А. В. Погудін, О. К. Погудіна

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

А. В. Погудін, О. К. Погудіна

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

Конспект лекцій

Харків «ХАІ» 2020

УДК 004.43 (075.8)
П-43

Рецензенти: д-р техн. наук, проф. М. Л. Угрюмов,
д-р техн. наук, проф. Г. А. Кучук

Погудін, А. В.

П-43 Алгоритмізація та програмування [Електронний ресурс] : консп. лекцій / А. В. Погудін, О. К. Погудіна. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2020. – 41 с.

Подано тематику, основні визначення та посилання на підручники або електроні посібники, що містять пояснення для самостійної підготовки студентів. Курс поділено на три змістові модулі: основи процедурного програмування, структури даних та алгоритми, графічний режим.

Для студентів, що навчаються в галузі знань 12 «Інформаційні технології», при самостійній підготовці до лекцій, повторенні й вивченні основного матеріалу курсу «Алгоритмізація та програмування».

Іл. 5. Бібліогр.: 17 назв

УДК 004.43 (075.8)

Змістовий модуль 1. ОСНОВИ ПРОЦЕДУРНОГО ПРОГРАМУВАННЯ

Міжпредметні зв'язки: зв'язок з елементами знань і умінь таких навчальних дисциплін, як „Дискретна математика" і „Основи програмування".

Мета лекцій змістового модуля: ознайомлення з поняттям алгоритму, його властивостями та способами подачі; формування в студентів навичок алгоритмічного мислення, уміння здійснювати постановку задачі для розроблення програмного забезпечення і реалізації алгоритмів у вигляді комп'ютерних програм мовою С++ з використанням інтегрованого середовища на прикладі Microsoft Visual Studio.

Опорні поняття: алгоритм, системи числення, властивості, блок-схема, головний алгоритм, допоміжний алгоритм, структура.

Посилання на приклади та більш детальне пояснення розглянутих на лекціях питань за такими темами:

- архітектура комп'ютерів, ознайомлення з підсистемами інтегрованого середовища на прикладі Microsoft Visual Studio [1, 5];
- поняття алгоритму й основні алгоритмічні структури програмування, властивості та способи опису алгоритму, різновиди алгоритмічних структур: послідовності, повторення, розгалуження та альтернативне розгалуження [3];
- елементи алгоритмічних мов: концепція типів даних, імена, значення, покажчики, змінні, константи, операції, вирази [4];
- структурне програмування: послідовність, розгалуження та цикли, блок-схема як одна з наочних форм зображення алгоритму [2];
- побудова блок-схеми алгоритму обчислення факторіала натурального числа, побудова блок-схеми алгоритму транспонування матриці [1];
- алгоритмічний вибір альтернатив, алгоритмічна конструкція повторення [1, 3];
- процедурно-орієнтоване програмування, рекурсія, підпрограми, їх різновиди та способи використання, призначення процедур і функцій, процедури користувача, поняття формальних і фактичних параметрів, локальних і глобальних змінних, процес виклику підпрограми [7, 8];
- опис алгоритму та розроблення програми, що використовує меню з вибором теми [6, 3];
- опис алгоритму та розроблення програми знаходження простого числа за його номером у послідовності всіх простих чисел з використанням процедур з параметрами і технологією низхідного проектування [6];
- фундаментальні роботи [7 - 9], що стосуються майже всіх тем, але викладені для студентів, що вже мають початкові знання з програмування С++.

Лекція № 1. АРХІТЕКТУРА КОМП'ЮТЕРІВ, ПРИНЦИПИ ФОН НЕЙМАНА. ОЗНАЙОМЛЕННЯ З ПІДСИСТЕМАМИ ІНТЕГРОВАНОГО СЕРЕДОВИЩА НА ПРИКЛАДІ MICROSOFT VISUAL STUDIO

Архітектурою комп'ютера називається його опис на деякому загальному рівні, що містить опис призначених для користувача можливостей програмування, системи команд, системи адресації, організації пам'яті тощо. Архітектура визначає принципи дії, інформаційні зв'язки і взаємне з'єднання основних логічних вузлів комп'ютера: процесора, оперативного запам'ятовуючого пристрою (ЗП), зовнішніх ЗП і периферійних пристроїв. Інтегроване середовище розроблення — комплексне програмне рішення для розроблення програмного забезпечення. Більшість сучасних середовищ розроблення мають можливість автодоповнення коду. Microsoft Visual Studio — інтегроване середовище розроблення, яке дозволяє розробляти як консольні програми, так і програми з графічним інтерфейсом, а також веб-сайти, веб-сторінки, веб-служби як в рідному, так і в керованому кодах для всіх платформ. Система числення, в якій значення кожної цифри залежить від місця в послідовності цифр у записі числа, називається позиційною. Загальноприйнятою в сучасному світі є десяткова позиційна система числення, яка з Індії через арабські країни прийшла до Європи. Найпоширенішою для подання чисел у пам'яті комп'ютера є двійкова система числення. Принципи фон Неймана такі: використання двійкової системи числення, однорідність пам'яті (тобто команди зберігаються в оперативній пам'яті разом з даними), адресність памяті (тобто кожна комірка памяті має унікальну адресу або номер, що однозначно її ідентифікує. Доступ до даних здійснюється за цією адресою), послідовне програмне управління (усі програми складаються з набору команд, що виконуються послідовно, починаючи з першої).

До тенденцій розвитку сучасних комп'ютерів належать: багатопроцесорність (магістральної або конвеєрної, векторної, матричної схем, а також кластерної архітектури); паралелізм (на рівні бітів та інструкцій, даних і завдань); оптичний, молекулярний та біокомп'ютер, нейрокомп'ютер.

Контрольні запитання

1. Що визначає архітектура комп'ютера?
2. Яке інтегроване середовище розроблення є найбільш популярним?
3. Що дозволяє робити інтегроване середовище Microsoft Visual Studio?
4. Як називають систему числення, в якій значення кожної цифри залежить від місця в послідовності цифр у записі числа?
5. Найпоширеніша система числення.

Лекція № 2. ПОНЯТТЯ АЛГОРИТМУ Й ОСНОВНІ АЛГОРИТМІЧНІ СТРУКТУРИ ПРОГРАМУВАННЯ. ВЛАСТИВОСТІ ТА СПОСОБИ ОПИСУ АЛГОРИТМУ. РІЗНОВИДИ АЛГОРИТМІЧНИХ СТРУКТУР: ПОСЛІДОВНОСТІ, РОЗГАЛУЖЕННЯ, ПОВТОРЕННЯ. РОЗГАЛУЖЕННЯ ТА АЛЬТЕРНАТИВНЕ РОЗГАЛУЖЕННЯ

Алгоритм — це послідовність точних команд виконавцеві алгоритму, спрямованих на розв'язання певної задачі. Базові структури алгоритмів — це способи керування процесами оброблення даних. Комбінуючи керуючі структури, можна складати алгоритми (програми) для різноманітних задач. Властивості алгоритму:

- дискретність: сукупність відокремлених одна від одної команд, кожна з яких виконується за кінцевий час;
- однозначність;
- формальність: будь-який виконавець, що володіє заданою системою команд виконавця, має виконати алгоритм й отримати такий самий результат;
- масовість: алгоритм повинен передбачати можливість розв'язання групи типових задач, зміни вхідних (початкових) даних у деяких допустимих межах;
- скінченність;
- результативність.

Способи подання алгоритмів: словесний, формульно-словесний, графічний, програмний.

Виділяють три базові алгоритмічні структури: 1. Лінійні (слідування) – це лінійна структура, яка являє собою послідовність команд, виконуваних поспіль. 2. Розгалуження (вибір) – це вид керуючої структури, що передбачає можливість вибору команд залежно від умови. 3. Повторення (цикл) – це процес, який виконується кілька разів залежно від виконання умови.

