***Д И П Л О М Н А Р А Б О Т А***

**На тема:**

*Учебно помагало „Въведение в програмирането със C#“* *за 8 клас общообразователна подготовка*

**Дипломант:** Дейвид Светланов Младенов

**Фак.номер:** 1601651011

**Научен ръководител:** доц. Д-р Коста Гъров

**Съдържание:**

1. Увод
2. Учебна програма по информатика за VIII клас (общообразователна подготовка)
3. Задачите в обучението по програмиране
4. Заключение
5. Приложение с решения на всички задачи
6. Увод

България е една от първите страни в света, която включва информатиката като задължителна дисциплина в своята учебна програма. Това се случва през учебната 1986-1987 година. По това време всички учебни програми и учебници са обвързани с наличната материална база от 8 битови персонални компютри Правец и език за програмиране Бейсик. Материалът по информатика бива разделен на две части -първо и второ равнище, като започва да се изучава в 6-ти клас по 64 часа годишно. В материала от първо равнище или наречен още задължителен минимум, се засягат теми като променливи, процедури, рекурсивни процедури, структури от данни, информационни системи, модифициране на програми и електронни пощи. Вижда се, че темите от учебният план са били на много високо ниво и могат да се сравняват с материал, присъщ на съвременни гимназиални или дори на някои университетски учебни планове.

В наши дни сме заобиколени от компютри. Всеки един от нас носи компютър в джоба си, а именно мобилния си телефон. Заради тази разпространеност на компютрите все повече ученици се насочват към изучаването информатика и информационни технологии, поради жаждата си да могат да ги управляват и те да изпълняват техните команди. Изучаването на информатика в училище дава стабилна основа на младите ученици за развитие в областта.

Обект на тази разработка ще бъде съставянето на учебно помагало за упражнение съгласно „Учебна програма по информатика за VIII клас“ (Общообразователна подготовка) и далеч по-модерни технологии за програмиране от споменатите по-горе в лицето на езика C# и .NET Core 3.0 платформата.

C# е съвременен обектно-ориентиран език за програмиране разработен от Microsoft, който се появява пред света през 2000-та година. В днешно време C# е един от най-популярните езици за програмиране и се използва от милиони разработчици по цял свят. Като език от високо ниво C# се слави с лесния си синтаксис и се препоръчва за начинаещи, които имат интерес в сферата на програмирането и именно за това съм избрал точно него за примерите в тази разработка.

1. **Учебна програма по информатика за VIII клас (общообразователна подготовка)**

**Очаквани резултати от обучението в края на класа**

|  |  |
| --- | --- |
| **Област на компетентност** | **Знания, умения и отношения**  **В резултат на обучението ученикът:** |
| **Информатика** | Описва предмета и ролята на информатиката за моделиране |
| Познава представянето на информация във вид на данни |
| Посочва примери на обекти и явления, при които е практически приложимо използването на средствата на обектно-ориентираното моделиране |
| **Обектно-ориентирано**  **програмиране** | Обяснява основните етапи при създаване и изпълнение на компютърна програма |
| Прилага обектно-ориентиран подход при създаване на несложна компютърна програма |
| Описва основни начини за създаване, изпълнение и тестване на програмен проект в интегрирана среда за разработка с използване на визуални графични средства |
| Използва библиотеки от готови компоненти |
| Спазва добър стил на програмиране |
| **Графичен**  **потребителски**  **интерфейс** | Разбира и използва основни компоненти на среда за визуално програмиране при разработка на софтуер |
| Проектира графичен потребителски интерфейс с визуални средства |
| Избира подходяща графична компонента в съответствие с необходимата функционалност на графичния интерфейс |
| Умее да настройва свойствата на графичните компоненти |
| Програмира подразбиращи се събития за основни компоненти от графичния интерфейс |
| **Алгоритми и структури**  **от данни** | Разбира същността на „тип данни“ |
| Разграничава различни типове данни |
| Определя подходящ тип данни за определена задача |
| Разбира същността на алгоритмите и начини за описанието им |
| Прилага основни управляващи конструкции |
| Структурира данни в едномерен масив |
| Прилага основни алгоритми за намиране на сума, минимален/максимален елемент и средно аритметично |
| Чете и записва данни в текстов файл |
| **Софтуерни**  **приложения** | Използва визуално програмиране за решаване на несложни задачи |
| Създава програми за графично изобразяване на геометрични обекти със стандартни средства в езика |
| Създава програмни приложения с мултимедийни компоненти |
| Представя аргументирано разработено софтуерно приложение пред публика |

