Луцький навчально-виховний комплекс

«Гімназія №14»

**Ігор Гісь**

**Розробка проектів**

**в Visual С++**



Луцьк - 2015

Схвалено науково-методичою радою НВК «Гімназія № 14» (протокол №\_ від \_\_\_\_\_\_\_\_ р.)

**Гісь І. В.** Розробка проектів в Visual С++ – Луцьк, 2012. – 48с.

Методичний порадник-практикум містить теоретичний матеріал та приклади розв’язаних задач з розділу «**Об’єктно-орієнтоване програмування. Основи комп’ютерного проектування** (Розробка проекту)». Проекти реалізовано, як CLR Windows Form.

**Автор - Гісь Ігор Володимирович,** учитель інформатики ВК гімназія №14, учитель-методист

**Рецензент - Остапчкук Л.Р. –** методист лабораторі інформатики Волинського ІППО

**Зміст**

[Вступ 4](#_Toc315847717)

[Структура вивчення 5](#_Toc315847718)

[Схема вивчення 6](#_Toc315847719)

[Етапи розв’язування задач на комп’ютері 7](#_Toc315847720)

[Приклади написання базових структуру в різних мовах програмування 9](#_Toc315847721)

[Логічні задачі 10](#_Toc315847722)

[Порядок роботи в середовиші 12](#_Toc315847723)

[Структура програми 12](#_Toc315847724)

[Слідування 13](#_Toc315847725)

[Розгалуження 14](#_Toc315847726)

[Цикл 15](#_Toc315847727)

[Підрограми 16](#_Toc315847728)

[Масив 17](#_Toc315847729)

[Рядки 19](#_Toc315847730)

[Операції над адресами. Вказівники 21](#_Toc315847731)

[Файли 22](#_Toc315847732)

[Розв’язування олімпіадних задач 25](#_Toc315847733)

[Задача 1. «Повороти» 25](#_Toc315847734)

[Задача 2. «Одиниці» 26](#_Toc315847735)

[Завдання 3. Трикутне число 27](#_Toc315847736)

[Задача 4. «Нафтові плями» 28](#_Toc315847737)

[Задача 5. «Прямокутники» 30](#_Toc315847738)

[Задача 6. «Квадрат» 30](#_Toc315847739)

[Завдання 7. «Ламана» 31](#_Toc315847740)

[Задача 8. «Білі плями» 33](#_Toc315847741)

[Задача 9. Цегляна стіна 35](#_Toc315847742)

[Задача 10. Порядок 36](#_Toc315847743)

[Задача 11. Код Грея 38](#_Toc315847744)

[Задача 12. Паліндроми 40](#_Toc315847745)

[Задача 13. Рядки 42](#_Toc315847746)

[Задача 14. Сума 44](#_Toc315847747)

[Задача 15. Квадратний корінь 47](#_Toc315847748)

[Задача 16. MATCHES 51](#_Toc315847749)

[Задача17. POINT 52](#_Toc315847750)

[Задача 18. POLYGON 54](#_Toc315847751)

[Задача 19. Точка 57](#_Toc315847752)

[Задача 20. Стіл 57](#_Toc315847753)

[Задача 21. Максимум 59](#_Toc315847754)

[Задача 22. Нумеролог 60](#_Toc315847755)

[Задача 23. Спіраль 61](#_Toc315847756)

[Задачі для опрацювання 63](#_Toc315847757)

[Структура слідування 1 63](#_Toc315847758)

[Структура слідування 2 63](#_Toc315847759)

[Структура слідування 3 64](#_Toc315847760)

[Структура розгалуження 1 65](#_Toc315847761)

[Розгалуження та вибір 2 65](#_Toc315847762)

[Структура розгалуження 3 66](#_Toc315847763)

[Структура циклу 1 68](#_Toc315847764)

[Структура циклу 2 69](#_Toc315847765)

[Структура циклу 3 70](#_Toc315847766)

[Обробка масивів 1 71](#_Toc315847767)

[Обробка масивів 2 72](#_Toc315847768)

[Обробка масивів 3 76](#_Toc315847769)

[Список використаних джерел 78](#_Toc315847770)

Єдиний спосіб вивчати нову мову програмування –

писати на ній програми.

