МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ПРОГРАМУВАННЯ ЗАСОБІВ МУЛЬТИМЕДІА

Методичні рекомендації до самостійної роботи студентів

спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня

Харків
ХНЕУ ім. С. Кузнеця
2018

УДК 004.032.6(07)
П78

Укладач В. В. Браткевич

Затверджено на засіданні кафедри комп'ютерних систем і технологій.
Протокол № 7 від 12.12.2017 р.

Самостійне електронне текстове мережеве видання

Програмування засобів мультимедіа : методичні рекомендації до самостійної роботи студентів спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня [Електронний ресурс] / уклад. В. В. Браткевич. – Харків : ХНЕУ ім. С. Кузнеця, 2018. – 32 с.

П78

Наведено методи, засоби та завдання для організації самостійного вивчення і поглиблення отриманих у межах лекційного курсу і аудиторних лабораторних робіт компетентностей, знань, умінь та навичок. Подано перелік літератури, яка необхідна для виконання завдань.

Рекомендовано для студентів спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня всіх форм навчання.

УДК 004.032.6(07)

Вступ

Навчальна дисципліна "Програмування засобів мультимедіа" належить до професіонального циклу базових навчальних дисциплін. Її вивчають студенти напряму підготовки "Видавничо-поліграфічна справа" всіх форм навчання протягом третього та четвертого семестрів. Навчальна дисципліна потребує від студентів інтенсивної самостійної роботи зі спеціальною літературою та програмним забезпеченням у час, вільний від обов'язкових навчальних занять. До методичних рекомендацій вміщено теми, які не розглядалися раніше у методичних рекомендаціях до лабораторних робіт.

Метою самостійної роботи є поглиблення знань, які були отримані на лекційних заняттях, а також підтвердження і реалізація навичок, що були сформовані на лабораторних заняттях. Важливим завданням самостійної роботи є формування компетентностей, що дозволяють студенту реалізовувати на практиці отримані знання.

Студенти повинні розширити свої знання про технологічні засоби створення мультимедійних додатків, вивчити матеріали, наявні у мережі Інтернет за досліджуваним питанням, і виконати запропоновані завдання.

Основні види самостійної роботи, які запропоновані студентам із навчальної дисципліни:

вивчення лекційного матеріалу;

робота з вивчення рекомендованої літератури;

вивчення основних термінів та понять за темами дисципліни;

підготовка до лабораторних занять;

перевірка особистих знань за запитаннями для самостійного контролю та виконання контрольних завдань;

робота над індивідуальним завданням.

Для закріплення і перевірки набутих компетентностей під час самостійної роботи до кожної роботи пропонуються довідкові матеріали, практичні завдання і запитання для самодіагностики.

Перелік індивідуальних варіантів для виконання самостійної роботи вибирається з роздаткового матеріалу та узгоджується з викладачем.

Самостійна робота 1. Розроблення консольних програм, які реалізують обчислення лінійних процесів

Самостійна робота 1 охоплює поглиблене вивчення таких тем:

тема 1 "Теоретичні та методологічні засади організації програм і даних";

тема 2 "Поняття типу даних";

тема 3 "Програмування обчислювальних процесів".

Основну увагу під час поглибленого вивчення поточних тем приділяли наступним питанням: тема 1 – використанню середовища Visual Studio для створення консольних додатків; тема 2 – особливостям застосування вбудованих типів даних; тема 3 – розробленню консольних програм, які реалізують обчислення лінійного математичного виразу згідно з індивідуальним варіантом.

Мета роботи: отримання знань та навичок зі створення в програмному середовище Visual Studio консольних додатків, які реалізують обчислення лінійних процесів.

У результаті виконання самостійної роботи студенти набувають таких компетентностей: здатність створювати консольні Windows-додатки із застосування простих типів даних; уміння застосовувати середовище Visual Studio для налагодження консольних додатків.

Результатом виконання самостійної роботи є налаштований консольний додаток та демонстрація його працездатності.

Завдання для самостійної роботи

Виконувати дану самостійну роботу доцільно перед виконанням першої лабораторної роботи "Інтегроване середовище системи програмування Visual Studio.NET".

У ході виконання роботи необхідно:

1. Проаналізувати математичний вираз, який надалі буде реалізовано в середовищі Visual Studio.NET і намалювати алгоритм його обчислення у вигляді графічної схеми.

2. Обґрунтувати вибір типів даних та вбудованих у середовище розроблення стандартних математичних функцій, які повною мірою відповідають вихідному математичному виразу.

3. Згідно з алгоритмом обчислення та з урахуванням відповідних типів даних сформувати перелік послідовних команд, виконання яких приводить до отримання кінцевого результату.

4. У середовищі Visual Studio.NET створити шаблон консольного додатку та набрати у вбудований у нього редактор тексту розроблену лінійну програму.

5. Виконувати компіляцію, налагодження і запуск програми; отримати роздруківку вихідного тексту програми і результату її роботи.

Контрольні запитання для самодіагностики

1. Що означає .NET? У чому полягає особливість .NET-платформи?
2. Перерахуйте основні етапи розроблення програми, що виконується.
3. Навіщо необхідний етап компіляції?
4. У чому сутність процедурно-орієнтованого стилю програмування? Яка структура процедурно-орієнтованої програми?
5. Опишіть відомі вам алгоритмічні структури.
6. Дайте огляд основних операцій C#.
7. Навіщо потрібні логічні операції? Наведіть приклади.
8. Що таке пріоритети операцій? Де і коли вони використовуються?
9. Що таке асоціативність операцій? Де і коли вони використовуються?
10. Охарактеризуйте клас Math. Наведіть приклад програми, де використовуються вбудовані математичні методи.

Методичні рекомендації до теми

Самостійна робота повинна виконуватися в наступній послідовності.

1. Аналіз математичного виразу, обчислення якого надалі буде реалізовано в середовищі Visual Studio.NET, та обґрунтування вибору типів даних.

На цьому етапі необхідно зіставити змінні і константи вихідного математичного виразу зі стандартними даними, які вбудовані в середовище розробки. Наприклад, у разі таких вихідних виразів:

$B = \overline{a+1} * e^{x}$;

$C = (| z+2 | - 1) * x^{2} * B;$

$D = 1 + x^{5} * e^{(x+1)} + \log_{10}(1+x),$

поточні змінні і константи можуть бути надані в наступним чином:

```
const double a = 2.3;
double x,z,B,C,D;
```

2. Обґрунтувати вибору типів стандартних математичних функцій, які в повній мірі відповідають вихідному математичному виразу.