**Учебно съдържание**

|  |  |  |
| --- | --- | --- |
| **Теми** | **Компетентности като очаквани резултати от обучението** | **Нови понятия** |
| **1. Основи на информатиката** | | |
| **1.1. Числата и техните представяния** | * Разширява и обобщава знанията, свързани с числата и технитепредставяния:   + непозиционни бройни системи;   + същност на позиционните бройни системи;   + формат на числата в десетична, двоична и шестнадесетична бройна система. * Превръща числа от десетична в двоична бройна система и обратно. * Извършва събиране, изваждане и умножение на две числа в двоична бройна система. * Дава примери за използване на двоична и шестнадесетична бройна система. | * непозиционни бройни системи * позиционни бройни системи * експоненциален формат и неговото предназначение * двоична бройна система * шестнадесетична бройна система |
| 1.2. Информационни дейности и процеси | * Обяснява предмета на информатиката и ролята ѝ в съвременното общество. * Изброява и описва основните информационни дейности събиране, съхраняване, преработка и разпространение и общата схема на информационните потоци. * Описва понятието информационен процес и дава примери на информационни процеси, свързани с решаване на житейски задачи. * Различава понятията информация и данни. * Обяснява и илюстрира с примери връзката между информация и данни. * • Обяснява и илюстрира с примери същността на дискретното представяне на информацията за трансформирането ѝ в данни. | * основни информационни дейности * информация, данни, дискретно представяне на информацията |
| 1.3. Алгоритми и езици за програмиране | * Дефинира понятието алгоритъм и описва основните му характеристики (резултатност, крайност, детерминираност, масовост). * Описва и проследява несложни, линейни и разклонени алгоритми с различни средства. * Описва същността, структурата и разновидностите на цикличните алгоритмични конструкции. * Обяснява същността и функционалното предназначение на език за програмиране. * Обяснява същността и предназначението на транслатор (интерпретатор, компилатор). * Представя исторически факти, свързани със създаването и развитието на съвременните езици и среди за програмиране. | * алгоритъм * език за програмиране * транслатор |
| **2. Среда за визуално програмиране** | | |
| 2.1. Интегрирана среда за визуално програмиране | * Отваря проект в интегрирана среда за визуално програмиране. * Редактира дизайна на графичния потребителски интерфейс на приложение в интегрирана среда за програмиране. * Запазва проект на приложение чрез средствата на интегрирана среда за програмиране. * Стартира приложение с графичен потребителски интерфейс чрез средствата на интегрирана среда за програмиране. * Разпознава основни компоненти на интегрирана среда за програмиране – графичен и текстов редактор, панел с контроли, панел за свойства на обект, панел за съобщения, панел за преглед на структурата на приложението. | * интегрирана среда за програмиране * свързващ редактор (linker) * програма за откриване и отстраняване на грешки (debugger) * редактор за проектиране на дизайн на графичен потребителски интерфейс |
| 2.2. Основни етапи на създаване и изпълнение на компютърна програма | * Анализира задача с несложен математически модел. * Създава математически модел за решаване на несложна задача. * Съпоставя математически модел с програмно решение на даден проблем. * Открива основните компоненти на математически модел в демонстрирано програмно решение на даден проблем. * Стартира чрез средствата на интегрирана среда предварително подготвена компютърна програма с графичен потребителски интерфейс. * Тества предварително подготвен несложен проект. * Разпознава видовете грешки при програмиране. * Разчита и прави предположение за естеството на синтактична грешка в даден проект. * Открива и прави предположение за причината за логическа грешка в дадено приложение. * Открива и прави предположение за причината за грешка по време на изпълнение на приложението. | * синтактични грешки в компютърна програма * логически грешки в компютърна програма * грешки по време на изпълнение на програмата |
| 2.3. Проектиране на графичен потребителски интерфейс | * Знае предназначението на основни контейнери и контроли – форма, етикет, текстово поле, бутон, диалогова кутия. * Разпознава основни свойства на графични обекти-контроли – име, състояние, етикет, фон, настройка на шрифт и др. * Проектира несложна форма, съдържаща етикет, текстово поле, бутон. * Настройва основни свойства на форма, етикет, текстово поле и бутон. * Именува обекти-контроли съгласно общоприета конвенция. * Задава функционалност на бутон, свързана с извеждането на статично съобщение в диалогова кутия. | * интерфейсен компонент (контрола) * контейнер на контроли * свойство на обект * метод на обект |