Оператори – це основні елементи, з яких «будуються» програми, призначені для виконання встановлених дій. За конструкцією оператори поділяють на групи: прості, складені. За характером дій: оператори-вирази, умовні оператори, переходу, циклу. Окремий вид складеного оператора – блок. Це група довільних операторів, об’єднаних фігурними дужками {…}. Усередині блока можна оголошувати локальні змінні.

Контрольні запитання

1. Що таке алгоритм?
2. Які Ви знаєте властивості алгоритмів?.
3. Що таке результат виконання алгоритму?
4. Для чого використовуються проміжні величини?
5. Які Ви знаєте методи запису алгоритмів?

Лекція № 3. ЕЛЕМЕНТИ АЛГОРИТМІЧНИХ МОВ: КОНЦЕПЦІЯ ТИПІВ ДАНИХ, ІМЕНА, ЗНАЧЕННЯ, ПОКАЖЧИКИ, ЗМІННІ, КОНСТАНТИ, ОПЕРАЦІЇ, ВИРАЗИ

Мова програмування (англ. Programming language) − це штучна мова, створена для передачі команд машинам, зокрема комп'ютерам. Мови програмування використовуються для створення програм, котрі контролюють поведінку машин, і запису алгоритмів.

Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми і дії, які виконує виконавець (комп'ютер) під її управлінням.

У групі універсальних мов високого рівня безумовним лідером сьогодні є мови С і С ++, що мають цілий ряд дуже істотних переваг: багатоплатформність − для всіх використовуваних у даний час платформ існують компілятори з мов С і С ++; наявність операторів, що реалізують основні структурні алгоритмічні конструкції (умовне оброблення, усі види циклів); можливість програмування на низькому (системному) рівні з використанням адреси оперативної пам'яті;

Усі типи даних мови С++ можна розділити на основні (базові) і ті, що конструюються. Основні типи даних часто називають арифметичними, тому що їх можна використовувати в арифметичних операціях. Для опису основних типів мови С++ використовують такі службові слова: int (цілий); char (символьний); bool (логічний); float (дійсний); double (дійсний з подвійною точністю); void (порожній, не має значення).

Ім'я змінної (ідентифікатор змінної) − це буквено-цифрова послідовність символів алфавіту, що починається з букви. Імена в програмі повинні бути унікальними і в жодному разі не мають збігатися з ключовими словами. Для кожної змінної, яка фігурує в програмі під певним ім'ям, компілятор резервує область пам'яті, в якій буде зберігатися значення змінної. Змінна у програмі – це іменована комірка пам'яті, яка призначена для зберігання деякого значення. Збережене значення може бути результатом обчислення виразу, числовою чи рядковою константою тощо. Ім'я змінної є іменем ідентифікатора. Це ім'я повинно відповідати синтаксису мови С++.

Елемент даних, що має фіксоване значення, яке під час розв'язання задачі не змінюється, називають ***константою.***

Комбінація знаків операцій та операндів, результатом якої є конкретне значення, називається ***виразом.*** Для запису виразу в С++ використовують відповідні операції. У С++ існують унарні, бінарні, тернарні операції. Вирази можуть бути: арифметичними, символьними, логічними.

Контрольні запитання

1. Які типи даних бувають?
2. Як називають основні типи даних?
3. Яким службовим словом описують символьний тип?
4. Чи можуть повторюватися імена змінних?
5. Що таке вираз?

Лекція № 4. СТРУКТУРНЕ ПРОГРАМУВАННЯ: ПОСЛІДОВНІСТЬ, РОЗГАЛУЖЕННЯ, ЦИКЛИ. БЛОК-СХЕМА ЯК ОДНА З НАОЧНИХ ФОРМ ЗОБРАЖЕННЯ АЛГОРИТМУ

Структурне програмування — це технологія створення програм, що дозволяє шляхом дотримання певних правил скоротити час розроблення і зменшити кількість помилок, а також полегшити можливість модифікації програми. Теоретично доведено, що будь-який алгоритм можна реалізувати лише з трьох структур, які називаються базовими конструкціями структурного програмування: це − послідовність, розгалуження, цикл.

Послідовністю називається конструкція, що реалізовує послідовне виконання двох або більше операторів (простих або складних).

Розгалуження задає виконання того чи іншого оператора залежно від виконання будь-якої умови. Реалізується за допомогою операторів if і switch.

Цикл реалізує багатократне виконання операторів. Реалізується за допомогою операторів циклу.

Блок-схема − це спосіб подання алгоритму в графічній формі, у вигляді геометричних фігур, сполучених між собою лініями (стрілками).

Блок-схема алгоритму зображує послідовність блоків, з'єднаних між собою стрілками, які вказують послідовність виконання і зв'язок між блоками. Всередині блоків записується їх короткий зміст.

Існують правила зображення блок-схем алгоритмів. Кожен алгоритм має початок і кінець. Кожна команда алгоритму подається у вигляді геометричних символів, які мають певну конфігурацію, залежно від характеру дій, що будуть виконуватись. Геометричні символи з’єднуються між собою лініями або стрілками, які вказують порядок виконання дій.

Контрольні запитання

1. Що таке структурне програмування?
2. Базові конструкції структурного програмування.
3. Чим відрізняються розгалуження від циклу?
4. Спосіб подання алгоритму в графічній формі.
5. Які геометричні символи використовуються у блок-схемах?

Лекція № 5. ПОБУДОВА БЛОК-СХЕМИ АЛГОРИТМУ ОБЧИСЛЕННЯ ФАКТОРІАЛА НАТУРАЛЬНОГО ЧИСЛА. ПОБУДОВА БЛОК-СХЕМИ АЛГОРИТМУ ТРАНСПОНУВАННЯ МАТРИЦІ

Факторіал − це добуток усіх натуральних чисел від 1 до N включно.

Вирішити це завдання можна декількома способами: циклічне та рекурсивне обчислення факторіала.

Циклічне обчислення. Для знаходження факторіала треба написати функцію, яка буде набувати значення N і повертати результат:

long factorial (long N) { long f=1; for(long i=2;i<=N;i++) f*=i; return f;}

Функція набуває значення N, після чого визначається змінна, яка буде зберігати в собі відповідь. Запускається цикл for від 2 до N, значення зміної і будуть використовуються для перемноження.

До початку обчислень необхідно дізнатися N. N може бути більше одиниці або дорівнювати їй (N> = 0). Тому для початку треба написати каркас додатку, який буде отримувати значення N і перевіряти його.

Рекурсивне знаходження факторіала. Для такого способу теж треба написати функцію, але вона не буде містити циклу. Рекурсивна функція − це функція, яка викликає сама себе:

long factorial (long N) { return N * factorial (N - 1); }

Функція factorial () буде набувати значення N і повертати у програму N * factorial (N-1). Тобто буде повертати значення N, помножене на результат розрахунку самої себе, але тільки з N-1.

Операція транспонування полягає в тому, що рядки і стовпці у вихідній матриці міняються ролями. У транспонованій матриці першим стовпцем служить перший рядок вихідної матриці, другим стовпцем − другий рядок вихідної матриці, і т. д. (рис. 5.1).

source (2,3) → transposed(3,2)

1	3	4
2	-1	-2

1	2
3	-1
4	-2

Рис. 5.1. Приклад транспонування матриці source

Якщо матриця source складається з n рядків і m стовпців - то матриця transposed повинна містити m рядків і n стовпців, наприклад:

for (int i = 0; i <n; i ++) for (int j = 0; j <m; j ++) transposed [j] [i] = source [i] [j];

Контрольні запитання

1. Що таке факторіал?
2. Якими способами можна обчислити факторіал?
3. Перевірте на практиці алгоритм для факторіалу 5!
4. Що таке рекурсивна функція?
5. Яких значень набуває функція factorial ()?