*Брайен Керніган*

# Вступ

Що потрібне, щоб досягнути успіху в олімпіадах з інформатики? Як показує практика, тільки знання мови програмування для цього не достатньо. Виявляється, що олімпіадні задачі з інформатики знаходяться на межі математики і програмування. І дуже часто виявляється, що, розв’язуючи задачі, школярі не тільки вчаться програмувати, але й вивчають нові розділи математики і програмування.

Щоб досягнути результатів в олімпіаді з програмування школяр повинен не тільки володіти мовою програмування, але і вміти придумувати і реалізовувати алгоритми розв’язку задач, оцінювати час їх роботи, тестувати і налагоджувати свої програми. Метою підготовки до олімпіад є не перемога в тій або іншій олімпіаді, а формування алгоритмічного мислення, розуміння, як розв'язується задача, записати її розв’язок мовою програмування, вміння тестувати і налагоджувати програми.

Посібник містить теоретичний матеріал та приклади розв’язаних задач з розділу «Методика складання алгоритмів та їх аналіз». Програми розв’язку задач реалізовано мовою програмування С++.

Мова формує наш спосіб мислення і визначає те, про що ми можемо думати. Прогрес комп'ютерних технологій визначив процес появи нових різноманітних знакових систем для запису алгоритмів – мов програмування.

C++ - універсальна мова програмування, задумана так, щоб зробити програмування приємнішим для серйозного програміста. C++ є надмножиною мови програмування C.

***Мета навчання***:

* розвиток логічного, аналітичного мислення та основних видів розумової діяльності: уміння використовувати індукцію, дедукцію, аналіз, синтез, робити висновки, узагальнення;
* розвиток уміння розв’язувати змістовні задачі різного рівня складності, олімпіадні задачі, користуючись відомими теоретичними положеннями, математичним апаратом, літературою та комп’ютерною технікою;
* навчити вчителів правильному підборі задач для підготовку учнів до участі в олімпіадах.

# Структура вивчення

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Тема** | **План** | **Засоби** |
| 1. | Розділ «Алгоритмізація і програмування» | * кількість годин на вивчення за програмою
* «+» і «-» вивчення розділу
* тематика вивчення: 1) базові структури алгоритмів; 2)методика складання алгоритмів та їх аналіз; 3) об’єктно-орієнтоване програмування, візуальне програмування.
* принцип IPO (input procedure output): визначення місцезнаходження і введення даних; задання операцій над даними; виведення результуючих даних
* етапи розв’язування задач з використанням ЕОМ.
 | презентація «Алгоритмізація і програмування»мозковий штурманаліз схеми «людина-задача-алгоритм-програма-комп’ютер»додаток: Етапи розв’язування задач  |
| 2. | Мови програмування | * структуровані мови програмування
* С – Деніс Рітчі (1972)
* C++ - 80-ті роки, Бьяртні страус труп (1979)
* середовище Borland C++ 3.1
 | завантаження і робота в середовищі |
| 3. | Лексеми мови програмування | * алфавіт мови і ключові слова
* директиви препроцесора (#include)
* сталі (const назва=значення)
* змінні і їх типи; (цілі: int спиок змінних; дійсні: float змінна=значення; символьний char с1=’A’, c2=65; логічний bool false, true)
* коментарі // ...

 /\* ... \*/* перша програма (структура програми)
 | приклад структури програми |
| 4. | Присвоєння, вирази, функції | * = аналіз виразів
* функції math.h
 | приклади виразів |
| 5. | Введення і виведення даних | * stdio.h: scanf (“%d”, &a) &змінна – адреса даних puts (“рядок”); %d –int %f – float %c –char %s – рядок символів
* conio.h: cin >> змінна; cout << «текст»<< p
 | приклади рядків введення і виведення  |
| 6. | Базові структури | * слідування
* розгалуження (==, !=, ! – не, && - і, || - або, IF (логічний вираз) команда 1; else команда 2)
* вибір (switch (вираз) {case ознака 1: команда 1; break; default: команда}
* цикл: for ( ) {}, while (умова) {}, do команда while (вираз);
* підпрограми
 | додаток з задачами:приклади розв’язаних задач і задачі для самостійного розв’язування  |
| 7. | Типи даних | * масиви
* рядки
* вказівники
* файли

  |

# Схема вивчення

# Етапи розв’язування задач на комп’ютері

Перш ніж одержати очікуваний результат роботи програми на комп'ютері, необхідно виконати досить багато копіткої підготовчої роботи.