| **3. Програмиране** | | |
| **3.1. Основни типове данни** | | |
| 3.1.1. Тип низ | * Познава правила за именуване на константи и променливи. * Декларира, описва и инициализира променливи и константи от тип низ. * Присвоява стойност на променлива от тип низ. * Въвежда и извежда данни от тип низ в/от текстово поле. * Извежда данни от тип низ в/от етикет. * Извършва конкатенация на низове. * Използва стандартни методи на интерфейсни компоненти за форматиране на текст. | * символ * низ * множество на допустимите данни * множество на допустимите операции * име, тип стойност на променлива * име, тип и стойност на константа * присвояване на стойност * конкатенация |
| 3.1.2. Целочислени типове данни | * Декларира, описва и инициализира променливи и константи от целочислен тип данни. * Използва вградени функции за преобразуване на низ в цяло число и обратното. * Въвежда и извежда данни от целочислен тип. * Използва различни целочислени типове данни. * Познава целочислените аритметични операции и техния приоритет. * Конструира аритметични изрази, съдържащи само целочислени данни, спазвайки синтаксиса и семантиката на конкретния език за програмиране. * Прилага и анализира резултатите от операциите – събиране, изваждане, умножение, деление, цяла част и остатък от целочислено деление. * Реализира модел за решаване на задачи, базиран на целочислени типове данни. | * целочислен тип данни * конвенция за именуване на константи и променливи |
| 3.1.3. Реални типове данни | * Декларира, описва и инициализира променливи и константи от реален тип. * Използва вградени функции за преобразуване на низ в реално число и обратното. * Въвежда и извежда данни от реален тип данни. * Използва различни реални типове данни. * Познава приоритетите на аритметичните операции при реални типове данни. * Конструира аритметични изрази, съдържащ реални типове данни, спазвайки синтаксиса и семантиката на конкретния език за програмиране. * Прилага и анализира резултатите от операциите – събиране, изваждане, умножение, деление. * Реализира модел за решаване на задачи, базиран на реални типове данни. | * реален тип данни |
| 3.1.4. Аритметични изрази и вградени математически функции. Приоритет на операциите | * Оценява числената стойност на аритметичен израз, записан на език за програмиране. * Записва аритметичен израз със средствата на език за програмиране. * Прилага и използва вградени в езика за програмиране математически функции – абсолютна стойност, повдигане на степен, закръгляване, извличане на цялата част на реално число. * Използва приоритет на операциите в аритметични изрази, съдържащи вградени функции. * Създава аритметични изрази, съдържащ различни типове данни, като се съобразява със съвместимостта им. * Описва синтаксис и семантика на оператор за присвояване. * Форматира изхода на реално число. | * аритметичен израз в език за програмиране * вградени математически функции в език за програмиране * съвместимост на типове данни * форматиран изход |
| **3.2 Създаване на компютърна програма за решаване на конкретна задача** | | |
|  | * Знае основните етапи при създаване на компютърна програма. * Анализира и проектира решението на конкретна задача. * Създава математическия модел за решаване на задачата. * Разработва алгоритъм за решаване на задачата. * Определя входно-изходни данни и техните типове. * Структурира и разработва графичен интерфейс, като използва обекти и декларира променливи. * Създава и описва програмния код. * Стартира, тества и валидира готовия проект. * Открива синтактични и логически грешки в програмата. * Отстранява синтактични и логически грешки при програмиране. * Спазва изисквания за оформяне на програмния код, включващи подравняване, коментари, именуване на програмните единици. | * коментари * оформяне на програмния код |
| **3.3. Програмни конструкции за реализация на разклонен алгоритъм** | | |
| 3.3.1. Булев тип данни | * Обосновава необходимостта от разклоняване на алгоритмичния процес. * Използва константите от булев тип данни. * Дава примери, в които се използва булев тип данни. * Декларира булева променлива. * Присвоява стойност на булева променлива. * Записва на език за програмиране булев израз, съдържащ операция за сравнение. * Изписва синтактично правилно на езика за програмиране основните логически операции – логическо отрицание, дизюнкция, конюнкция. * Познава приоритета на логическите операции. * Пресмята без използване на компютър стойността на булев израз. * Съставя със средствата на език за програмиране сложен булев израз, отговарящ на дадена логическа ситуация. | * булев тип данни * булеви константи – false, true * логически операции * приоритет на логическите операции * булев израз |