У будь-якій мові програмування повинні бути математичні функції, для обчислення: sin, cos, tg, ступеня і так далі. У мові Сі Шарп надається цілий клас математичних методів. Це клас Math. У цьому класі методи статичні. Для його підключення потрібно прописати в початку програми: using System. Для виклику методу, необхідно прописати: Math.Функція(). У класі Math є 25 методів математичних обчислень. Нижче наведено перелік операторів, які реалізують обчислення математичних виразів, що розглядаються у якості прикладу.

```
B = Math.Sqrt(a+1) * Math.Exp(x);
C =(Math.Abs(z+2)-1) * Math.Pow(x,2) * B;
D = Math.Pow(x,5 )* Math.Exp(x+1) + Math.Log10(1+x);
```

3. Формування переліку послідовних команд, виконання яких приводить до отримання кінцевого результату.

Цей етап доцільно виконувати безпосередньо в середовищі Visual Studio.NET. Для цього необхідно створити шаблон консольного додатка та набрати у вбудованому в нього редакторі тексту розроблену лінійну програму.

4. Компіляція, налагодження і запуск програми.

Вибрати пункт меню Debug / Start Without Debugging (або натиснути Ctrl + F5).

Для контролю правильності обчислень рекомендується за допомогою калькулятора заздалегідь підготувати кілька контрольних прикладів (з урахуванням особливих випадків) і порівняти відповідні результати.

Література: [1; 2].

Самостійна робота 2. Розроблення консольних програм, які реалізують обчислення циклічних процесів з розгалуженням

Самостійна робота 2 продовжує поглиблене вивчення теми 3: "Програмування обчислювальних процесів". Основну увагу під час поглибленого вивчення цієї теми приділяють наступним питанням: розробленню консольної програми, яка реалізує обчислення математичного виразу з розгалуженням згідно з індивідуальним варіантом; розробленню

консольної програми, яка реалізує створення таблиці обчислення значень математичного виразу згідно з індивідуальним варіантом.

Мета роботи: отримання знань та навичок зі створення в програмному середовище Visual Studio консольних додатків, які реалізують циклічні обчислення математичних виразів із розгалуженням.

У результаті виконання самостійної роботи студенти набувають таких компетентностей: застосовувати алгоритмічні структури управління програмою та варіанти їх реалізації мовою C#; здатність створювати багатострокові табличні документи.

Результатом виконання самостійної роботи є налаштований консольний додаток та демонстрація його працездатності.

Завдання для самостійної роботи

Виконувати дану самостійну роботу доцільно перед виконанням лабораторної роботи "Програмування циклічних обчислювальних процесів".

У ході виконання роботи необхідно:

1. Проаналізувати математичний вираз з можливістю вибору одної з трьох гілок обчислення, який надалі буде реалізовано в середовище Visual Studio.NET. Намалювати алгоритм його обчислення у вигляді графічної схеми. Алгоритм повинен забезпечувати формування таблиці з десяти рядків, в кожному з яких наведена інформація щодо обчислення заданої функції, її поточного аргументна та номера гілки.

2. За допомогою калькулятора підготувати контрольні приклади, які дозволяють перевірити дію алгоритму за кожною з гілок обчислення.

3. Відповідно до графічної схеми алгоритму скласти перелік команд, які забезпечують формування таблиці з результатами обчислення.

4. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється.

5. Виконати компіляцію, налагодження і запуск програми; отримати роздруківку вихідного тексту програми і результату її роботи.

Контрольні запитання для самодіагностики

1. Дайте визначення поняттю "поток управління програмою".

2. Що становить собою структура вибору if, коли її треба використовувати?

3. Опишіть структуру вибору if / else.

4. У чому полягає специфіка множинного вибору? Як цей вибір доцільно реалізувати за допомогою структури switch?

5. Наведіть синтаксис і відповідний приклад застосування умовного виразу.

6. Дайте загальний огляд структур повторення.

7. Опишіть особливості застосування циклу з передумовою (while). Наведіть приклад.

8. Коли доцільно застосовувати цикл з постумовою (do / while)? Наведіть приклад.

9. Дайте синтаксис оператора циклу for.

10. Наведіть загальні рекомендації відносно вибору циклів.

Методичні рекомендації до теми

Самостійна робота повинна виконуватися в наступній послідовності.

1. Аналіз математичного виразу, обчислення якого надалі буде реалізовано в середовищі Visual Studio.NET.

Наприклад, необхідно обчислити і вивести на екран у вигляді таблиці значення функції F на інтервалі від $X_{почат}$ до $X_{кінц}$ з кроком dX

$$F = \begin{cases} a*x^2 + b / c, & \text{при } x < 1 \text{ і } c \neq 0, \\ (x - a) / (x - c)^2, & \text{при } x > 1{,}5 \text{ і } c = 0, \\ a / x^2, & \text{в інших випадках,} \end{cases}$$

де a, b, c – дійсні числа. Причому, функція F повинна приймати дійсне значення, якщо вираз ($A_Ц$ & $B_Ц$) МОД2 $C_Ц$ не дорівнює нулю, і ціле значення в іншому випадку. Через $A_Ц$, $B_Ц$ та $C_Ц$ позначені цілі частини значень a, b, c, операції & і МОД2 (додавання по модулю 2) – порозрядні.

Як результат виконання цього пункту, – необхідно отримати набір чисельних прикладів, який охоплює дію алгоритму обчислення по кожній із трьох гілок вихідного виразу. Крім цього, слід передбачити вихідні дані, які приводять до особливих випадків (наприклад, ділення на нуль).

2. Побудова графічної схеми алгоритму обчислення і написання коду програми.

На даному етапі необхідно обрати дві графічні алгоритмічні схеми – одну для зображення процесу що розгалуження на відповідні гілки, а другу – для організації циклічного формування строк кінцевої таблиці.