**1.Постановка задачі.** Розв'язування будь-якої задачі починається з ЇЇ постановки, викладеної мовою чітко визначених математичних понять. На першому кроці необхідно добре уявити, в чому саме полягає дана задача, які необхідні початкові дані, яку інформацію вважати результатами розв'язання.

**2. Побудова математичної моделі.** Побудова математичної моделі алгоритму — дуже відповідальний етап. Не завжди умова сформульованої задачі містить в собі математичну формулу, яку можна застосувати для розробки алгоритму задачі, не завжди розв'язок задачі вдається одержати в явному вигляді, що зв'язує вхідні дані га результат. Для цього створюється інформаційна математична модель об'єкта, що вивчається. Вибір виду моделі залежить від інформаційної сутності об'єкта, а не від його фізичної природи. Тобто, настільки важливе прикладне значення задач, як однотипність методів, якими вони розв'язуються. Наприклад, логічні моделі використовуються як для моделювання словесних міркувань, так і для опису логічних схем автоматики. Розв'язуючи задачу про рух тіла під дією прикладених до нього сил, ми перш за все записуємо рівняння його руху на основі законів механіки. Проте, крім сили тяжіння, на тіло діє і сила опору повітря. Постає питання достовірності математичної моделі і реальної картини досліджуваного об'єкта. Іноді буває неможливо врахувати всі реальні фактори, що впливають на нього. Тому дуже важливим є вміння виділити серед усіх факторів головні і другорядні, щоб останніми можна було знехтувати. При цьому може скластися ситуація, коли наперед невідомо, якими саме факторами можна знехтувати, і тому може бути кілька математичних моделей, які описують один і той самий об'єкт з різним ступенем достовірності. Відповідність моделі реальному об'єкту перевіряється практикою, комп'ютерним експериментом. Критерій застосування практики дає можливість оцінити побудовану модель і уточнити її за необхідності. Чим достовірніше математична модель відображає реальні сторони об'єкта, тим точніші одержувані результати.

**3. Побудова алгоритму.** Наступним етапом є розробка алгоритму обробки інформації на основі побудованої математичної моделі. Тепер необхідно знайти спосіб розв'язування цієї задачі. Для цього можуть бути застосовані вже відомі методи, проведена їх оцінка, аналіз, відбір або розроблені нові методи. Наприклад, вибір методу розв'язування системи рівнянь, що описує дану математичну модель. Під час створення складних алгоритмів застосовується метод покрокової розробки. Сутність цього методу полягає в тому, що алгоритм розробляється «зверху донизу». На кожному етапі приймається невелика кількість рішень, що призводить до поступової деталізації, уточнення як виконуваної, так і інформаційної структури алгоритму. Такий підхід дозволяє розбити алгоритм на окремі частини — модулі, кожний з яких розв'язує свою самостійну підзадачу. Це дає можливість сконцентрувати зусилля на розв'язуванні кожної підзадачі, що реалізується у вигляді окремої процедури. Для кожного такого модуля визначаються свої методи реалізації алгоритму та структура даних, якими він оперує. Останнім етапом у методі покрокової розробки є об'єднання окремих незалежних модулів у єдине ціле. Для цього між модулями повинні бути встановлені зв'язки, тобто узгоджена передача інформації від одних модулів до інших: результати виконання одних модулів є вхідною інформацією для інших.

Якщо поставлена задача є складним завданням, то важливість розбиття алгоритму на окремі модулі неможливо переоцінити.

**4. Вибір мови програмування.** Алгоритм, призначений для виконання на комп'ютері, має бути записаний мовою програмування. Різноманітність існуючих мов програмування потребує від програміста реальної оцінки складності та характеру задачі. Тільки після цього він зможе здійснити оптимальний вибір мови програмування для реалізації поставленої задачі. Враховуючи можливість розбиття всього алгоритму на окремі модулі, для кожного з них вибір мови програмування можна здійснити окремо. Процес розробки програми, як і алгоритму, може здійснюватися за принципом «зверху донизу». Це дозволяє одержати добре структуровану програму, читання і розуміння якої значно полегшене.