| 3.3.2. Условен оператор | * Описва синтаксиса и семантиката на кратка и пълна форма на условен оператор. * Описва разклонен алгоритъм с помощта на условен оператор. * Използва условен оператор за проверка на коректността на входните данни за програма. * Използва условен оператор за обработка на свойства на радиобутон и поле за отметка. | * условен оператор * съставен оператор |
| 3.3.3. Вложени условни оператори | * Обяснява семантиката на вложени условни оператори в кратка и пълна форма. * Проиграва изпълнението на фрагмент на програма, съдържаща вложени условни оператори. * Записва синтактично и логически правилно вложени условни оператори. * Създава модел и алгоритъм за решаване на задача чрез използване на вложени условни конструкции. * Реализира модел за решаване на задача чрез използване на вложени условни оператори. * Заменя вложен условен оператор с единичен и обратно. | * вложен условен оператор |
| **3.4. Програмни конструкции за реализация на циклични алгоритми** | | |
| 3.4.1. Циклични алгоритми | * Посочва елементите на циклична конструкция – инициализация, тяло на цикъла и условие на цикъла. * Оценява необходимостта от използването на алгоритми с циклични конструкции с предусловие и постусловие. * Записва синтактично и логически правилно оператори за цикъл с предусловие, постусловие и управлявани от брояч. * Прилага алгоритми с циклични конструкции за проверка на входни данни. * Оценява необходимостта от използване на алгоритми с циклични конструкции с условие или управлявани от брояч. * Преобразува програмен код, съдържащ циклична конструкция, управлявана от брояч в циклична конструкция или управлявана от условие. * Открива синтактични и логически грешки в програмния код на алгоритми с циклична конструкция. * Прилага циклични алгоритми за управление на графичен потребителски интерфейс. | * структура на циклични алгоритмични конструкции * оператор за цикъл с предусловие * оператор за цикъл с постусловие * оператор за цикъл, управляван от брояч * списъчно поле |
| 3.4.2. Приложение на условни и циклични конструкции | * Прилага циклични алгоритми за изчертаване на графични примитиви. * Използва циклични алгоритми за въвеждане и извеждане на данни от файл. * Прилага програмни конструкции за реализация на алгоритми за намиране на сума, минимален/максимален елемент, средно аритметично и др. в редици от числа, въвеждани от потребителския интерфейс/клавиатурата. | * графичен примитиви * кутия за изображения * текстов файл |
| **3.5. Тестване и верификация на програма** | | |
|  | * Обяснява и разграничава понятията тестване и верификация. * Дефинира тестови данни. * Дефинира очаквани резултати от тестването при определени входни данни. * Използва инструмент за откриване и отстраняване на грешки (debugger). * Прилага процедури за тестване и верификация на вече създадени програми. | * тестване * верификация * тестови данни |
| **3.6. Съставни типове данни. Едномерен масив** | | |
| 3.6.1. Едномерен масив | * Разбира необходимостта от използване на масиви. * Идентифицира елементите на масив. * Разпознава индекс и стойност на елемент на масив. * Дефинира масив със средствата на език за програмиране. * Създава и инициализира масив със средствата на език за програмиране. * Осъществява достъп до елемент на масив. * Обхожда, въвежда и извежда стойностите на елементите на масив. * Използва списъчно поле за извеждане на стойностите на елементите на масив. | * логическо описание на масив * базов тип на масив * индекс и стойност на елемент от масив |
| 3.6.2. Основни алгоритми за работа с едномерен масив | * Пресмята сбор и произведение на стойностите на елементите на едномерен масив. * Търси елемент от масива с максимална и минимална стойност. * Търси елементи от масива, отговарящи на дадено условие. | * последователно търсене |
| **4. Създаване на софтуерен проект** | | |
|  | * Описва етапите при реализиране на софтуерен проект. * Извършва проучване и анализ на решения за даден групов проект. * Създава модел за решаване на проблема, поставен в заданието на проекта. * Проектира графичен потребителски интерфейс. * Създава програмен код за реализация на модела. * Създава тестови примери с входни данни и очаквани резултати. * Изготвя документация за софтуерния проект. * Презентира и защитава готовия софтуерен проект. |  |