Якщо для розгалуження обрана графічна схема з двома гілками (їй відповідає оператора if – else), а для циклічного повтору – схема циклу

з передумовою (йому відповідає оператора for) то фрагмент коду, який забезпечує обчислення функції F, може бути таким:

```
…
double a, b, c, x0, xN, dx, F;
for (double i = x0; i <= xN; i = i + dx)
        {
           if (i < 1 && c != 0)
           {
              F = a * Math.Pow(i, 2) + b / c;
           }
           else if (i > 1.5 && c == 0)
           {
              F = (i - a) / Math.Pow(i - c, 2);
           else
           {
              F =  a / Math.Pow(i, 2);
           }
           if (((A & B) ^ C) != 0)
           {
              double F1 = F;
              Console.WriteLine(F1);
           }
           else
           {
              int F2 = Convert.ToInt32(F);
              Console.WriteLine(F2);
           }
…
```

3. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється.

4. Виконати компіляцію, налагодження і запуск програми; отримати роздруківку вихідного тексту програми і результату її роботи.

Література: [1; 4].

Самостійна робота 3. Розроблення консольних програм, які реалізують оброблення масивів

Самостійна робота 3 присвячена поглибленому вивченню матеріалу теми 4 "Масиви". Основна увага при поглибленому вивчені цієї теми

приділялась питанню розроблення консольної програми, яка реалізує оброблення масиву простих типів даних згідно із індивідуальним варіантом.

Мета роботи: отримання знань та навичок зі створення в програмному середовище Visual Studio консольних додатків, які реалізують оброблення масивів простих типів даних.

У результаті виконання самостійної роботи студенти набувають таких компетентностей: знання типових алгоритмів оброблення масивів і здатність застосовувати оператори управління мови C# для їх оброблення.

Результатом виконання самостійної роботи є налаштований консольний додаток та демонстрація його працездатності.

Завдання для самостійної роботи

Виконувати дану самостійну роботу доцільно перед виконанням лабораторної роботи "Оброблення одномірних масивів і матриць".

У ході виконання роботи необхідно:

1. Підготувати чисельні контрольні приклади початкового масиву та результати його оброблення згідно з індивідуальним варіантом.

2. Надати словесний опис дії алгоритму та на цій основі розробити його графічну схему (блок-схему).

3. Обґрунтувати вибір операторів мови C#, які найкращим чином відповідають реалізації графічної схеми, та написати відповідний код програми оброблення масиву.

4. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється.

5. Відкомпілювати текст програми, усуваючи в разі потреби помилки.

6. Дослідити роботу програми, аналізуючи виконання контрольних чисельних прикладів.

Контрольні запитання для самодіагностики

1. У чому полягають особливості призначення, оголошення й визначення масиву?

2. Як відтворюється доступ до окремих елементів масиву?

3. Наведіть приклади варіантів ініціалізації масиву.

4. Опишіть загальну схему перебору елементів масиву за допомогою оператора foreach.

5. Наведіть приклади алгоритмів пошуку заданих елементів масиву.

6. Наведіть приклади алгоритмів перетворення масиву.

7. У чому полягає сутність алгоритму сортування елементів масиву методом "Пузирик"?

8. Як реалізується доступ до елементів двовимірного масиву?

9. У чому полягає сутність подання двовимірного масиву як масиву масивів?

10. Наведіть приклади оброблення матриць.

Методичні рекомендації до теми

Самостійна робота повинна виконуватися в наступній послідовності.

1. Аналіз індивідуального завдання, підготовка відповідного завдання чисельного початкового прикладу масиву та результатів його оброблення.

Наприклад, необхідно написати програму, яка в одновимірному масиві, що складається з 10 елементів типу double, обчислює суму елементів, розташованих між першим третім нульовими елементами. У якості початкового масиву потрібно надати масив, який включає в себе три або більше нульових елементів. Також на цьому етапі доцільно визначити ім'я змінних, які надалі будуть використовуватися в програмі, наприклад такі:

```
double m[10 ];  // ім'я поточного масиву
int cont_zero = 0;   // лічильник нульових елементів масиву
double sum_zeroPrimero_zeroTercero = 0; // сума, яку треба обчислити
int ind_zeroPrimero=0;  // індекс першого нульового елементу
int ind_zeroTercero=0;  // індекс третього нульового елементу
```

2. Надати словесний опис дії алгоритму та на цій основі розробити його графічну схему (блок-схему).

Під час опису алгоритму, необхідно використовуючи ключові фрази типу: "... початковий стан змінних перед початком циклу ... ", "... на кожній ітерації проходу масиву виконуються наступні дії ... ", "... як результат отримуємо ... ".

Графічна схема алгоритму повинна містить виключно стандарті блоки [2] зображення відповідних елементів.

3. Обґрунтувати вибір операторів мови C#, які найкращим чином відповідають реалізації графічної схеми, та написати відповідний код програми оброблення масиву.

У якості продовження прикладу, нижче наведено фрагмент програми, який знаходить індекси першого та третього нульових елементів масиву, після чого обчислює суму елементів проміж ними:

```
...
 for(int i=0; i<size; i++)
  {
    if(m[i]==0)
       ++cont_zero;
    if(cont_zero==1 && ind_zeroPrimero==0)
       ind_zeroPrimero=i;
    if(cont_zero==3 && ind_zeroTercero==0)
       ind_zeroTercero =i;
  }
 for(int i=ind_zeroPrimero+1; i<ind_zeroTercero; i++)
        sum_zeroPrimero_zeroTercero +=m[i];
...
```

4. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється.

5. Відкомпілювати текст програми, усуваючи в разі потреби помилки.

Особу увагу потрібно надати помилкам, коли значення поточного індексу масиву виходять за межі діапазону.

6. Дослідити роботу програми, аналізуючи виконання контрольних чисельних прикладів.

Література: [1; 4].

Самостійна робота 4. Розроблення консольних програм, які реалізують оброблення структур

Самостійна робота 4 присвячена поглибленому вивченню матеріалу теми 5 "Структури". Основна увага при поглибленому вивчені цієї теми приділялась питанню розроблення консольної програми, яка реалізує оброблення масиву структур згідно із індивідуальним варіантом.

Мета роботи: отримання знань та навичок зі створення в програмному середовище Visual Studio консольних додатків, які реалізують оброблення структур та їх масивів.