Лекція № 6. АЛГОРИТМІЧНИЙ ВИБІР АЛЬТЕРНАТИВ. АЛГОРИТМІЧНА КОНСТРУКЦІЯ ПОВТОРЕННЯ

Алгоритмічна конструкція, що дозволяє виконавцеві алгоритму вибрати ту чи іншу послідовність дій, називається розгалуженням, або конструкцією вибору альтернатив. Є такі різновиди конструкції вибору, як вибір з двох альтернатив і поліваріантний вибір. Алгоритмічна конструкція альтернативного розгалуження, або конструкція вибору з двох альтернатив дозволяє виконавцеві алгоритму вибрати один із двох варіантів дій залежно від істинності деякої умови. У альтернативні розгалуження реалізуються умовним оператором (розгалуження) і умовним виразом. Операторний блок − це послідовність операторів, що оточені фігурними дужками { }. У мовах С і С++ алгоритмічну конструкцію поліваріантного виразу реалізовано оператором перемикання.

Оператор циклу з передумовою виконується за таким алгоритмом. Обчислення умови продовження циклу, що записана в його заголовку. Якщо вона істинна, то виконується тіло циклу, інакше виконання циклу припиняється. Після виконання тіла циклу буде знову перевірена умова його продовження. Оператор циклу з постумовою працює за таким алгоритмом. Спочатку виконуються оператори, що входять до складу тіла циклу. Потім обчислюється умова завершення циклу. Якщо вона істинна, цикл завершує свою роботу, інакше повторюється виконання його тіла.

Виконання оператора for здійснюється за таким алгоритмом. Спочатку обчислюються та порівнюються значення виразів x_o (початкове значення) і x_n (кінцеве значення). Якщо початкове значення більше за кінцеве, то виконання циклу завершується, інакше лічильнику циклу присвоюється початкове значення і виконується тіло циклу. Після виконання тіла циклу порівнюється поточне значення лічильника з його кінцевим значенням, і, якщо ці значення не дорівнюють одне одному, то поточне значення лічильника збільшується на h (значення кроку) і виконання циклу триває далі.

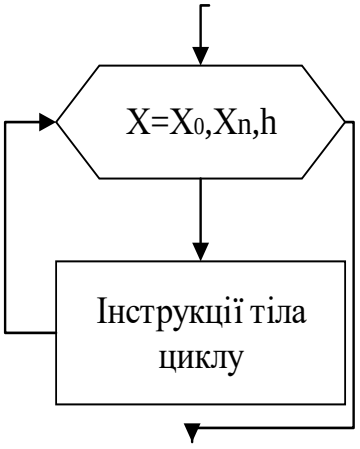

Рис. 6.1. Цикл for

Контрольні запитання

1. Що працює ефективніше: умовний вираз чи оператор if...else?
2. Що таке умовний вираз і як він записується?
3. Чи можна перервати роботу програми за допомогою оператора break?
4. Що таке алгоритмічна конструкція повторення?
5. Цикл із постумовою.

Лекція № 7. ПРОЦЕДУРНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ. РЕКУРСІЯ. ПІДПРОГРАМИ, ЇХ РІЗНОВИДИ ТА СПОСОБИ ВИКОРИСТАННЯ. ПРИЗНАЧЕННЯ ПРОЦЕДУР І ФУНКЦІЙ. ПРОЦЕДУРИ КОРИСТУВАЧА. ПОНЯТТЯ ФОРМАЛЬНИХ І ФАКТИЧНИХ ПАРАМЕТРІВ, ЛОКАЛЬНИХ І ГЛОБАЛЬНИХ ЗМІННИХ. ПРОЦЕС ВИКЛИКУ ПІДПРОГРАМИ

C++ та інші схожі з ним мови програмування належать до категорії процедурних мов. Кожен оператор такої мови є вказівкою комп'ютеру виконати певну дію. Програміст створює перелік інструкцій, а комп'ютер виконує дії, що відповідають цим інструкціям.

Рекурсія − визначення частини функції (методу) через саму себе, тобто це функція, яка викликає сама себе, безпосередньо (в своєму тілі) або побічно (через іншу функцію).

Підпрограма − частина програми, яка реалізує певний алгоритм і дозволяє звернення до неї з різних частин загальної (головної) програми. У термінах мов програмування: функції (C), процедури (Pascal), методи. Підпрограми поділяються на стандартні і визначені користувачем. Перші входять до складу мови програмування і викликаються для виконання за строго фіксованими іменами. Другі розробляються самим користувачем. Підпрограми користувача описуються або в програмі, або в бібліотеках користувача, які за необхідності теж підключаються до тексту програми. Функції дозволяють зробити програму модульною, тобто розділити програму на кілька маленьких підпрограм (функцій), які в сукупності виконують поставлене завдання.

Формальні параметри – це змінні, що набувають значення аргументів (параметрів) функції. Якщо функція має декілька аргументів (параметрів), їх тип та імена розділяються комою. При виклику функції, що має аргументи, компілятор здійснює копіювання копій формальних аргументів у стек. При виклику функції фактичні параметри копіюються в спеціальні комірки пам'яті в стеку (стек – частина пам'яті). Ці комірки пам'яті відведені для формальних параметрів. Таким чином, формальні параметри (через використання стеку) отримують значення фактичних параметрів.

Контрольні запитання

1. До якої категорії мов належить C++?
2. Як називається функція, яка викликає сама себе?
3. Чим підпрограма відрізняється від функції?
4. Який тип програм розробляється користувачем?
5. Чим відрізняються фактичні та формальні параметри?

Лекція № 8. ОПИС АЛГОРИТМУ ТА РОЗРОБЛЕННЯ ПРОГРАМИ, ЩО ВИКОРИСТОВУЄ МЕНЮ З ВИБОРОМ ТЕМИ

Нижче наведено заготовку програми використання меню для оброблення елементів масиву:

```
void InputArray(int Vec[],int N);
void GenArray(int Vec[],int N);
void ProcessArray(int Vec[],int N);
void OutputArray(int Vec[],int N);
void main()
{
   const int Max=100;
  int Arr[Max],N;
   while(1)
   {
   cout<< “1. Введення фактичного числа елементів масиву”<<endl;
   cout<< “2. Ручне введення елементів масиву”<<endl;
   cout<< “3. Генерація елементів масиву”<<endl;
   cout<< “4. Обробка масиву”<<endl;
   cout<< “5. Виведення масиву-результату”<<endl;
   cout<< “6. Вихід з програми”<<endl;
   cout<< “Уведи номер теми меню”<<endl;
   char ch=getche();
   cout<< endl;
       switch()
       {
       case ‘1’: cout<< “Уведи фактичне число елементів масиву”;
                    cin>>N; break;
       case ‘2’:  InputArray(Arr , N); break;
       case ‘3’: GenArray(Arr , N); break;
       case ‘4’:  ProcessArray(Arr, N); break;
       case ‘5’:  OutputArray(Arr , N); break;
       case ‘6’: return;
       default: cout<<endl<<“Уведи число від 1 до 6. Зрозумів?”; <<endl;
       }
   }
}
void InputArray(int Vec[],int N)
{
cout<<endl<<"Хазяїн, виконую введення! "<<endl;
}
void GenArray(int Vec[],int N)
```

```
{
}
void ProcessArray(int Vec[],int N)
{
}
void OutputArray(int Vec[],int N)
{
}
```

Зверніть увагу також на заготовки функцій, які корисно зробити відразу, а далі доопрацьовувати, наповнюючи їх відповідним кодом.

Контрольні запитання

1. Назвіть усі прототипи функцій, що застосовані у прикладі.
2. Які цикли використовує надана програма?
3. Як потрібно змінити програму, щоб не застосовувати прототипи функцій?
4. Що виконує оператор switch?
5. Чи буде працювати код програми без додавання змісту функцій генерації, оброблення та виводу масиву.?