1. Задачите в обучението по програмиране

В тази глава ще срещнете креативни авторски задачи от различно естество, които са пряко свързани с упражнение върху материала обхващащ учебният план за 8 клас общообразователна подготовка. Задачите са направени така, че да се покаже практическото приложение на изучаваният материал, а не само прости примери чрез, които учениците да не могат да разберат как да прилагат изученото. Всички задачи са приложими за различни езици за програмиране, но в тази разработка ще работим изцяло в контекста на езика C# и .NET Core 3.0 платформата, като към момента на разработка това е най-новата и стабилна версия на платформата. За интегрирана среда за разработка на софтуер (IDE – Integrated Development Environment) ще използваме Visual Studio 2019, което е достъпно да бъде свалено от сайта на Microsoft, напълно безплатно (Visual Studio 2019 Community).

**Бройни системи**

Бройната система представлява символен метод за представяне на числата посредством ограничен брой символи, наречени цифри. Съществуват два вида бройни системи – непозиционни и позиционни.

Непозиционна бройна система наричаме тази бройна система, чийто стойности на всяка цифра не зависят от позицията им. Примери за такива бройни системи са гръцката и римската:

|  |  |
| --- | --- |
| **Гръцка цифра** | **Десетична стойност** |
| Ι | 1 |
| Г | 5 |
| Δ | 10 |
| Η | 100 |
| Χ | 1000 |
| Μ | 10000 |

|  |  |
| --- | --- |
| **Римска цифра** | **Десетична стойност** |
| I | 1 |
| V | 5 |
| X | 10 |
| L | 50 |
| C | 100 |
| D | 500 |
| M | 1000 |

Позиционна бройна система наричаме тази бройна система, чийто стойности на всяка цифра зависят от позицията им. Това означава, че стойността на цифрата в числото не е строго определена и зависи от това на коя позиция се намира съответната цифра в дадено число. Примери за такива бройни системи са двоичната, осмичната, десетичната и шестнадесетичната:

|  |  |  |  |
| --- | --- | --- | --- |
| **Двоична** | **Осмична** | **Десетична** | **Шестнадесетична** |
| 0000 | 0 | 0 | 0 |
| 0001 | 1 | 1 | 1 |
| 0010 | 2 | 2 | 2 |
| 0011 | 3 | 3 | 3 |
| 0100 | 4 | 4 | 4 |
| 0101 | 5 | 5 | 5 |
| 0110 | 6 | 6 | 6 |
| 0111 | 7 | 7 | 7 |
| 1000 | 10 | 8 | 8 |
| 1001 | 11 | 9 | 9 |
| 1010 | 12 | 10 | A |
| 1011 | 13 | 11 | B |
| 1100 | 14 | 12 | C |
| 1101 | 15 | 13 | D |
| 1110 | 16 | 14 | E |
| 1111 | 17 | 15 | F |