У результаті виконання самостійної роботи студенти набувають таких компетентностей: знання типових алгоритмів створення, ініціалізації та оброблення масивів структур, а також і здатність застосовувати оператори управління мови C# для оброблення даних типу struct.

Результатом виконання самостійної роботи є налаштований консольний додаток та демонстрація його працездатності.

Завдання для самостійної роботи

Виконувати дану самостійну роботу доцільно перед виконанням лабораторної роботи "Оброблення структур".

У ході виконання роботи необхідно:

1. Проаналізувати табличний документ, варіант якого взяти з додаткового фала з індивідуальними завданнями. Як результат – створити опис екземпляру структури з полями, які відповідають найменуванням колонок таблиці.

2. Надати словесний опис дії алгоритму створення та оброблення табличного документу і на цій основі розробити його графічну схему (блок-схему).

3. Обґрунтувати вибір операторів мови C#, які найкращим чином відповідають реалізації графічної схеми, та написати відповідний код програми оброблення масиву структур.

4. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється.

5. Відкомпілювати текст програми, усуваючи в разі потреби помилки.

6. Дослідити роботу програми, аналізуючи виконання контрольних прикладів.

Контрольні запитання для самодіагностики

1. Коли доцільно використовувати структури?

2. Як реалізується призначення, оголошення й визначення структур?

3. Наведіть приклади оголошення, визначення, ініціалізації та виведення на екран елементів заданої структури (без застосування операції **new)**.

4. Наведіть приклади оголошення, визначення, ініціалізації та виведення на екран елементів заданої структури із застосуванням операції **new**.

5. Які особливості оброблення елементів структур?

6. Які особливості оброблення масивів структур?

Методичні рекомендації до теми

Самостійна робота повинна виконуватися в наступній послідовності.

1. Аналіз індивідуального завдання, підготовка макета табличного документа та чисельних прикладів заповнення відповідних клітинок таблиці згідно з похідними формулами обчислення.

Наприклад, якщо потрібно розробити програму для формування відомості про вартість виданих книг, то один з варіантів чисельного прикладу подібної відомості може мати вид, який надано в табл. 1.

2. Створити опис екземпляру структури з полями, які відповідають найменуванням колонок таблиці.

Таблиця 1

Відомість вартості виданих книг

n/n	Автор	Вартість	Видано	Витрати
1	Рубіна	34,45	3	103,35
2	Улицька	45,78	10	457,80
3	Пушкін	75,23	3	225,69
Разом:		155,46	16	786,84

З табл. 1 витікає, що структура повинна мати чотири поля, кожному з яких потрібно надати ім'я та зіставити найбільш зручний для оброблення відповідний тип. Якщо позначити структуру як Stroka, то її опис може виглядати так:

```
struct Stroka
{
        public string name;                 // Автор книги
        public double stoimost;             // Вартість виданої книги
        public int kolich;                  // Кількість виданих книг
        public double sum_stoimost;         // Вартість виданих книг
};
```

2. Надати словесний опис дії алгоритму створення та оброблення табличного документа і на цій основі розробити його графічну схему (блок-схему).

На цьому етапі потрібно виділити наступні кроки алгоритму: 1) організація діалогу з користувачем для заповнення колонок "Автор", "Вартість", "Видано"; 2) виконання обробки попереднього вводу даних, та формування змісту колонки "Витрати" та відповідних підсумкових значень; 2) формування заголовку (шапки) таблиці; 4) порядкове виведення раніш обчислювальних даних; 5) формування рядка "Разом".

3. Обґрунтувати вибір операторів мови C#, які найкращим чином відповідають реалізації графічної схеми, та написати відповідний код програми оброблення масиву екземплярів структури Stroca.

У якості продовження прикладу, нижче наведено фрагмент програми, який реалізує перший та другий кроки алгоритму оброблення масиву екземплярів структури Stroca:

```
…
int kol = Convert.ToInt32(Console.ReadLine());
                Stroka[ ] Tabl = new Stroka[kol];
                for( int i=0; i < Tabl.Length; i++)
                {
                        Console.WriteLine("Автор книги?");
                        Tabl[i].name = Console.ReadLine();
                        Console.WriteLine("Вартість книги?");
                Tabl[i].stoimost = Convert.ToDouble(Console.ReadLine());
                        Console.WriteLine("Кількість книг?");
                        Tabl[i].kolich = Convert.ToInt32(Console.ReadLine());
                }
                //  Виконання розрахунків:
                double s1=0, s2=0, s3=0;
                for( int i=0; i < Tabl.Length; i++)
                {
                        Tabl[i].sum_stoimost = Tabl[i].stoimost * Tabl[i].kolich;
                        s1 += Tabl[i].stoimost;
                        s2 += Tabl[i].kolich;
                        s3 += Tabl[i].sum_stoimost;
                }
...
```

4. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється.

5. Відкомпілювати текст програми, усуваючи у разі необхідності помилки.

6. Дослідити роботу програми, аналізуючи виконання контрольних прикладів.

Література: [1; 3].

Самостійна робота 5. Розробка консольних програм з використанням функцій

Самостійна робота 8 присвячена поглибленому вивченню матеріалу теми 6 "Функції". Основна увага при поглибленому вивчені цієї теми приділялась питанню розроблення консольної програми, яка реалізує згідно з індивідуальним варіантом оброблення масивів різноманітних даних із застосуванням певних функцій.

Мета роботи: отримання знань та навичок щодо розробки функцій користувача та їх застосування для оброблення різноманітних даних.

У результаті виконання самостійної роботи студенти набувають таких **компетентностей**: здатність розробляти програми із застосування стандартних функцій та функцій користувача; знання механізми опису і виклику функцій та способів обміну інформацією з функцією.

Результатом виконання самостійної роботи є налаштований консольний додаток та демонстрація його працездатності.

Завдання для самостійної роботи

Виконувати дану самостійну роботу доцільно перед виконанням лабораторної роботи "Використання функцій".

У ході виконання роботи необхідно:

1. Проаналізувати зміст індивідуального завдання і виділити ті пункти завдання, які можуть бути реалізовані у вигляді окремих процедур – функцій.

2. Підготувати чисельні контрольні приклади початкового масиву та результати його оброблення згідно з індивідуальним варіантом.

