Изпитна тема № 1: Изчисления, линейни, разклонени и циклични алгоритми

1.

1.1 Дефинира понятията: програмиране, език за програмиране, среда за разработка (IDE), компилация и интерпретация. (10 т.)

- *Програмиране* - може да се определи като „казване на компютъра какво да прави“. Това става като програмиста предостави инструкции на устройството, написани на избран от него език за програмиране.
- Език за програмиране – средство за комуникация с компютърната система. Използват се за създаването на програми, които контролилат поведението на машината, реализират алгоритми точно във вид на човешка комуникация. Разделят се на езици от ниско (Assembly и машинен код), средно (C) и високо ниво (C# и Java).
- *Среда за разработка* – IDE (Integrated Development Environment) е софтуерно приложение, което предоставя цялостна среда за разработване на софтуер. Обикновено се състои от редактор на код, инструменти за автоматизиране на построяване на приложението и дебъгер. Примери:
  - Visual Studio
  - InteliJ
  - Arduino IDE
  - Android Studio
- *Компилация* – превод на код от високо ниво в код от ниско ниво преди изпълнение, за да бъде изпълнен директно от компютъра. Компилаторни езици са: C++, C#, Java и др.
- *Интерпретация* – превежда написания код ред по ред по време на изпълнение. Интерпретаторни езици са: JavaScript, Python и PHP.

1.2 Демонстрира употребата на основни функционалности на среда за разработка (създаване и зареждане на проект, стартиране на проект). (2 т.)

- File/New/Project/избираме вида апликация/въвеждаме име/, натискаме Enter и пишем код.???

2.

2.1 Описва понятието променлива. (2 т.)

- *Променлива* – заделено място в паметта, променящо стойността си по време на изпълнение. В зависимост от типа се заделя различен размер памет. Характеризират се с име, размер и стойност.

2.2 Сравнява типове променливи. (4 т.)

- Тип данни - определя множеството от допустими данни и операциите, които могат да се извършват с променливата.
  В C# се включват следните видове:
  - Целочислени типове – sbyte, byte, short, ushort, int, uint, long, ulong
  - Числа с плаваща запетая – float, double
  - Булев тип – bool

- Символен тип – char
- Символен низ (стринг) – string
- Обектен тип – object

2.3 Разработва програми/програмни фрагменти с аритметични и логически изрази с участието на променливи и числа. (6 т.)

```
int a, b;

a = int.Parse(Console.ReadLine());

b = int.Parse(Console.ReadLine());

if(a > b)

{

Console.WriteLine("a>b");

}else if(a < b)

{

Console.WriteLine("a<b");

}
```

3.

3.1 Описва условни конструкции – пълна и кратка форма. (2 т.)

- *Условните конструкции* - функции на езика за програмиране, чрез които можем да изпълняваме различни действия в зависимост от някакво условие. Имат:
  - Кратка форма

```
if (<условие>)
{
        //тяло на условната конструкция
}
```

  - Пълна форма

Конструкцията if може да съдържа и else клауза, с която да окажем конкретно действие в случай, че булевият израз (който е зададен в началото if (булев израз)) върне отрицателен резултат (false). Така построена, условната конструкция наричаме if-else и поведението ѝ е следното: ако резултатът от условието е позитивен (true) – извършваме едни действия, а когато е негативен (false) - други. Форматът на конструкцията е:

```
If (<условие 1>)
{
        //тяло на условната конструкция
}else{
```

```
        //тяло на else конструкция
}

или

If (<условие 1>)
{
        //тяло на условната конструкция
}else if(<условие 2>){
        //тяло на else конструкция
}
```

3.2 Описва оператор за многовариантен избор (switch). (2 т.)

- *Оператор за многовариантен избор (switch)* – използва дадена променлива, в зависимост от стойността на която се извършва определено действие (case). Пример:
  - switch (<променлива>)

```
case <стойност 1>:
        <действие>;
        break;
case <стойност 2>:
        <действие>;
        break;

default: // всички стойности, за които не е предвидено действие

        break;
```

Конструкцията switch-case работи като поредица if-else блокове. Когато работата на програмата ни зависи от стойността на една променлива, вместо да правим последователни проверки с if-else блокове, можем да използваме условната конструкция switch . Тя се използва за избор измежду списък с възможности. Конструкцията сравнява дадена стойност с определени константи и в зависимост от резултата предприема действие.

3.3 Сравнява условни конструкции с няколко условия (else if) и оператор за многовариантен избор (switch). (4 т.)