Как се образуват числата от десетичната бройна система? За нашият пример ще използваме числото 27865, представяме всяка една цифра образуваща числото като започнем от ляво на дясно по следната формула:

27865 съдържа 5 символа, така че решението на задачата изглежда по следния начин:

27865 =

За разлика от десетичната бройна система, която е лесно разбираема от човека двоичната бройна система е по-сложно, но същевременно разбираема за изчислителната машина (компютър). За представяне на число в двоична бройна система се използват само числата 0 и 1. Прието е, когато едно число се записва в бройна система, различна от десетичната, във вид на индекс в долната му част да се отразява, коя бройна система е използвана за представянето му. Например със записа 1110(2) означаваме число в двоична бройна система. Важно е да знаем, че ако едно двоично число завършва на 0, то е четно, а ако завършва на 1, то е нечетно.

Преобразуването на число от двоична в десетична бройна система става, когато всяка една двоична цифра се умножава по 2 на степен, позицията, на която се намира:

11001(2) = 124 + 123 + 022 + 021 + 120 = 16(10) + 8(10) + 1(10) = 25(10)

От това следва, че 11001(2) = 25(10)

При преминаване от десетична в двоична бройна система, се извършва преобразуване на десетичното число в двоично. За целите на преобразуването се извършва делене на две с остатък. Така се получават частно и остатък, който се отделя.

148 : 2 = 74 остатък 0;

74 : 2 = 37 остатък 0;

37 : 2 = 18 остатък 1;

18 : 2 = 9 остатък 0;

9 : 2 = 4 остатък 1;

4 : 2 = 2 остатък 0;

2 : 2 = 1 остатък 0;

1 : 2 = 0 остатък 1;

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

148(10) = 10010100 (2)

Аритметични правила за събиране, изваждане и умножение на двоични числа:

|  |  |  |
| --- | --- | --- |
| 0 + 0 = 0 | 0 – 0 = 0 | 0 x 0 = 0 |
| 1 + 0 = 1 | 1 – 0 = 1 | 1 x 0 = 0 |
| 0 + 1 = 1 | 1 – 1 = 0 | 0 x 1 = 0 |
| 1 + 1 = 10 | 10 – 1 = 1 | 1 x 1 = 1 |

**Интегрирана среда за разработка (IDE)**

Интегрираната среда за разработка представлява софтуерно приложение, което разполага с пълния набор от инструменти нужни на програмиста да създаде собствено софтуерно приложение. То се състои от няколко основни компонента: редактор на код, компилатор/интерпретатор, дебъгер, система за контрол на версиите. В контекста на тази разработка интегрираната среда за разработка, която ще използваме е Visual Studio 2019. След като сте изтеглили и инсталирали Visual Studio 2019, може да преминем към запознаване с неговият интерфейс.

A screenshot of a cell phone

Description automatically generated

След отваряне на Visual Studio 2019, попадаме на началния екран където има три основни секции. В секцията номерирана с номер 1 на снимката, виждаме последно отваряните проекти, в секция номер 2 са някои основни преки пътища, които за момента няма да използваме. Секция номер 3 е желаната от нас натискаме върху надписа. Попадаме в един празен екран, забелязваме че интегрираната среда за разработка наподобява много на обикновена програма тоест има менюта и бутони. Основни секции на интегрираната среда за разработка:

1. стандартно навигационно меню (navigation menu)
2. често използвани функции на средата и преки пътища (shortcuts)
3. файловете на решението (solution explorer)
4. списък с компилационни грешки и предупреждения (errors and warnings)
5. текстови редактор на кода (text editor)

A screenshot of a computer

Description automatically generated

**Създаване на решение (Solution)**

A screenshot of a cell phone

Description automatically generated

След като сме в интегрираната среда за разработка, от навигацията горе в ляво избираме File 🡪 New 🡪 Project.