3. Для кожної функції надати словесний опис дії алгоритму, який лежить в основі її роботи, та на цій основі розробити загальну графічну схему (блок-схему) опису алгоритму для усього додатку.

4. Обґрунтувати вибір операторів мови C#, які найкращим чином співвідносяться зі графічними схемами для кожної з функцій, та написати відповідний код опису та виклику функцій.

5. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється.

6. Відкомпілювати текст програми, усуваючи у разі потреби помилки.

7. Дослідити роботу програми, аналізуючи виконання контрольних чисельних прикладів.

Контрольні запитання для самодіагностики

1. Дайте визначення функції. Чому функція може слугувати прикладом коду, який повторно виконується?

2. Перерахуйте основні етапи виконання функції.

3. Який синтаксис запису значень, що повертаються з функції?

4. Охарактеризуйте параметри функцій. У чому полягає відповідність параметрів?

5. У чому полягають основні ідеї реалізації процесу обміну інформацією з функцією.

6. Що таке область дії змінних? Охарактеризуйте параметри і значення, що повертаються, за порівнянням із глобальними даними.

7. У чому полягають особливості передачі параметрів за посиланням і за значенням.

10. Які особливості використання функції і масиву?

Методичні рекомендації до теми

У якості прикладу розглянемо завдання де потрібно знайти у вихідному масиві цілих чисел суму елементів, значення яких менше нуля. Самостійну роботу слід виконувати в наступній послідовності.

1. Аналіз змісту індивідуального завдання і виділення пунктів завдання, які можуть бути реалізовані у вигляді окремих процедур – функцій.

Як правило, оброблення масиву випереджає його контрольний друк, також ця процедура дуже доцільна, якщо масив повинен обробляться і значення його елементів змінюються, або змінюється їх порядок (наприклад, після сортування). Друга процедура, яка витікає із завдання – це формування суми елементів, значення яких менше нуля. Надаємо цим процедурам відповідні назви: pech_mas( ) та sum_negativ( ).

2. Підготовка відповідного завданню чисельного початкового прикладу масиву та результатів його оброблення.

У якості початкового масиву потрібно надати масив, який включає в себе три або більше нульових елементів.

3. Словесний опис і побудова відповідних графічних схем виконується аналогічно пункту 2 методичних рекомендацій до самостійної роботи 3. Крім двох блок-схем опису функцій, також необхідно навести загальну блок-схему, де кожний функції відповідає її стисле графічне зображення у вигляді прямокутника.

4. Розроблення коду відповідних функцій.

На цьому етапі необхідно записати сигнатуру функції і код її тіла. Сигнатура містить ім'я функції, після якого в круглих дужках вказуються параметри функції, а також поперед ім'ям – значення, що повертається. Наприклад, визначення функцій **pech_mas** та **sum_negativ**, може мати такий вид:

```
static void pech_mas(int[ ] m)
        {
            for (int i = 0; i < m.Length;  i++)
                Console.Write("{0} ",m[i]);
            Console.WriteLine(); Console.WriteLine();
        }
static void sum_negativ[ ] m)
        {
            double sum_otr = 0.0;
            for(int i=0; i<m.Length; i++)
                if(m[i]<0)
                    sum_ negativ += m[i];
            Console.WriteLine( "sum_ negativ = {0}", sum_ negativ);
        }
```

Ключове слово **static** означає, що функція статична і до цієї функції можна звернутися з будь-якого місця без створення екземпляра об'єкта класу.

5. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється. Наприклад, таким чином:

```
using System;
    class Class1
{
    // місце для визначення функцій pech_mas  та sum_negativ
    static void Main(string[ ] args)
    {
        int[ ] mas = {-1,5,2,3,5,0,3,9,2,0,1,6,0,-3,2};
        pech_mas(mas);
        sum_ negativ (mas);
        Console.WriteLine();
    }
}
```

Слід звернути увагу на оформлення виклику функцій. У розглянутому прикладі обидві функції мають значення **void** типу, що повертаються і, отже, немає необхідності в явному вигляді (за допомогою операції привласнення) запам'ятовувати результати їх виконання.

6. Відкомпілювати текст програми, усуваючи в разі потреби помилки, і дослідити її роботу, аналізуючи виконання контрольних чисельних прикладів.

Література: [1; 3; 5].

Самостійна робота 6. Розроблення консольних програм з використанням об'єктно-орієнтованого стилю програмування (ООП)

Самостійна робота 8 присвячена поглибленому вивченню матеріалу теми 7 "Особливості об'єктно-орієнтованого стилю програмування", теми 8 "Методи класів" та теми 9 "Відносини між класами". Основну увагу під час поглибленого вивчення цих тем приділяють питанням розроблення консольних програм з ієрархією класів, які створюють певний клас із властивостями і методами та здійснюють оброблення екземплярів даного класу.

Мета роботи: отримання знань та навичок зі створення в програмному середовище Visual Studio об'єктно-орієнтованих додатків, які реалізують оброблення багатомірних масивів даних.

У результаті виконання самостійної роботи студенти набувають таких компетентностей: здатність використовувати об'єктно-орієнтований стиль програмування для реалізації певних сценаріїв замовника; здатність написати програму, яка складається з базового класу і створеного з нього двох-трьох похідних класів, а також здійснює елементарне оброблення і виведення інформації про властивості та результати оброблення щодо окремих об'єктів успадкованого класу.

Результатом виконання самостійної роботи є налаштований об'єктно-орієнтований консольний додаток та демонстрація його працездатності.

Завдання для самостійної роботи

Виконувати дану самостійну роботу доцільно перед виконанням лабораторної роботи "Розроблення програм з ієрархією класів".

У ході виконання роботи необхідно:

1. Підготувати чисельні контрольні приклади початкової матриці та результати її оброблення згідно з індивідуальним варіантом.

2. Надати словесний опис дії алгоритму оброблення матриці та на цій основі виділити клас, який інкапсулює: сукупність методів, що реалізують окремі пункти оброблення матриці; призначений для користувача конструктор із параметрами; відповідні властивості елементів класу.

3. Написати код класу користувача, який відповідає словесному опису, і містить конструктор, методи та властивості.

4. Написати код основної програми, що містить два класи: стандартній із методом **main()** і клас користувача. Програма повинна забезпечувати створення екземпляру класу користувача та виклик відповідних методів екземпляру даного класу.

5. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється.

6. Відкомпілювати текст програми, усуваючи в разі потреби помилки. Дослідити роботу програми, аналізуючи виконання контрольних чисельних прикладів.

Контрольні запитання для самодіагностики

1. У чому полягає сутність парадигми ООП?

2. Дайте визначення об'єкта.

3. Наведіть основні принципи ООП.

4. Наведіть приклади мультимедійних сценаріїв, де застосовуються абстрагування, інкапсуляція і спадкування.

5. Як здійснюється доступ до методів?

6. Що таке статичні поля, який їх зв'язок із методами класу?

7. Призначення конструкторів і деструкторів.

8. Як здійснюється обмін інформацією між методами?

9. Назвіть можливі типи відносин між класами.

10. Наведіть приклад простої ієрархії класів.

11. Як здійснюється доступ до елементів класу в разі наявності спадкоємства?

12. Дайте визначення абстрактних та віртуальних класів.

Методичні рекомендації до теми

У якості прикладу розглянемо завдання де потрібно: 1) обчислити в вихідному двомірному масиві цілих чисел (матриці) суму елементів, які знаходяться на кожному рядку та на кожному стовбці; 2) знайти номер рядка і номер стовпця, на пересіченні яких знаходиться число раніше введене з клавіатури. Самостійна робота повинна виконуватися в наступній послідовності.

1. Підготовка відповідного завданню чисельного прикладу матриці та результатів її оброблення. Один із варіантів організації інтерфейсу користувача може бути таким:

```
Please input the dimensions of the matrix:
5
12

Base matrix
6  2  1  4  7  7  3  1  1  4  4  4
2  6  1  6  3  4  3  7  8  3  2  4
1  1  3  4  8  7  8  2  6  1  3  1
1  6  5  3  1  3  1  6  2  5  3  3
1  8  8  3  8  4  0  2  0  5  2  5

Sum_Row
44
49
45
39
46

Sum_Column
11  23  18  20  27  25  15  18  17  18  14  17

Please input any integer = 0
Coordinates 0 => i = 5  j = 7
```

2. Написати код класу користувача, який відповідає словесному опису, і містить конструктор, методи та властивості.

Перед розробленням коду класу користувача (наприклад, з ім'ям **matriz**) необхідно з постановки завдання виділити перелік окремих задач, кожна з яких може реалізуватися у вигляді окремого методу.

Один із варіантів такого переліку може бути таким: **fill ()** – метод заповнення матриці випадковими числами; **show( )** – метод виведення матриці на екран монітору; **summR()** – метод обчислення суми елементів

матриці по строкам; **summC( )** – метод обчислення суми елементів матриці по стовбцям; **find(int v, out int ii, out int jj)** – метод пошуку заданого з клавіатури першого значення в матриці.

Нижче наведено код, який потрібно розмістити одразу після директиви **using System**. Тобто перед класом **class Program**, ім'я якого співпадає з ім'ям консольного шаблону програми.

```
using System;
class Matrix
  {
    Int [, ] matrix;

    public Matrix(int m, int n)
    {
      matrix = new int[m, n];
    }

    public void fill()   // Метод заповнення матриці випадковими числами
    {
      Random random = new Random();
      for (int i = 0; i < matrix.GetLength(0); i++)
      {
        for (int j = 0; j < matrix.GetLength(1); j++)
        {
          matrix[i,j] = random.Next(0,9);                    // От 0 - 9
        }
      }
    }

    public void show()    // Метод виведення матриці на екран монітора
    {
      for (int i = 0; i < matrix.GetLength(0); i++)
      {
        for (int j = 0; j < matrix.GetLength(1); j++)
        {
          Console.Write(matrix[i, j] + "   ");
        }
        Console.WriteLine();
      }
    }

    public Matrix summR()   // Метод обчислення суми елементів матриці за
                            //  рядками (ROW)
    {
```

```
    Matrix result = new Matrix(matrix.GetLength(0), 1);
    for (int i = 0; i < matrix.GetLength(0); i++)
    {
        result.matrix[i, 0] = 0;
        for (int j = 0; j < matrix.GetLength(1); j++)
        {
            result.matrix[i, 0] += matrix[i, j];
        }
    }
    return result;
  }

  public Matrix summC()   // ()   // Метод обчислення суми елементів матриці за
                                  //  стовбцями (COLUMN)
  {
    Matrix result = new Matrix(1, matrix.GetLength(1));
    for (int j = 0; j < matrix.GetLength(1); j++)
    {
        result.matrix[0, j] = 0;
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            result.matrix[0, j] += matrix[i, j];
        }
    }
    return result;
  }
  public void find(int v, out int ii, out int jj)     // Метод пошуку заданого (value)
  // першого значення в матриці з використанням OUT
        {
            for (int i = 0; i < matrix.GetLength(0); i++)
               {
                   for (int j = 0; j < matrix.GetLength(1); j++)
                      {
                          if (matrix[i, j] == v)
                             {
                                 ii = i;
                                 jj = j;
                                 return;
                             }
                      }
               }
            ii = jj = -1;
        }
} //  end class Matrix
```

3. Написати код, який забезпечує створення екземпляру класу **Matrix** (наприклад, з ім'ям **m**), та виклик відповідних методів, які реалізують індивідуальне завдання. Код потрібно розмістити у тілі функції **Main**.

```
class Program
{
   static void Main(string[] args)
   {
      Console.WriteLine("Please input the dimensions of the matrix: ");
      Matrix m = new Matrix(Convert.ToInt32(Console.ReadLine()),
Convert.ToInt32(Console.ReadLine()));
      m.fill();
      Console.WriteLine("\nBase matrix");
      m.show();

      Console.WriteLine("\nSum_Row");
      m.summR().show();
      Console.WriteLine("\nSum_Column");
      m.summC().show();

      Console.Write("\nPlease input any integer = ");
      int i, j, num = Convert.ToInt32(Console.ReadLine());
      m.find(num, out i, out j);
      Console.Write("Coordinates {0} => i = {1}\tj = {2}", num, i+1, j+1);

      Console.ReadLine();
   } // end Main
} // end class Program
```

4. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати (у вбудований у нього редактор тексту) програму, що розробляється.

