális változó** (Local variable): Olyan változó, amely csak egy adott függvényen belül érvényes és használható.
105. **Függvény tesztelés** (Function testing): A függvények ellenőrzése, hogy biztosítsuk, helyesen működnek-e a várt módon.
106. **Kódrefaktorálás** (Code refactoring): A meglévő kód átalakítása annak érdekében, hogy tisztább, olvashatóbb és hatékonyabb legyen anélkül, hogy megváltoztatnánk annak működését.
107. **Funkcionális típus** (Functional type): A függvények és azok visszatérési értékei alapján meghatározott típusok, amelyek a függvényekkel kapcsolatos műveleteket segítenek.
108. **Program futtatása** (Program execution): A program elindítása és végrehajtása a számítógépen.
109. **Memória** (Memory): A számítógép azon része, amely ideiglenesen tárolja az adatokat és utasításokat a program futása alatt.
110. **Véletlenszám-generálás** (Random number generation): A számítógépes programokban használt eljárás, amely véletlenszerű számokat generál.
111. **Újrahasználhatóság** (Reusability): A kód azon képessége, hogy más programokban vagy programrészletekben újra felhasználható legyen.
112. **Öröklődés** (Inheritance): Az objektumorientált programozásban egy osztály képes örökölni egy másik osztály jellemzőit és működését.
113. **Polimorfizmus** (Polymorphism): Az objektumorientált programozás olyan fogalma, amely lehetővé teszi, hogy egy függvény többféle formában működjön.
114. **Absztrakció** (Abstraction): Az a folyamat, amely során a lényeges információkat kiemeljük, miközben elrejtjük az összetett részleteket.
115. **Funkciósor** (Function chain): Több függvény egymás után történő hívása, ahol az egyik függvény kimenete a következő bemenete.
116. **Hibaüzenet log** (Error log): A program által generált naplófájl, amely tartalmazza a hiba részleteit és annak okait.
117. **Aszinkron programozás** (Asynchronous programming): Olyan programozási technika, ahol a műveletek párhuzamosan, nem egymást követve hajtódnak végre.
118. **Szálkezelés** (Thread management): A párhuzamos végrehajtás kezelése, amely lehetővé teszi, hogy egy program több szálon dolgozzon egyszerre.
119. **Debugger** (Debugger): Eszköz, amely segít a program hibáinak keresésében és azok javításában.

120.**Kódolvasás** (Code reading): A program kódjának átvizsgálása annak megértésére és hibák keresésére.