В отворения прозорец има търсачка в нея изписваме „Blank“ и средата ни показва предложение за „Blank Solution“, селектираме го и натискаме бутона „Next“.

A screenshot of a video game

Description automatically generated

В първата секция слагаме говорещо за решението име за целта на демото го именуваме „MyFirstSolution“. От втората секция избираме местоположението на решението. След като сме готови натискаме бутона „Create“

A screenshot of a video game

Description automatically generated

В „Solution Explorer“ може да видим, че нашето празно решение е създадено успешно.

A black and red text

Description automatically generated

Първата ни програма

В това упражнение ще се запознаем с това как да създадем нашата първа програма, чийто цел ще бъде изписването на текст в конзолата. Ще разберем още как се стартира тя и някои важни концепции относно именуването и.

След като вече имаме създадено решение, а именно решението от предишното упражнение, което беше именувано „MyFirstSolution“ вече може да добавим първата ни програма към него.

A screenshot of a computer screen

Description automatically generated

1 – натискаме десен бутон на мишката върху решението

2 – избираме от менюто Add

3 – натискаме върху New Project

4 – избираме Console App (.NET Core)

5 – натискаме Next

A screenshot of a cell phone

Description automatically generated

6 – на полето „Project name“ трябва да обърнем малко повече внимание. Име специална конвенция за именуване на нашите проекти, която гласи следното, проект или решение винаги се именуват с описателни имена, като първата буква винаги е главна, ако името съдържа няколко думи никога не се слагат паузи, вместо паузи изписваме всяка следваща дума долепена до предходната като първата и буква трябва да е главна.

Примери:

* грешен пример – „Hello world program exercise“
* верен пример – „HelloWorldProgramExercise“

7 – Натискаме бутона Create

Важно е да се споменем, че когато добавяме програма във вече създадено решение, както е в случая, не променяме местоположението му (Location) за да бъде създадена програмата вътре във вече създаденото решение!

A screenshot of a video game

Description automatically generated

8 – както вече знаем това е решението

9 – програмата, която създадохме

10 – файла, в който ще въвеждаме кода на програмата ни

A picture containing clock

Description automatically generated

A screenshot of a cell phone

Description automatically generated

11 – отваряща и затваряща скоба на namespace HelloWorld, което означава, че това е обхвата (scope) на този namespace или казано по-просто, обхвата на програмата, която създадохме с име „HelloWorld“, който отговаря на секцията от 9-та стъпка

12 – отваряща и затваряща скоба на class Program, също както на предходната стъпка двете скоби дефинират обхвата на класа Program, който съответства на секцията от стъпка 10

13 – отваряща и затваряща скоба на метода Main, както при предходните два примера скобите и тук дефинират обхвата на метода Main, за методи ще говорим по-късно в тази разработка, за момента е нужно да знаем, че всичкия код, който искаме да бъде изпълнен трябва да се намира между отварящата и затварящата скоба на Main метода, защото той е началната точка при стартиране на конзолно приложение

14 – секция за писане на логически код

Нека се фокусираме върху последната секция и разгледаме по-подробно кода, който се състои в случая от един ред, но същевременно ще изпълни заданието ни. Ако разбираме Английски език, командите които са изписани ще ни се сторят доста описателни и естествени. Буквално казваме на компютъра „Конзола, напиши ред, който да съдържа следният текст 🡪 „Hello World!““. До тук сме се справили с задачата, но никъде не виждаме нито конзола, нито текст, това е защото нашата програма все още не е стартирана, за да я стартираме трябва да натиснем клавишната комбинация Ctrl + F5. След натискане на тази клавишна комбинацията, програмата се компилира тоест, проверява се за грешки, предупреждения или други проблеми, които могат да възпрепятстват нейното изпълнение и се стартира след това. Желаният резултат от изпълнението и трябва да изглежда по следния начин:

A screenshot of a computer

Description automatically generated

Може да се поздравим, понеже програмата изпълни исканията ни напълно, нямаше грешки или предупреждения. Ако за момента това Ви се струва малко, ако изпълните всички задания до края на курса, ще се научите да правите далеч по-сложни и полезни приложения. Нека обобщим наученото с решението на няколко задачи.