**5. Складання програми.** Цей етап потребує лише знання вибраної мови програмування. Суть його полягає в тому, щоб на основі розроблених алгоритмів і структур даних створити програму для комп'ютера.

**6. Компіляція програми.** Переведення програми на машинну мову здійснюється за допомогою спеціальних програм — компіляторів. Однією з їх функцій є перевірка відсутності в програмі синтаксичних помилок. Не тіште себе надією, що ваша, навіть найпростіша, програма написана бездоганно. Серед програмістів побутує прислів'я:
«Якщо програма не має помилок, це означає, що у вас поганий компілятор!!!»

**7. Налагодження програми, контрольний прорахунок.** Виправлення усіх синтаксичних помилок у програмі на попередньому етапі .зовсім не позбавляє вас від помилок іншого типу — змістовних, логічних. Вони з'являються під час помилкового трактування умови поставленої задачі, через недосконалість математичної моделі або недоліки у побудованому алгоритмі, що призводить до одержання помилкового результату. Такі помилки не можуть бути усунені на стадії компіляції, тому що для їх виявлення необхідна інформація про сутність самої задачі. Це може зробити лише сам розробник. Процес налагодження програми полягає втому, що готується система тестів, за допомогою якої перевіряється робота програми в різних можливих режимах. Кожний тест містить набір вхідних даних, для яких відомий результат. Для більшості програм виникає необхідність добору не одного, а серії тестів, щоби перевірити якнайбільше можливих ситуацій, які можуть виникнути *в* процесі роботи програми. Якщо для всіх тестів результати роботи програми збіглися з розрахунками, то можна вважати, що логічних помилок немає.

**9. Експлуатація програми.** Тепер програму можна тиражувати і пропонувати іншим користувачам, доповнивши її відповідною документацією для користувачів. Документація повинна містити опис виконуваної задачі, опис середовища, в якому може працювати програма, обмеження на вхідні дані, формат вхідних та вихідних даних. Більшість розглянутих етапів розв'язування задач на комп'ютері виконуються людиною і носять творчий, евристичний характер.

**Панель елементів Windows Form (CLR)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Інструмент** | **Призначення** | **Властивості** | **Події (приклад програмного коду)** |
| **form** |  |  |  |
| **label** |  |  |  |
| **textbox** |  |  |  |
| **Button** |  |  |  |
| **listBox** |  |  |  |
| **comboBox** |  |  |  |
| **pictureBox** |  |  |  |
| **richTextBox** |  |  |  |
| **listView** |  |  |  |
| **checkedListBox** |  |  |  |
| **radioButton** |  |  |  |
| **checkbox** |  |  |  |
| **groupBox** |  |  |  |
| **panel** |  |  |  |
| **menuStrip** |  |  |  |
| **contextMenuStrip** |  |  |  |
| **openFileDialog** |  |  |  |
| **saveFileDialog** |  |  |  |
| **timer** |  |  |  |

Побудова графічних примітивів

private: System::Void pictureBox1\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

 {

////малює лінію

 e ->Graphics ->DrawLine(Pens::Black, 10, 20, 100, 20);

 }

 private: System::Void pictureBox2\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

 {

 int x = 10; //координата х

 int y = 10; //координата у

 int w = 28; //ширина

 int h = 60; //висота

//FillRectangle малює зафарбований прямокутник

 e ->Graphics ->FillRectangle(Brushes::Green, x, y, w, h);

 }

private: System::Void pictureBox3\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

 {

 int x = 10; //координата х

 int y = 10; //координата у

 int w = 28; //ширина

 int h = 60; //висота

 // //створення олівця програміста

 System::Drawing::Pen^ aPen;

 aPen = gcnew System::Drawing::Pen(Color::Red, 2);

 aPen ->Width = 4;

 aPen ->Color = Color::Green;

 // //DrawRectangle малює контур прямокутника

 e ->Graphics ->DrawRectangle(aPen, x, y, w, h);

 }

private: System::Void pictureBox4\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

 {

 // //ламана лінія

 array<Point>^ p;

 p = gcnew array<Point>(5);

 p[0].X = 10; p[0].Y = 50;

 p[1].X = 20; p[1].Y = 20;

 p[2].X = 30; p[2].Y = 50;

 p[3].X = 40; p[3].Y = 20;

 p[4].X = 50; p[4].Y = 50;