- else if обхожда всяко едно условие, докато то не се изпълни, а switch директно намира стойността на променливата и изпълнява съответния case.
- else if се използва за сравнение на две променливи, проверка на булев израз и др., докато switch проверява на стойността на дадена променлива.
- switch е по-четим от else if

3.4. Определя кои фрагменти от код се изпълняват, колко пъти и в какъв ред при условен оператор и/или многовариантен избор. (8 т.)

4.

4.1 Описва програми/програмни фрагменти с оператори за цикли (2 т.)

- *Цикъл (loop)* е основна конструкция в програмирането, която позволява многократно изпълнение на даден фрагмент сорс код. В зависимост от вида на цикъла, програмният код в него се повтаря или фиксиран брой пъти (for), или докато е в сила дадено условие(while & do-while).

4.2 Разработва програми/програмни фрагменти с оператори за цикли. (6т.)

- *while*:

```
int a;
a = 0;
while (a <= 10)
{
   a++; // добавяме 1 към „a“
   Console.WriteLine($"a = {a}");
}
```

- *do-while:*

```
int a;
do
{
    a = int.Parse(Console.ReadLine());
} while (a != 10);

Console.WriteLine($"a = {a}");
```

- *for:*

```
int a;
a = 0;
for (int i = 0; i < 10; i++)
{
   a++;
   Console.WriteLine($"a = {a}");
}
```

4.3 Различава операторите за цикли. (4 т.)

- За разлика от while и do-while, при for се задава колко пъти да се повтори тялото на цикъла;
- При do-while първо се извършва зададеното в цикъла действие, след което се проверява дали условието е изпълнено, докато при
- while първо се проверява условието и след това се извършва даденото действие.

4.3 Определя и посочва кои фрагменти код се изпълняват, колко пъти и в какъв ред при оператори за цикли (8 т.)

5.

5.1 Обяснява същността на подпрограмите (функции/методи). (2 т.)

- *Метод* – именувано парче код, което решава даден проблем и които могат да бъдат извикани толкова пъти, колкото имаме нужда. Методът може да приема параметри и да връща стойност. Пример:

```
internal class Program
  {
    public static double GetSquare (double num)
      return num * num;
    }
    static void Main(string[]args)
    {
      GetSquare(2);
    }
  }
```

В езика C# декларираме методите в рамките на даден клас, т.е. между отварящата { и затваряща } скоби на класа. Декларирането представлява регистрирането на метода в програмата, за да бъде разпознаван в останалата част от нея.

[static] <return_type> <metod_name>([param_list])

Задължителни елементи в декларацията на един метод:

– Тип на връщаната стойност. Връщаната стойност може да бъде както int , double , string и т.н., така и void . Ако типът е void , то това означава, че методът не връща резултат, а само изпълнява дадена операция.

– Име на метода - определено е от нас, трябва да описва функцията, която е изпълнявана от кода в тялото му. В примера името е GetSquare , което ни указва, че задачата на този метод е да изчисли лицето на квадрат.

– Списък с параметри. Декларира се между скобите (и) , които изписваме след името му. Тук изброяваме поредицата от параметри, които метода ще използва. Може да присъства само един параметър, няколко такива или да е празен списък. Ако няма параметри, то ще запишем само скобите () . В конкретния пример декларираме параметъра double num .

5.2 Посочва видове параметри и връщани стойности (4 т.)

- Параметрите биват три вида: входни, изходни, входно-изходни.

Таблица, показваща различията между трите вида параметри

Вид	входни	изходни	входно-изходни
Предават се	по стойност	по адрес	по адрес
Фактически параметър е	израз от типа на формалния	променлива от типа на формалния	инициализирана променлива от типа на формалния
Методът работи с	копие на	оригинала на	оригинала на

	фактически параметър	фактическия параметър	фактическия параметър
Служебна дума	няма	out	ref
Примери	(int a, float x)	(out int s, out int pr)	(ref int s, ref int pr)

5.3 Дава пример за предимствата от използването на методи. (10 т.)

- Разделяме големи програми на малки части.
- По-добро структуриране и по-добра четимост на кода.
- Подобряват разбираемостта на кода.
- Избягване на повторението на код.
- Подобряват поддръжката на кода.
- Преизползване на кода.

6. Анализира фрагмент/и от код и идентифицира и поправя правилно грешките в написания програмен код, така че да реши поставената задача. Допълва кода, ако и когато това е необходимо. (24 т.)