Лекція № 9. ОПИС АЛГОРИТМУ ТА РОЗРОБЛЕННЯ ПРОГРАМИ ЗНАХОДЖЕННЯ ПРОСТОГО ЧИСЛА ЗА ЙОГО НОМЕРОМ У ПОСЛІДОВНОСТІ ВСІХ ПРОСТИХ ЧИСЕЛ З ВИКОРИСТАННЯМ ПРОЦЕДУР З ПАРАМЕТРАМИ І ТЕХНОЛОГІЮ НИЗХІДНОГО ПРОЕКТУВАННЯ

Число називається простим, якщо воно має рівно два різних дільники: одиницю і саме число. Числа, які мають більшу кількість дільників, називаються складеними. Таким чином, якщо ми вміємо розкладати числа на множники, то ми вміємо і перевіряти числа на те, чи є вони простими. Наприклад:

```
bool isprime(int n){
    if(n==1) // 1 -  просте число
        return false;
    // перебираємо можливі дільники від 2 до sqrt (n)
    for(d=2; d*d<=n; d++){
        // якщо число розділилося без остачі, то воно є складеним
        if(n%d==0)
            return false;
        }
    // якщо немає нетривіальних дільників, то число є простим
    return true;
    }
```

Час роботи такого тесту зростає експоненціально щодо бітової довжини n. Цей тест називається перевіркою перебору дільників.

Якщо потрібно знайти всі прості числа на досить широкому інтервалі, то першим бажанням, напевно, буде протестувати кожне число з інтервалу індивідуально. На щастя, якщо у нас достатньо пам'яті, можна використовувати більш швидкі і прості «алгоритми решета». У цій лекції розглянуто два з них: класичне решето Ератосфена, відоме ще стародавнім грекам, і решето Аткіна – найбільш досконалий сучасний алгоритм цієї групи.

Контрольні запитання

1. Яке число називається простим?
2. Які переваги алгоритму перевірки перебору дільників?
3. Поясніть принцип роботи класичного решета Ератосфена?
4. Які переваги алгоритму решета Аткіна?

ПРИКЛАД ТЕСТОВИХ ЗАДАЧ МОДУЛЯ 1

1. У якому випадку можна не використовувати фігурні дужки в операторі вибору if?

 якщо в тілі оператора if немає жодного оператора
 якщо в тілі оператора if два і більше операторів
 немає правильної відповіді
 якщо в тілі оператора if усього один оператор

2. Як значення буде надруковано при виконанні вказаного нижче фрагмента програми?

```
int x = 3; switch (x) {case 0: int x = 1; cout << x; break;
   case 3: cout << x; break; default: x = 2; cout << x; }
```

 0
 2
 нічого не друкується, програма взагалі не буде працювати
 3
 1

3. Чому буде дорівнювати змінна a після виконання наступного фрагмента коду?

```
int a; for (a = 0; a <10; a ++) {}
```

 10
 9
 1
 фрагмент помилковий

4. Форма запису і коментарі до циклу:

// форма запису оператора циклу do while:
do // початок циклу do while
{/ * блок операторів * /;}
while {/ * умова виконання циклу * /} // кінець циклу do while

 коректні
 неправильні

5. Який з наступних операторів є оператором порівняння двох змінних?
 =
 equal
 : =
 ==

6. Строкові типи даних у С ++:
 рядки в С ++ подаються як масиви елементів типу char, що закінчуються термінатором рядка – символом з нульовим значенням ('')
 рядки в С ++ подаються як масиви елементів типу char, що закінчуються термінатором рядка – символом з нульовим значенням '0'
 рядки в С ++ подаються як масиви елементів типу char, що закінчуються термінатором рядка – символом з нульовим значенням ('/ 0')

7. Оберіть правильне (повне) визначення функції:
 void funct (x) {cout << "Hello"}
 void funct (int) {cout << "Hello"}
 int funct (int x) {return x = x + 1;}
 int funct ();

Змістовий модуль 2. СТРУКТУРИ ДАНИХ І АЛГОРИТМИ

Міжпредметні зв'язки: зв'язок з елементами знань і умінь таких навчальних дисциплін, як „Дискретна математика" і „Теорія алгоритмів".

Мета лекцій змістового модуля: ознайомити з деякими структурами даних і алгоритмами організації програм, навчити оцінювати обсяг витрачених ресурсів пам'яті і дискового простору. Увагу також приділено надійності і достовірності рішень, їх стабільності.

Опорні поняття: модуль, динамічна пам'ять, адреса пам'яті, абсолютні змінні, покажчик, спискові структури даних

Посилання на приклади та більш детальне пояснення розглянутих на лекціях питань за такими темами:

– принципи модульного програмування: призначення модулів, структура модулів, використання модулів [9];

– приклад використання модулів: концепція модульного програмування як наступний етап розвитку програмування [9];

– методології розроблення програм: послідовність розроблення, життєвий цикл та його етапи, засоби, що впливають на якість, опис алгоритму та розроблення програми «редактор текстів» [10];

– принципи адресації пам'яті на ПК у реальному режимі роботи центрального мікропроцесора: абсолютні адреси, сегменти та зсуви адрес, структура образу (.exe) файла, розподіл пам'яті при виконанні програми [9, 10];

– поняття абсолютних змінних: приклади використання абсолютних змінних у програмах, призначення прапорців клавіатури та доступ до них [11];

– поняття покажчика: його оголошення, типізовані та нетипізовані покажчики та операції над ними [12,13];

– використання покажчиків для доступу до динамічної пам'яті [14];

– виділення та вивільнення динамічної пам'яті: стандартні функції для роботи з адресами [1, 12];

– приклади використання покажчика на масив і масив покажчиків, використання покажчиків при опрацюванні структур [1, 6];

– спискові структури даних: визначення лінійного списку та його різновидів, робота з лінійним списком [3];

– алгоритм роботи зі списком: алгоритм створення одноелементного списку [1, 11];

– алгоритм вставки елемента всередину списку, алгоритм видалення елемента з кінця списку [1 ,8];

– принципи модульного програмування: призначення модулів, структура модулів, використання модулів [1,13].

Лекція № 10. ПРИНЦИПИ МОДУЛЬНОГО ПРОГРАМУВАННЯ. ПРИЗНАЧЕННЯ МОДУЛІВ. СТРУКТУРА МОДУЛІВ. ВИКОРИСТАННЯ МОДУЛІВ

Модульне програмування – парадигма програмування, орієнтована на зменшення складності програмних продуктів і можливості перенесення окремих рішень з одних програмних проектів до інших. Побудована модульна архітектура підкреслює поділ функціональності програми на незалежні змінні модулі, такі, що кожен з них містить усе необхідне для реалізації одного аспекту функціональності. Модулі, як правило, включені до програми через інтерфейси.

Модульне програмування дозволяє зменшити обсяг вихідних текстів програм, зробити їх прозорішими, прискорити написання і тестування програм, зменшити витрати на супровід (експлуатацію) програм. Ознакою якості модульного програмування є організація програми, що складається з можливо більшої кількості модулів при можливо меншій кількості зв'язків між ними.

Виділяють два напрями вживання модулів. Перший – пов’язаний з формуванням бібліотек підпрограм, повністю готових до роботи. Це зручно в рамках будь-якої технології програмування. Достатньо підключити модуль до програми – і можна використовувати його елементи в цій програмі.

Другий напрям – конструювання програм великого розміру. У зв'язку з сегментною організацією пам'яті ПК існують обмеження на розмір програми.

Контрольні запитання

1. Що таке модульне програмування?
2. Що робить модульне програмування?
3. Скільки існує напрямів уживання модулів?
4. З чим пов’язаний перший напрям уживання модулів?
5. У чому проблема другого напряму?