Когато имаме повече от един проект в нашето решение и искаме при натискане на комбинацията Ctrl + F5 да се стартира проекта, по който работим в момента трябва да направим следната настройка:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

1 – десен бутон върху решението

2 – избираме Set Startup Project

3 – отбелязваме Current selection и натискаме OK

**Задача 1:** Създайте решение (Solution) в което ще събирате всички задачи от обучението в тази разработка. Именувайте го по следния начин „IntroToProgrammingCSharpTasks“.

**Задача 2:** **TheHiProgram**

Създайте конзолно приложение, което изписва следния текст „Hello! I am <вашето име>“.

Желано поведение на програмата:

A close up of a logo

Description automatically generated

Примитивни типове данни и променливи

Всички модерни програми в днешно време съдържат променливи в кода си. Най-просто казано променливата (variable) представлява резервирано място в оперативната памет на компютъра, където ние можем да запазваме данни от различни видове (числа, символи, текст и т.н). От името им може да разберем, че тази стойност може да бъде променяна по време на изпълнение на програмата.

Типовете данни представляват диапазони от стойности, които имат еднакви характеристики. Съществуват няколко групи примитивни типове данни в C#, те се наричат примитивни защото са част от езика:

* Целочислени – sbyte, byte, short, ushort, uint, int, long, ulong
* Реални с плаваща запетая – float, double
* Реални с десетична точност – decimal
* Булев – bool
* Символен – char
* Символен низ – string
* Обект – object

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип данни** | **Стойност по подразбиране** | **Минимална стойност** | **Максимална стойност** |
| sbyte | 0 | -128 | 127 |
| byte | 0 | 0 | 255 |
| short | 0 | -32768 | 32767 |
| ushort | 0 | 0 | 65535 |
| int | 0 | -2147483648 | 2147483647 |
| uint | 0u | 0 | 4294967295 |
| long | 0L | -9223372036854775808 | 9223372036854775807 |
| ulong | 0u | 0 | 18446744073709551615 |
| float | 0.0f |  |  |
| double | 0.0d |  |  |
| decimal | 0.0m |  |  |
| bool | false | Има само две стойности true и false | |
| char | ‘\u0000’ | ‘\u0000’ | ‘\uffff’ |
| string | null | - | - |
| object | null | - | - |

Декларирането на променлива става по указаният начин, за целта на примера ще деклариране няколко променливи от различен тип:

A screen shot of a computer

Description automatically generated

В числата от едно до пет на схемата, са показани стъпките за деклариране на променлива:

1 – тип данни

2 – говорещо за променливата име

3 – знак равно, който означава присвояване на стойност

4 – началната стойност, която искаме да притежава нашата променлива

5 – знак точка и запетая, който означава край на декларацията

В останалите числа на схемата ще се запознаем с някои особености и конвенции:

6 – променливите в C# винаги се именуват с малка буква, ако името за да бъде достатъчно описателно е нужно да съдържа повече от една дума, то тогава всяка следваща дума се изписва с главна буква. В имена на променливи не се допуска слагане на паузи, служебни думи от езика или числа!

7 – двойни кавички използваме при декларация на символен низ

8 – единични кавички използване при декларация на един символ

**Задача 3: NumbersInDifferentNumeralSystems**

Създайте конзолно приложение, което пресмята стойността на числото 555 в двоична, осмична и шестнадесетична бройна система и изписва резултатите в конзолата.

Желано поведение на програмата:

A screenshot of a cell phone

Description automatically generated