5. Відкомпілювати текст програми, усуваючи у разі потреби помилки. Дослідити роботу програми, аналізуючи виконання контрольних чисельних прикладів.

Література: [1; 6].

Самостійна робота 7. Розроблення мультимедійних програм

Самостійна робота 7 присвячена поглибленому вивченню матеріалу теми 10 "Принципи створення візуальних інтерфейсів" та теми 11 "Програмування графіки". Основну увагу під час поглибленого вивчення цих тем

приділяють питанням розроблення програм, які згідно з індивідуальним варіантом створюють: типовий каркас графічного Windows-додатка; MDI- і SDI-додатки за допомогою компонентів Designer Forms; Windows-додаток з елементами графіки; мультимедійні Windows-додатки з елементами звука, відео та анімації.

Мета роботи: отримання знань та навичок зі створення в програмному середовище Visual Studio графічних Windows MDI- і SDI-додатків з елементами звука, відео та анімації.

У результаті виконання самостійної роботи студенти набувають таких компетентностей: здатність здійснювати розроблення об'єктно-орієнтованих додатків, в яких застосовуються: методи форми, діалогові вікна, немодальні вікна, багатодокументний інтерфейс, компоненти .NET, шаблони, колекції; здатність розробляти графічні додатки зі звуком, відео та анімацією.

Результатом виконання самостійної роботи є налаштований об'єктно-орієнтований графічний додаток з анімацією та демонстрація його працездатності.

Завдання для самостійної роботи

Виконувати дану самостійну роботу доцільно перед виконанням лабораторної роботи "Розроблення Windows-додатків з елементами графіки".

У ході виконання роботи необхідно:

1. Підготувати рисунок, який буде анімовано.
2. Надати словесний опис дії алгоритму формування рисунка на початкової формі.
3. Надати словесний опис дії алгоритму анімації рисунка.
4. Написати код, який реалізує алгоритми формування та ї анімації рисунка.
5. У середовищі Visual Studio.NET створити шаблон консольного додатка та набрати у відповідні обробники подій код, який забезпечує формування рисунку та його анімацію.

Контрольні запитання для самодіагностики

1. Що розуміють під поняттям "технологія GDI+"?
2. Як створюється поверхня малювання Graphics?
3. Які графічні структури застосовуються в GDI+?
4. Наведіть приклад синтаксису застосування пір'я Pen.
5. У чому полягає сутність трансформацій графічних об'єктів?

6. Опішить способи малювання прямих ліній.

7. Опішить способи малювання геометричних примітивів.

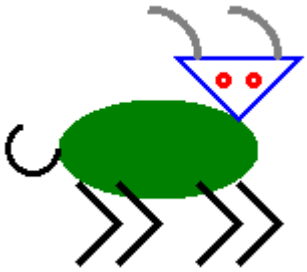
8. Як додати анімаційній ефекти в графічний додаток?

9. Що таке крива Безьє? Як вона формується и коли її доцільно застосовувати?

10. Опішить алгоритм роботи з картинками.

Методичні рекомендації до теми

У якості прикладу розглянемо завдання де потрібно: 1) створити графічний додаток, під час запуску якого з'являється наступне зображення ягня:



2) при подвійному кліку по формі ягня повинно танцювати, тобто переміщатися по формі, імітуючи танець.

Малювання за допомогою GDI + дуже нагадує звичайне малювання. Графіка виводиться на полотні (клас Graphics) пір'ям (клас Pen), кистями (клас Brush) і шрифтами (клас Font). Основний клас для малювання це клас Graphics. Полотно повинно належати до конкретного вікна – головній формі або якомусь елементу управління форми.

Windows є подієво-керованою операційною системою, тому будь-яка дія в програмі є обробленням тієї чи іншої події. Малювати слід під час виникання події Paint. Ця подія виникає кожен раз коли Windows виявляє, що поверхня вікна слід оновити (під час створення вікна, а також після перекриття вікна іншими вікнами). У разі виклика методу, який виконує оброблення події Paint, через параметри буде передано посилання на екземпляр класу Graphics, на якому слід малювати.

Самостійна робота повинна виконуватися в наступній послідовності.

1. Створіть новий проект "Lamb" (шаблон C# Windows Forms Application).

2. Встановіть розміри форми – властивість Size = 600; 480.

3. Розмістить на цій формі елемент Panel і встановіть у властивостях цієї панелі: BackColor = White, а поле Dock = Fill. У результаті панель повинна стати білого кольору і зайняти всю площину екрану.

4. Малювання на формі та анімація зображення. Для цього у властивостях панелі перейти до списку подій (натиснуть квадратну кнопку з блискавкою). Знайти в списку подію Paint та написати напроти цієї події onPaint і натиснуть Enter. Студія створить шаблон обробника цієї події і запропонує написати для нього код. Необхідно написати наступне:

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Lamb
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      float x0 = 230;
      float y0 = 180;
      float t = 0;

      private void onPaint(object sender, PaintEventArgs e)
      {
         Graphics g = e.Graphics; // створення екземпляра (g) класу Graphics
         DrawSheep(g, x0, y0, t); // малювання ягня
      }

      void DrawSheep(Graphics g, float x0, float y0, float t)
      {
         Pen p1 = new Pen(Color.Blue, 2); // заготовляємо пір'я і кисті
         Pen p2 = new Pen(Color.Red, 3);  // для малювання ягня
         Pen p3 = new Pen(Color.Gray, 4);
         Pen p4 = new Pen(Color.Black, 3);
         SolidBrush b1 = new SolidBrush(Color.Green);

         g.FillEllipse(b1, x0, y0, 100, 50); // туловище

         float xhead = x0 + 60; // обчислюємо координати голови
         float yhead = y0 - 20;
         g.DrawPolygon(p1, new PointF[] { new PointF(xhead, yhead), new
PointF(xhead + 30, yhead + 30),
            new PointF(xhead + 60, yhead) }); // малюємо голову
```

