***Сорс-контрол системи - устройство и видове. Команди за работа със сорс-контрол системи. Видове типове данни, бройни системи и понятие за обект. Работа с масиви (едномерни и многомерни) и списъци. Дебъгване и работа с дебъгер. Символни низове и обработка на текст. Речници (хеш-таблици)***

1. **Сорс-контрол системи - устройство и видове**

**Git and GitHub**

Използваме системите за контрол на версиите за улесняване на работата в екип Системите за контрол на версиите използват централно хранилище, в което се съхраняват файловете на проекта. Когато някой започне да работи по този проект, той създава копие на файловете от това хранилище на своята система/локалния компютър/ и работи с тези копия. В процеса на работа потребителят може да изпраща към хранилището направените от него промени и да получава промените, направени по проекта от другите хора в екипа.След обновяване на хранилището, в него се запазват старите версии на променените файлове. Така тези файлове могат да бъдат възстановени при необходимост. Системата за контрол на версиите също така следи за конфликти – различни промени на различните потребители, които ползват хранилището и ги разрешава.

1. **Команди за работа със сорс-контрол системи**
2. **Git-разпределена система за контрол на   
   програмния код (source-control system)**
   * Най-популярната в света (към момента)
   * Свободен софтуер с отворен код
   * Работи с локални и отдалечени хранилища

**\*Git bash – команден интерфейс към Git. Основни Git команди**

* Клониране на съществуващо Git хранилище

git clone [отдалечен url]

* Изтегляне и сливане на промени от отдалечено хранилище

git pull

* Подготовка (добавяне / избор) на файлове за запис

git add [файл] ("git add ." добавя всичко)

* Предаване (commit) към локалното хранилище

git commit –m "[вашето съобщение]"

* Проверка на статуса (промените) в локалното хранилище

git status

* Създаване на ново локално хранилище (в текущата папка)

git init

* Създаване на отдалечено (+ кратко име за отдалечен Git URL)

git remote add [remote name] [remote url]

* Изпращане на промени (към отдалечено хранилище)

git push [remote name] [local name]

**Пример:**

cd Examination

dir

git status

*(edit some file)*

git add .

git commit -m "changes"

git push

Променяме директорията на Examination.

Извеждаме съдържанието на тази директория.

Проверяне статуса .

Редактираме файл .

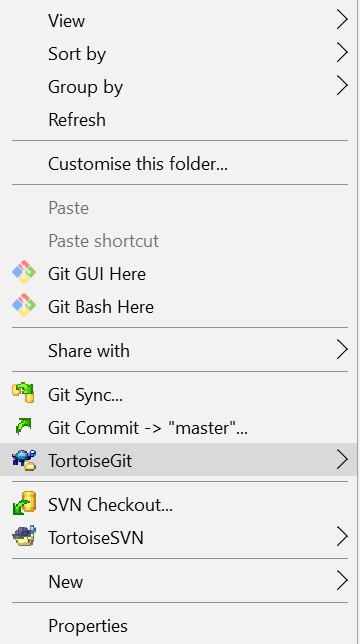
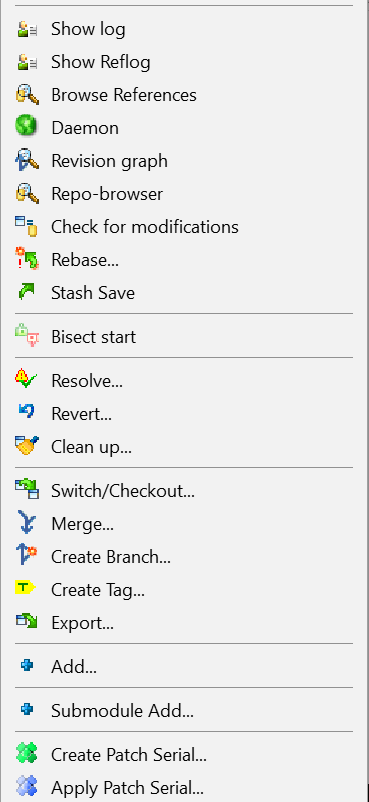
Добавяме към локалното хранилище всички файлове, които са променени.

Предаваме промените към локалното хранилище като добавяме съощение "changes"

Изпращаме към отдалеченото хранилище.