Лекція № 11. ПРИКЛАД ВИКОРИСТАННЯ МОДУЛІВ. КОНЦЕПЦІЯ МОДУЛЬНОГО ПРОГРАМУВАННЯ ЯК НАСТУПНИЙ ЕТАП РОЗВИТКУ ПРОГРАМУВАННЯ

Наведемо приклад двомодульної програми.

```
/ * Модуль 1, головний * /
#include <stdio.h>
extern int min (int, int, int); / * знаходить мінімальне значення з 3 * /
extern int max (int, int, int); / * знаходить максимальне значення з 3 * /
void main ()
    {int a, b, c; a = 2; b = 10; c = 14;
    / * Надрукувати добуток мінімального і максимального чисел * /
    printf ( "% d \ n", max (a, b, c) * min (a, b, c)); / * Відповідь 28 * /}

/ * Модуль 2 * /
#include <stdio.h>
externintmin (int, int, int); // знаходить мінімальне значення з 3
extern int max (int, int, int); // знаходить максимальне значення з 3
int max (int a1, int b1, int c1)
```

```
    {if (a1> b1) {if (c1> a1) return c1; else return a1;}else
                 {if (b1> c1) return b1; else return c1;}}
int min (int a1, int b1, int c1) {if (a1 <b1)
    {if (c1 <a1) return c1; else return a1;}else
    {if (b1 <c1) return b1;else return c1;}}
```

Програма складається з двох модулів. У головному модулі містяться виклики функцій, які є у другому модулі. Як видно з тексту програми, при описі функцій ми використовуємо ключове слово extern. Нехай перша програма називається modul1, а друга – modul2. Тоді після першого етапу трансляції на диску з'являться об'єктні модулі modul1.obj і modul2.obj. На другому етапі трансляції відбувається об'єднання цих модулів і на диску з'являється модуль modul1.exe.

Контрольні запитання

1. Протестуйте на практиці розглянуту у лекції програму з іншими даними.
2. Яка концепція модульного проектування?
3. Що виводиться в першому модулі?
4. Яке ключове слово використовується в програмі?
5. Що відбудеться на другому етапі програми?

Лекція № 12. МЕТОДОЛОГІЇ РОЗРОБЛЕННЯ ПРОГРАМ. ЖИТТЄВИЙ ЦИКЛ ПРОГРАМИ ТА ЙОГО ЕТАПИ. ЗАСОБИ, ЩО ВПЛИВАЮТЬ НА ЯКІСТЬ ПРОГРАМИ

Методологія розроблення програмного забезпечення – сукупність методів, застосовуваних на різних стадіях життєвого циклу розроблення програмного забезпечення, що мають спільний філософський підхід, і відповідно до цього підходу дозволяють забезпечити найкращу ефективність процесів розроблення.

Поширені методології програмування: водоспадна модель (waterfall), макетування (prototyping), ітеративне та інкрементне розроблення (iterative and incremental development), спіральна модель (spiral model), швидке розроблення програмного забезпечення (rapid application development), екстремальне програмування (extreme programming), різні види методологій гнучкого розроблення (agile methodology).

Процес розроблення складається з множини підпроцесів, або дисциплін, деякі з яких показані нижче. У моделі водоспаду вони йдуть одна за одною, в інших аналогічних процесах їх порядок або склад змінюється. Розглянемо підпроцеси водоспадної моделі:

- аналіз вимог;
- специфікація програмного забезпечення;
- проектування програмного забезпечення;
- програмування;
- тестування програмного забезпечення;
- системна інтеграція;
- упровадження програмного забезпечення;
- супровід програмного забезпечення.

Контрольні запитання

1. Що таке методологія розроблення програм?
2. Які методології програмування найбільш поширені?
3. З яких підпроцесів складається процес розроблення водоспадної моделі?
4. Який найперший етап створення програми?
5. Який останній етап створення програми?

Лекція № 13. ПРИНЦИПИ АДРЕСАЦІЇ ПАМ'ЯТІ НА ПК У РЕАЛЬНОМУ РЕЖИМІ РОБОТИ ЦЕНТРАЛЬНОГО МІКРОПРОЦЕСОРА. АБСОЛЮТНІ АДРЕСИ, СЕГМЕНТИ ТА ЗСУВИ АДРЕС. СТРУКТУРА ОБРАЗУ (.EXE) ФАЙЛА. РОЗПОДІЛ ПАМ'ЯТІ ПРИ ВИКОНАННІ ПРОГРАМИ

У першому IBM PC використовувався процесор 8088, який міг виконувати 16-розрядні команди, застосовуючи 16-розрядні внутрішні регістри, і адресувати тільки 1 Мбайт пам'яті, використовуючи 20 розрядів для адреси. Все програмне забезпечення ПК спочатку було призначено для цього процесора; воно було розроблено на основі 16-розрядної системи команд і моделі пам'яті місткістю 1 Мбайт.

Усі програми, що виконуються в реальному режимі, повинні використовувати тільки 16-розрядні команди, 20-розрядні адреси і підтримуватися архітектурою пам'яті, розрахованою на місткість до 1 Мбайт.

Для програмного забезпечення цього типу зазвичай використовується однозадачний режим, тобто одночасно може виконуватися тільки одна програма. Немає ніякого вбудованого захисту для запобігання перезапису елементів пам'яті однієї програми або навіть операційної системи іншою програмою.

Врешті-решт, коди операндів (вхідні і вихідні) повинні десь розташовуватися. Наприклад, вони можуть знаходитися в пристроях вводу/виводу (непоширений випадок). Визначення місця положення операндів відбувається кодом команди. Причому існують різні методи, за

допомогою яких код команди може визначити, відкіля брати вхідний операнд і куди поміщати вихідний операнд. Ці методи називаються методами адресації. Пряма (вона ж абсолютна) адресація припускає, що операнд (вхідний чи вихідний) знаходиться в пам'яті за адресою, код якої знаходиться усередині програми відразу ж за кодом команди. Наприклад, команда може виконувати очистку (робити нульовим) вмісту комірки пам'яті з адресою 1000000. Код цієї адреси 1000000 буде розташовуватися в пам'яті, усередині програми в наступній адресі, яка йде у комірці пам’яті, що містить код команди очищення.

Контрольні запитання

1. Який процесор використовувався в першому IBM PC?
2. Які команди повинні використовувати програми, що виконуються в реальному режимі?
3. Який режим зазвичай використовується для програмного забезпечення цього типу?
4. Що таке методи адресації?
5. Що таке пряма адресація?

Лекція № 14. ПОНЯТТЯ АБСОЛЮТНИХ ЗМІННИХ. ПРИКЛАДИ ВИКОРИСТАННЯ АБСОЛЮТНИХ ЗМІННИХ В ПРОГРАМАХ. ПРИЗНАЧЕННЯ ПРАПОРЦІВ КЛАВІАТУРИ ТА ДОСТУП ДО НИХ. ПРИКЛАД ПРОГРАМИ, У ЯКІЙ НАВЕДЕНО СПОСІБ ВИКОРИСТАННЯ ПРАПОРЦІВ КЛАВІАТУРИ

Поєднання клавіш визначається для таких елементів управління, як контекстні меню, прапорці, перемикачі, кнопки тощо.

Однак спочатку розглянемо стандартні сполучення клавіш, визначені в операційній системі за замовчуванням. Як правило, вони відповідають за найважливіші дії з управління системою, тому рекомендуємо відразу ж запамятати їх.

Отже, розглянемо основні сполучення клавіш, що відповідають за найважливіші дії з управління операційною системою:

– поєднання клавіш Ctrl+F2 дозволяє швидко звернутися до меню;

– працюючи з рядком меню, Ви можете використовувати клавіші ← і → для переміщення між різними вкладками. Альтернативним варіантом є використання клавіш Shift+Tab і Tab відповідно;

– перебуваючи в меню, Ви можете використовувати клавіатуру для набору перших символів назви пунктів меню Це дозволить швидко виділити потрібний Вам пункт. Особливо зручним виявиться цей прийом у

випадку дуже об’ємних меню, які можуть навіть не поміщатися на одному екрані;

- для вибору пункту меню можна використовувати клавішу Enter;
- клавіша Esc може бути використана для закриття меню.