```
            g.DrawEllipse(p2, x0 + 80, y0 - 12, 5, 5); // малюємо глаза
            g.DrawEllipse(p2, x0 + 95, y0 - 12, 5, 5);

            g.DrawArc(p3, x0 + 20, y0 - 45, 50, 50, 0, -90); // малюємо роги
            g.DrawArc(p3, x0 + 60, y0 - 45, 50, 50, 0, -90);

            float xlegs = x0 + 10; // обчислюємо координати ніг
            float ylegs = y0 + 42; // и малюємо чотири ноги

            g.DrawLines(p4, new PointF[ ] { new PointF(xlegs, ylegs), new PointF(xlegs +
20 * (1 - t), ylegs + 20 + 20 * t), new PointF(xlegs, ylegs + 40 + 40 * t) });
            xlegs = xlegs + 20;
            g.DrawLines(p4, new PointF[ ] { new PointF(xlegs, ylegs), new PointF(xlegs +
20 * (1 - t), ylegs + 20 + 20 * t), new PointF(xlegs, ylegs + 40 + 40 * t) });
            xlegs = xlegs + 40;
            g.DrawLines(p4, new PointF[ ] { new PointF(xlegs, ylegs), new PointF(xlegs +
20 * (1 - t), ylegs + 20 + 20 * t), new PointF(xlegs, ylegs + 40 + 40 * t) });
            xlegs = xlegs + 20;
            g.DrawLines(p4, new PointF[ ] { new PointF(xlegs, ylegs), new PointF(xlegs +
20 * (1 - t), ylegs + 20 + 20 * t), new PointF(xlegs, ylegs + 40 + 40 * t) });

            g.DrawArc(p4, x0 - 25, y0 + 12, 25, 25, 0, 245); // малюємо хвіст
        }

        float dt = 0.01f;

        private void onTick(object sender, EventArgs e)
        {
            t = t + dt;
            if ((t < -0.3f)||(t > 0.7f))
            {
                dt = -dt;
            }

            x0 = 230 + (float)(100 * Math.Sin(10 * t));
            y0 = 180 - (float)(20 * Math.Sin(30 * t));

            panel1.Invalidate();
        }

        private void onDClick(object sender, EventArgs e)
        {
            timer1.Enabled = true;
        }
    }
      }
```

Малювання ягня винесено в окрему функцію DrawLamb, яка має в якості параметрів (клас Graphics), координати ягня і ще один параметр t, який задає вигин ніг ягня для зображення танцю.

Малювання полягає в тому, що викликаються різні методи класу Graphics, які описують графічні примітиви. GDI+ використовує традиційну для машинної графіки систему координат: початок координат – верхній лівий кут, вісь X зростає зліва на право, а вісь Y – зверху вниз. Більшість функцій GDI+ можуть працювати як з цілими координатами, так і з типом float, округляючи автоматично до цілих пікселів.

Сутність анімації полягає в змінювані з одного боку положення ягня, а з іншого вигину лап. Вигин лап у ягня визначається параметром t із відрізка [−0.3,0.7], а положення барана обчислюється залежно від параметра t.

Потрібно, щоб хтось регулярно викликав метод для обчислення координат і перемальовування ягня. Для цих цілей є невізуальний компонент Timer. Для його підключення необхідно перейти в дизайнер форми і в наборі компонентів знайти "годинник" – Timer. Далі – перетягнути його на форму. Він розташується під формою. Натиснути на нього правою клавішею і у властивостях встановити значення Interval = 60. Властивість Interval встановлює через скільки мілісекунд повинен бути "Tick" годин. Значення, яке дорівнює 1 000, відповідає одній секунді. Слід враховувати, що інтервал "цокання" є відносно приблизними, і може варіюватися залежно від потужності комп'ютера і його поточного завантаження. Список подій таймера складається з однієї події Tick.

Після обчислення нових координат, викликається метод Invalidate() – він примусово ініціює для панелі подію Paint, що змушує перемалювати ягня.

Далі потрібно запустити таймер. Для цього необхідно відкрити дизайнер форми, перейти до списку подій "panel1" і для події DoubleClick написати "onDClick" і натиснуть Enter. У шаблон, який з'явився, потрібно ввести відповідний код (timer1.Enabled = true;).

Після запуску додатка та його компіляції з'являється форма із зображенням ягня, а після подвійного кліку по формі ягня повинно танцювати.

Завдання. Доповніть код методу DrawLamb(), операторами, які забезпечують хитання голови ягня.

Література: [1; 4; 7].

Рекомендована література

1. Браткевич В. В. Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни "Основи програмування" для студентів напряму підготовки 6.051501 "Видавничо-поліграфічна справа" всіх форм навчання / уклад. В. В. Браткевич. – Харків : Вид. ХНЕУ, 2015. – 118 с.

2. Гаврилов В. П. Основи програмування : конспект лекцій для студентів напряму підготовки 0927 "Видавничо-поліграфічна справа" усіх форм навчання / уклад. В. П. Гаврилов, В. В. Браткевич, І. О. Бондар. – Харків : Вид. ХНЕУ, 2007. – 172 с.

3. Лабор В. В. Си Шарп. Создание приложений для Windows / В. В. Лабор – Минск : Харвест, 2011 – 384 с.

4. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах / Ч. Петцольд : пер. с англ. – Москва : Издательско-торговый дом "Русская Редакция", 2009. – 576 с.

5. Программист – программисту. C#. / Карли Ватсон и др. : пер. с англ. – Москва : Издательство "Лори", 2010. – 862 с.

6. Робинсон У. C# без лишних слов / У. Робинсон : пер. с англ. – Москва : ДМК Пресс, 2010. – 352 с.

7. GD и функции для работы с изображениями [Электронный ресурс]. – Режим доступа : http://php.net/manual/ru/ref.image.php.

Зміст

НАВЧАЛЬНЕ ВИДАННЯ

ПРОГРАМУВАННЯ ЗАСОБІВ МУЛЬТИМЕДІА

Методичні рекомендації до самостійної роботи студентів

спеціальності 186 "Видавництво та поліграфія" першого (бакалаврського) рівня

Самостійне електронне текстове мережеве видання

Укладач **Браткевич** Вячеслав Вячеславович

Відповідальний за видання *О. І. Пушкар*

Редактор *О. В. Анацька*

Коректор *О. В. Анацька*

План 2018 р. Поз. № 287 ЕВ. Обсяг 32 с.

Видавець і виготовлювач – ХНЕУ ім. С. Кузнеця, 61166, м. Харків, просп. Науки, 9-А

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру ***ДК № 4853 від 20.02.2015 р.***