 // //DrawLines малює ламану лінію, використовуючи масив типу Point

 // //елементи якого містять координати вузлових точок лінії

 // //малює лінію послідовно сполучаючи точки: першу з другою,

// //другу з третьою і так далі

 e ->Graphics ->DrawLines(Pens::Black, p);

 }

private: System::Void pictureBox5\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

 {

 array<Point>^ p;

 p = gcnew array<Point>(5);

 // //багатокутник

 p[0].X = 10; p[0].Y = 30;

 p[1].X = 10; p[1].Y = 10;

 p[2].X = 30; p[2].Y = 20;

 p[3].X = 50; p[3].Y = 10;

 p[4].X = 50; p[4].Y = 30;

 // //FillPolygon малює зафарбований багатокутник

 e ->Graphics ->FillPolygon(Brushes::Gold, p);

 // //DrawPolygon малює контур багатокутника

 e ->Graphics ->DrawPolygon(Pens::Black, p);

 }

private: System::Void pictureBox6\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

 {

 int x = 10; //координата х

 int y = 10; //координата у

 int w = 28; //ширина

 int h = 60; //висота

////заданий стиль лінії

 array<float>^ myPattern;

 myPattern = gcnew array<float>(2);

 myPattern[0] = 5; //штрих

 myPattern[1] = 5; //пропуск

 System::Drawing::Pen^ aPen;

aPen = gcnew System::Drawing::Pen(Color::Red, 2);

 aPen ->Color = Color::Olive;

 aPen ->DashStyle = System::Drawing::Drawing2D::DashStyle::Custom;

 aPen ->DashPattern = myPattern;

 e ->Graphics ->DrawRectangle(aPen, x, y, w, h);

 }

private: System::Void pictureBox7\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

 {

 int x = 10; //координата х

 int y = 10; //координата у

 int w = 28; //ширина

 int h = 60; //висота

 // //штриховий пензлик HatchBrush

System::Drawing::Drawing2D::HatchBrush^ hBrush = gcnew System::Drawing::Drawing2D::HatchBrush(

 System::Drawing::Drawing2D::HatchStyle::DottedGrid, Color::Black, Color::SkyBlue);

 e ->Graphics ->FillRectangle(hBrush, x, y, w, h);

 }

private: System::Void pictureBox8\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e)

 {

 // //використання структури Point для завдання координат точки

 Point p1 = Point(5,10);

 Point p2 = Point(5, 50);

 e ->Graphics ->DrawLine(Pens::Lime, p1, p2);

 // //малювання точки в один піксел

 e ->Graphics ->FillRectangle(Brushes::Red, 50, 20, 1, 1);

 // //виводить на графічну поверхню рядок тексту

 e ->Graphics ->DrawString("Графічні примітиві", this ->Font, Brushes::Black, 10, 80);

 }

private: System::Void pictureBox13\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {

 e->Graphics->DrawEllipse(Pens::Black, 10, 10 , 40, 20);

 e->Graphics->DrawArc(Pens::Black, 10, 50 , 40, 20, 0,180);

 e->Graphics->DrawPie(Pens::Black, 10, 100 , 40, 40, 30,90);

 e->Graphics->FillEllipse(Brushes::Blue, 10, 150, 20, 40); }



# Список використаних джерел

1. Готуємось до олімпіад з інформатики / Упоряд. І.Скляр. – К. Ред. загальнопед. газ., 2005. – 128 с. – (Б-ка „Шк. світу”).
2. Грузман М.З. Эвристика в информатике. – Винница: Арбат, 1998.-308с.
3. Караванова Т. Основи алгоритмізації та програмування. 750 задач з рекомендаціями та прикладами. – К.: Форум. – 2002. – 283 с.
4. <http://algolist.manual.ru/> - широкий огляд алгоритмів і методів розв’язування задач.
5. <http://www.olympiads.ru/> - олімпіадна інформатика.
6. <http://www.schoololymp.byethost32.com/> - школа олімпійського резерву.
7. <http://www.vippoolimp.byethost14.com/> - Волинська учнівська інтернат олімпіада.