Однак для насправді допитливих користувачів розглянемо три найзагадковіші клавіші на клавіатурі: PrtScn, Scroll Lock.

Раніше ця клавіша справді робила те, про що каже її назва – надсилала текст поточного екрану на принтер. Сьогодні натискання клавіші PrtScn робить знімок усього екрана та копіює його до буфера обміну в пам’яті комп’ютера.

У більшості програм натискання клавіші Scroll Lock ні на що не впливає. У деяких програмах натискання клавіші Scroll Lock змінює режим роботи клавіш зі стрілками та клавіш Page Up і Page Down.

Контрольні запитання

1. Що дає поєднання клавіш Ctrl+F2?
2. Які клавіші використовуються для переміщення між різними вкладками?
3. Яка клавіша може бути використана для закриття меню?
4. Для чого слугує клавіша PrtScn?
5. Що робить клавіша Scroll Lock?

Лекція № 15. ПОНЯТТЯ ПОКАЖЧИКА. ЙОГО ОГОЛОШЕННЯ. ТИПІЗОВАНІ ТА НЕТИПІЗОВАНІ ПОКАЖЧИКИ ТА ОПЕРАЦІЇ НАД НИМИ

Покажчик – це змінна, яка містить адресу іншої змінної в пам’яті. Наприклад, якщо змінна a містить адресу змінної b, то це означає, що змінна a вказує на змінну b.

Оголошення покажчиків можна здійснити одним з таких способів:

<тип> *ptr;

<тип> *ptr = <змінна-покажчик>;

<тип> *ptr = &<ім’я змінної>.

При оголошенні покажчиків символ «*» може знаходитися перед ім’ям покажчика або відразу після оголошення типу покажчика і поширювати свою дію тільки на одну змінну-покажчик, перед яким він записаний.

У програмах на мовах високого рівня покажчики можуть бути типізованими і нетипізованими. При оголошені типізованого покажчика визначається й тип об’єкта в пам’яті, який адресується цим покажчиком. Хоча фізична структура адреси не залежить від типу й значення даних, які зберігаються за цим адресом, компілятор уважає покажчики на різні типи

такими, що мають різний тип. Таким чином, коли йде мова про типізовані покажчики, правильніше говорити не про єдиний тип даних „покажчик”, а про цілу сім’ю типів: „покажчик на ціле”, „покажчик на символ”. Нетипізований покажчик використовується для подання адреси, за якою містяться дані невідомого типу. Робота з нетипізованими покажчиками суттєво обмежена, вони можуть використовуватися тільки для збереження адреси, звертатися за такою адресою не можна. Основними операціями, в яких беруть участь покажчики, є присвоєння, отримання адреси, вибір.

Контрольні запитання

1. Що таке покажчик?
2. Які існують способи оголошення покажчика?
3. Які покажчики можуть бути?
4. Яка різниця між типізованим і нетипізованим покажчиками?
5. Основні операції, в яких беруть участь покажчики?

Лекція № 16. ВИКОРИСТАННЯ ПОКАЖЧИКІВ ДЛЯ ДОСТУПУ ДО ДИНАМІЧНОЇ ПАМ’ЯТІ

Доступ до виділених ділянок динамічної пам’яті, що називаються динамічними змінними, здійснюється тільки через покажчики. Час існування динамічних змінних – від початку створення до кінця програми або до явного вивільнення пам’яті. У мові С++ застосовують два способи роботи з динамічною пам’яттю. Перший з них дістався в спадщину від мови С і використовує сукупність функцій malloc(), другий – працює з операціями new і delete, які здійснюють динамічний розподіл і скасування з вищим пріоритетом, ніж інші функції.

Оператор **new** виділяє пам’ять і повертає її адресу. За допомогою оператора **delete** відбувається вивільнення пам’яті, на яку вказує змінна-покажчик.

Загальна форма запису оператора **new**:

змінна-покажчик = new тип змінної;.

Оператор **delete** має вигляд

delete [ ] змінна-покажчик; .

Динамічні масиви створюють за допомогою операції new, при цьому необхідно вказати їх тип і розмірність. Наприклад, для одновимірного масиву дійсних чисел, що має 100 елементів, треба записати:

```
int n = 100;
float *p = new float[n];
```

змінна-покажчик на float виділяє у динамічній пам’яті ділянку для розміщення 100 елементів дійсного типу.

Слід пам’ятати, що динамічні масиви при їх створенні не можна ні ініціювати, ні обнуляти.

Контрольні запитання

1. Як здійснюється доступ до виділених ділянок динамічної пам’яті?
2. Який час існування динамічних змінних?
3. Які способи існують для роботи з динамічною пам’яттю?
4. Що робить оператор new ?
5. Чи можна ініціювати динамічні масиви при їх створенні?

Лекція № 17. ВИДІЛЕННЯ ТА ВИВІЛЬНЕННЯ ДИНАМІЧНОЇ ПАМ’ЯТІ. СТАНДАРТНІ ФУНКЦІЇ ДЛЯ РОБОТИ З АДРЕСАМИ

У С++ об’єкти можна розміщати у пам’яті статично (під час компіляції) або динамічно (при виконанні програми, шляхом виклику функцій зі стандартної бібліотеки).

Динамічне виділення пам’яті у мові С++ здійснюється за допомогою оператора new. Оператор new здійснює пошук неперервної області пам’яті в області пам’яті, що називається некерованою кучею. Некерована куча – це структура даних, за допомогою якої реалізована пам’ять, що може бути виділена динамічно в ході виконання програми, а також це область пам’яті, зарезервована під цю структуру. Інакше, куча – це довгий відрізок адреси пам'яті, поділений на блоки різних розмірів, що йдуть підряд. Пам’ять у кучі поділяється на зайняту і вільну. Перед початком роботи програми вся пам’ять у кучі позначається як вільна. При виклику оператора динамічного виділення пам’яті у кучі відбувається пошук неперервного сегмента вільної пам’яті заданого розміру.

Час такого пошуку є значним і займає більшу частину часу, що необхідна для виконання операції динамічного виділення пам’яті. Якщо такий сегмент було знайдено в кучі, то він помічається як зайнятий, і програмі повертається адреса його початку, інакше – програмі повертається ознака відсутності такого сегмента в пам’яті, найчастіше – NULL. Якщо в ході виконання програми значення адреси початку цього сегмента втрачається, то зайняту пам’ять вивільнити буде неможливо. Якщо ця ситуація проявляється неодноразово, то це може призвести до вичерпання вільної пам’яті в системі. Коли динамічно виділена область пам’яті стає непотрібною, то її слід вивільнити за допомогою оператора вивільнення динамічно виділеної пам’яті. При його виклику область пам’яті, на яку вказує покажчик і яка була попередньо динамічно виділена

з кучі, позначається як вільна, і її можна буде в подальшому використовувати заново.

Для того щоб отримати значення змінної, на яку вказує даний покажчик, використовуємо оператор розіменування *:

```
int a = 123; int *pnt = &a; int b = *pnt;
//123 - значення, на яке вказує покажчик *pnt = 1234;
```

Контрольні запитання

1. За допомогою чого здійснюється динамічне виділення пам’яті?
2. Що таке некерована куча?
3. Що відбувається при виклику оператора динамічного виділення пам’яті?
4. Що відбувається, якщо сегмент пам’яті був знайдений?
5. Який оператор використовуємо для отримання значення змінної?

Лекція № 18. ПРИКЛАДИ ВИКОРИСТАННЯ ПОКАЖЧИКА НА МАСИВ І МАСИВ ПОКАЖЧИКІВ. ВИКОРИСТАННЯ ПОКАЖЧИКІВ ПРИ ОПРАЦЮВАННІ СТРУКТУР

У мові С++ масиви і покажчики пов’язані між собою: ім’я масиву визначається як покажчик-константа на початковий (нульовий) елемент масиву. Так, наприклад, при оголошенні одновимірного масиву у вигляді int mas [20]; його ім’ям буде mas, а покажчиком на адресу початкового елемента масиву – &mas[0].