**\*TortoiseGit-** **Графичен клиент**

Диалогови прозорци за основните команди и работа с репозиторита

**** 

1. **GitHub-** **Социална мрежа за разработчици**

GitHub е сайт за хостване на програмен код. Безплатен за проекти с отворен код и платени планове за частни хранилища.

GitHub предоставя:

* + Git хранилище за програмен код
  + Тракер на задачи (бъг тракер)
  + Табло на проекта (тип Канбан)
  + Уики страници (документация)

1. **Видове типове данни, бройни системи и понятие за обект**

**Видове типове данни**

**Типовете данни** представляват множества (диапазони) от стойности, които имат еднакви характеристики. Типовете данни се характеризират с:

-     Име – например цели числа в C# int;

-     Размер (колко памет заемат) – например 4 байта;

-     Стойност по подразбиране (default value) – например 0.

Базовите типове данни в C# се разделят на следните видове:

1. **Целочислени типове**

Целите числа имат диапазон (минимална и максимална стойност). Могат да пазят или да не пазят знак.

* **sbyte** [-128 …127]: със знак, 8-битов [-27 … 27-1]
* **byte** [0 … 255]: без знак, 8-битов [0 … 28-1]
* **short** [-32 768 … 32 767]: със знак, 16-битов [-215 … 215-1]
* **ushort** [0 … 65 535]: без знак 16-битов [0 … 216-1]
* **int** [-2 147 483 648 … 2 147 483 647]: със знак 32-битов [-231 … 231-1]
* **uint** [0 … 4 294 967 295]: без знак 32-битов [0 … 232-1]
* **long** [-9 223 372 036 854 775 808 … 9 223 372 036 854 775 807]: със знак 64-битов [-263 … 263-1]
* **ulong** [0 … 18 446 744 073 709 551 615]: без знак 64-битов [0 … 264-1]

Примери:

**byte Age = 20;**

**uint Days = 700005;**

**int HexDigit=0xFE;**

**long AstroWay=9876543L;**

1. **Реални типове с плаваща запетая – float, double;**

* **float (±1.5 × 10−45 до ±3.4 × 1038) 32-битов, точност 7 знака след запетаята, стойността по подразбиране 0.0F**
* **double (±5.0 × 10−324 до ±1.7 × 10308) 64-бита, точност от 15-16 знака след запетаята , стойността по подразбиране 0.0D**

Съдържат реални числа, например: **1.25**, **-0.38**

Примери:

**float floatPI = 3.141592f;**

1. **Реални типове с десетична точност – decimal;**
   * **decimal** (±1,0 × 10-28 до ±7,9 × 1028)
     + 128-битов, с точност до 28-29 знака
   * Използва се за финансови изчисления

-     Булев тип – bool;

-     Символен тип – char;

-     Символен низ (стринг) – string;

-     Обектен тип – object.

1. **Бройни системи**
   1. **Двоична бройна система**

**Основа-2**

**Цифри-0,1**

|  |  |  |
| --- | --- | --- |
| **+** | **0** | **1** |
| **0** | **0** | **1** |
| **1** | **1** | **10** |

**Двоичната бройна система** (**binary numeral system**), е системата, която се използва за представяне и обработка на числата в съвременните електронноизчислителни машини.

* 1. **Шестнадесетична бройна система**

**Основа-16**

**Цифри-0,1,2,3,4,5,6,7,8,9,А,B,C,D,E,F**

* 1. **Десетична бройна система**

**Основа-10**

**Цифри-0,1,2,3,4,5,6,7,8,9**

**Преобразуване от десетична в двоична бройна система**

148:2=74 имаме остатък 0;

74:2=37   имаме остатък 0;

37:2=18   имаме остатък 1;

18:2=9     имаме остатък 0;

9:2=4       имаме остатък 1;

4:2=2       имаме остатък 0;

2:2=1       имаме остатък 0;

1:2=0       имаме остатък 1;

След като вече семе извършили деленето, записваме стойностите на остатъците в ред, обратен на тяхното получаване, както следва:

10010100

т.е. 148(10) = 10010100(2)

**С код в C#**

**………………**

**Преобразуване от двоична в дестична бройна система**

**11001**(2)**=1**×24+ **1**×23+**0**×22+**0**×21+**1**×20 =

=  **16**(10)+ **8**(10)+ **1**(10) = **25**(10)

От това следва, че **11001**(2)= **25**(10)

**С код в C#**

**………………**

.