Існує два способи доступу до елементів масиву:

– з використанням індексу елемента масиву, наприклад mas[2] або mas[i];

– з використанням адресного виразу, тобто виразу з покажчиком на масив, наприклад *(mas + 2) або *(mas + i).

Ім’я покажчика на масив можна записати так:

```
int mas [20];
int *ptr1;
ptr1 = mas;      //ptr1 = &mas[0];
```

тут вирази &mas[0] і mas – еквівалентні.

Як параметр структуру можна передавати у функцію трьома способами: за значенням; за адресою; за посиланням.

При передаванні структури за адресою передається тільки адреса структури в пам’яті (покажчик на структуру). Цей спосіб є більш ефективним, коли для полів структури потрібно виділяти багато пам’яті (розмір покажчика може бути набагато менше розміру структурної змінної).

Покажчик на структуру, який передається у функцію, може бути двох типів:

– некерований покажчик (*);

– керований покажчик (^). У цьому випадку структура повинна оголошуватись з ключовим словом ref.

Контрольні запитання

1. Як у С++ пов’язані масиви та покажчики?
2. Назвіть способи доступу до елементів масиву.
3. Як можна передати структуру як параметр?
4. Що передається при передаванні структури?
5. Назвіть типи покажчиків на структуру.

Лекція № 19. СПИСКОВІ СТРУКТУРИ ДАНИХ. ВИЗНАЧЕННЯ ЛІНІЙНОГО СПИСКУ ТА ЙОГО РІЗНОВИДІВ. РОБОТА З ЛІНІЙНИМ СПИСКОМ

Використання покажчиків є ефективним способом побудови динамічних структур даних. Розміри динамічних структур даних визначаються і можуть зменшуватися під час виконання. Основними різновидами динамічних структур є лінійні списки, дерева (нелінійні списки) і динамічні масиви.

Зв'язний лінійний список – це сукупність однотипних компонентів, які послідовно зв'язані між собою за допомогою покажчиків. Кожен компонент списку, крім останнього, містить покажчик на наступний (або на наступний попередній) компонент. Доступ до першого компонента здійснюється за допомогою покажчика на нього, а доступ до кожного наступного компонента – з використанням покажчика, який зберігається у попередньому компоненті. Перший компонент списку називається його вершиною, або головою. Над зв'язними лінійними списками виконуються такі дії: додавання нового компонента на початок списку; додавання нового компонента в кінець списку; вставка нового компонента між двома наявними компонентами списку,. видалення компонента зі списку.

Зв'язні лінійні списки поділяють на такі різновиди: однозв'язний лінійний список; двозв'язний лінійний список; однозв'язний циклічний список; двозв'язний циклічний список.

Однозв'язний лінійний список – це список, в якому попередній компонент посилається на наступний. Двозв'язний лінійний список – це список, в якому попередній компонент посилається на наступний, а наступний – на попередній. Однозв'язний циклічний список – це однозв'язний лінійний список, в якому останній компонент посилається на

перший. Двозв'язний циклічний список – це двозв'язний лінійний список, в якому останній компонент посилається на перший, а перший компонент – на останній.

Контрольні запитання

1. Основні різновиди динамічних структур.
2. Що таке зв’язний лінійний список?
3. Як називається перший компонент списку?
4. Які дії виконуються над зв’язними списками?
5. Чим відрізняються однозв’язний і двозв’язний лінійні списки?

Лекція № 20. АЛГОРИТМ РОБОТИ ЗІ СПИСКОМ. АЛГОРИТМ СТВОРЕННЯ ОДНОЕЛЕМЕНТНОГО СПИСКУ

При описі алгоритмів, що використають такі структури, прийнято спеціальну термінологію; так, ми поміщаємо елемент на верх стека або видаляємо верхній елемент. Унизу стека перебуває найменш доступний елемент, і він не видаляється доти, доки не будуть видалені всі інші елементи. Часто говорять, що елемент опускається (push down) у стек або що стек піднімається (pop up), якщо видаляється верхній елемент.

Розглянемо алгоритм роботи зі списком.

Крок 1. Описання структурного типу, за допомогою якого можна створити наш однозв’язний список.

Крок 2. У програмі (зазвичай у функції main ()) визначити покажчик на початок майбутнього списку:

```
List *u = NULL;
```

Крок 3. Виконати початкове заповнення списку.

Крок 4. Створити перший елемент:

```
u = new List;  // Виділяємо пам’ять під елемент списку
u->d.a = 3;    // Заповнюємо поля з даними
u->next = NULL;// Покажчик на наступний елемент пустий
```

Крок 5. Продовжити формування списку, додаючи нові елементи в його кінець.

Цей процес продовжити доти, доки не буде сформований весь список.

Алгоритм створення одноелементного списку.

Крок 1. Ініціалізувати початок списку.

Крок 2. Ініціалізувати кінець списку.

Крок 3. Записати ознаку того, що перший елемент є останнім.

Для опису алгоритмів використовується таке оголошення:

```
struct Single_List {
//структура даних
 int Data; //інформаційне поле
Single_List *Next; //адресне поле };
 Single_List *Head; //покажчик на перший елемент списку
Single_List *Current; //покажчик на поточний елемент списку (за необхідності)
```

Створення односпрямованого списку. Для того щоб створити список, потрібно створити спочатку перший елемент списку, а потім за допомогою функції додати до нього інші елементи. За відносно невеликих розмірів списку найбільш витонченим і красивим є використання рекурсивної функції. Додавання може виконуватися як в початок, так і в кінець списку. Розглянемо додавання до кінця списку:

```
// створення односпрямованого списку (додавання в кінець)
void Make_Single_List(int n,Single_List** Head)
{ if (n > 0) { (*Head) = new Single_List();
//виділяємо пам’ять під новий елемент
cout << "Введіть значення "; cin >> (*Head)->Data;
//уводимо значення інформаційного поля
(*Head)->Next=NULL; //обнулення адресного поля
 Make_Single_List(n-1,&((*Head)->Next)); } }
```

Контрольні запитання

1. Яка перша дія для створення списку?
2. За допомогою чого можна створити однозв’язний список?
3. У якій функції визначається покажчик на початок майбутнього списку?
4. Як створюється елемент списку?
5. Як створити одноелементний список?

Лекція № 21. АЛГОРИТМ ВСТАВКИ ЕЛЕМЕНТА ВСЕРЕДИНУ ОДНОСПРЯМОВАНОГО СПИСКУ. АЛГОРИТМ ВИДАЛЕННЯ ЕЛЕМЕНТА З КІНЦЯ ОДНОСПРЯМОВАНОГО СПИСКУ

У динамічні структури легко додавати елементи, тому що для цього достатньо змінити значення адресних полів. Включення першого і наступних елементів списку відрізняються один від одного. Тому у функції, що реалізує дану операцію, спочатку здійснюється перевірка, на яке місце вставляється елемент. Далі реалізується відповідний алгоритм додавання (рис. 21.1):

```
/*включення елемента з заданим номером до односпрямованого списку*/
Single_List* Insert_Item_Single_List(Single_List* Head, Int Number, int DataItem)
{ Number--; Single_List* NewItem=new(Single_List);
NewItem->Data=DataItem;
NewItem->Next=NULL;
if (Head==NULL)
 {//список порожній
Head=NewItem;//створюємо перший елемент списку
} else
{//список не порожній
Single_List* Current=Head;
for(int i=1; iNext!=NULL; i++)
Current=Current->Next;
if (Number==0)
{ //вставляємо новий елемент на перше місце
NewItem->Next=Head;
Head=NewItem; }
else {//вставляємо новий елемент на не перше місце
 if (Current->Next!=NULL) NewItem->Next=Current->Next;
 Current->Next=NewItem; } }
return Head; }
```

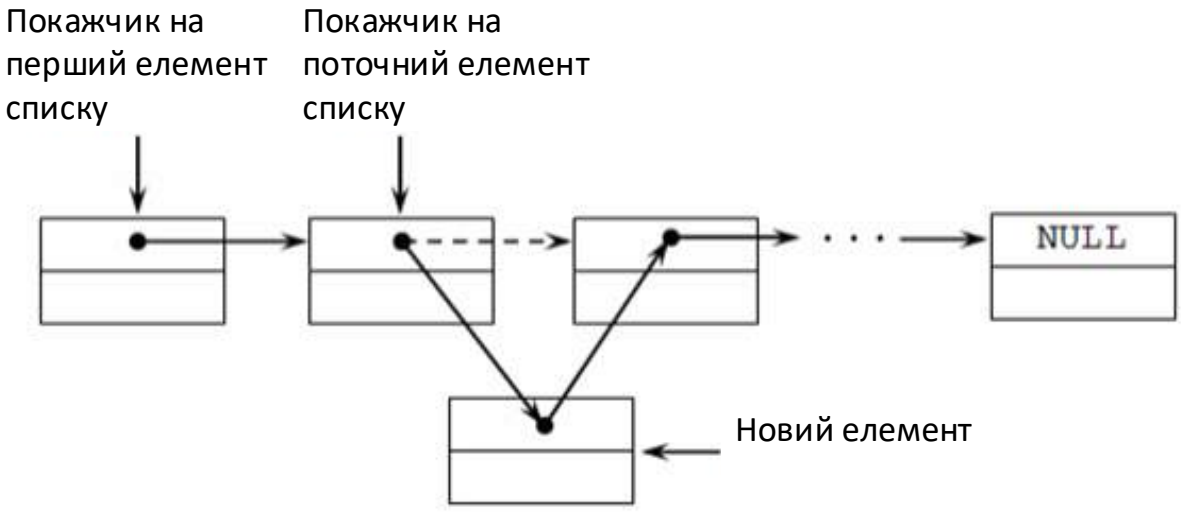

Рис. 21.1. Додавання елемента з односпрямованого списку

Виключення елемента з односпрямованого списку. З динамічних структур можна виключати елементи, тому що для цього достатньо змінити значення адресних полів. Операція виключення елемента односпрямованого списку здійснює виключення елемента, на який встановлено покажчик поточного елемента. Після виключення покажчик

поточного елемента встановлюється на попередній елемент списку або на новий початок списку, якщо видаляється перший. Алгоритми виключення першого і наступних елементів списку відрізняються один від одного. Тому у функції, що реалізує дану операцію, здійснюється перевірка, який об'єкт був видалений. Далі реалізується відповідний алгоритм виключення (рис. 21.2):

```
//виключення елемента за заданим номером з односпрямованому списку
Single_List* Delete_Item_Single_List(Single_List* Head, int Number)
{ Single_List *ptr;//допоміжний покажчик
Single_List *Current = Head;
 for (int i = 1; i < Number && Current != NULL; i++) Current = Current->Next;
if (Current != NULL)
{//перевірка на коректність
 if (Current == Head)
{//виключення першого элемента
Head = Head->Next; delete(Current); Current = Head; }
 else
{//виключення не першого елемента
ptr = Head; while (ptr->Next != Current)
ptr = ptr->Next; ptr->Next = Current->Next;
delete(Current); Current=ptr; } } return Head; }
```

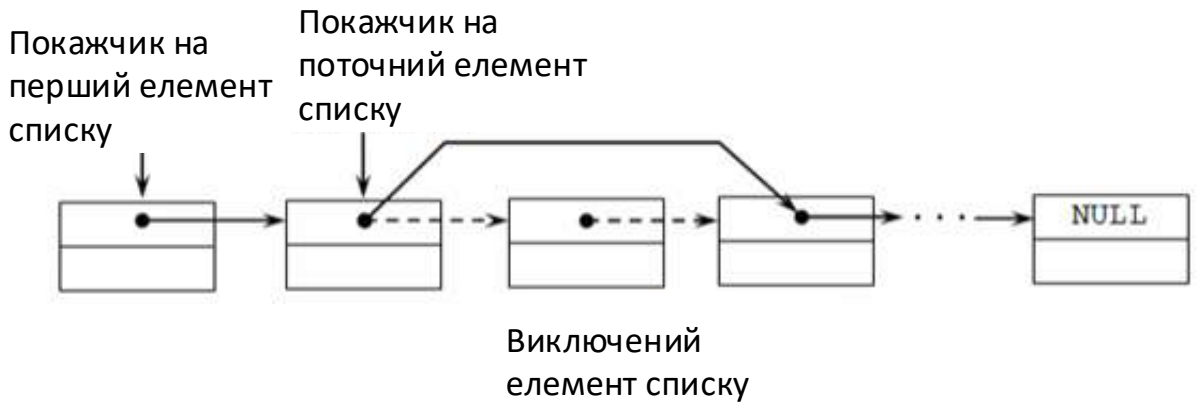

Рис. 21.2. Виключення елемента з односпрямованого списку

Контрольні запитання

1. Сформуйте блок-схему наведених алгоритмів.
2. Скільки пам’яті вивільняється при виключені елемента (див. рис. 21.2)?
3. Елементи односпрямованого списку розміщуються разом у пам’яті чи знаходяться у різних місцях?
4. Як додається новий елемент списку (опишіть словесно)?

ПРИКЛАД ТЕСТОВИХ ЗАДАЧ МОДУЛЯ 2

1. Прокоментуйте рядки наведеного коду.

Код	Коментар
struct List	
{Int x;	
List * Next, * Head; };	
void Add (int x, List * & MyList)	
{List * temp = new List;	
temp-> x = x;	
temp-> Next = MyList-> Head;	
MyList-> Head = temp; }	

2. Які дані і де вони будуть зберігатися в разі виклику наведеної вище функції Add:

```
List * MyList = new List;
MyList-> Head = NULL;
for (int i = 0; i <10; i ++) Add (i, MyList);
```

3. У програмі присутні такі операнди:

```
int var = 123;
int * ptrvar = & var;
```

Який з рядків виводить адресу змінної var?

- cout << & var << endl;
- cout << ptrvar << endl;
- cout << var << endl;
- cout << * ptrvar << endl;

4. Дано код:

```
void fun5 (int & x, int * y) {????????????? ; } // 1
void main ()
{int a = 5, * b = new int;
fun5 (a, b);
cout << endl << a << "" << (* b) << endl;
getch (); }
```

Що записати в тексті функції в рядку «// 1», щоб вивести 5 15:

- y = x + 10;
- * y = * x + 10;

□ * y = x + 10;
□ y = * x + 10;
□ помилка

5. При визначенні структури необхідно використовувати таке ключове слово:
□ record
□ struct
□ object
□ structure

Змістовий модуль 3. ГРАФІЧНИЙ РЕЖИМ

Міжпредметні зв'язки: зв'язок із елементами знань і умінь таких навчальних дисциплін, як „Теорія алгоритмів" та „Основи програмування".

Мета лекцій змістового модуля: ознайомити з методами та алгоритмами сучасної комп'ютерної графіки. Проаналізувати основні засоби формування зображення двовимірних об'єктів та анімації нескладних об'єктів. Освоїти практичні задачі графічних програм мовою С++ з використанням інтегрованого середовища на прикладі Microsoft Visual Studio

Опорні поняття: система координат, піксель, графічні примітиви, фрактальні зображення, анімація.

Посилання на приклади та більш детальне пояснення розглянутих на лекціях питань за такими темами:

– графіка у консольному вікні. Графічна система координат. графічні драйвери та режими сучасних відеокарт і їх характеристики та можливості. Приклад нескладної програми, яка використовує графічний режим [5,17];

– координатна площина. Поняття пікселя, кольору фону, використання кольорів і стилів. Загальні положення. Керування кольором і стилями. Заливка. Використання відтінків, що потребує доволі складної техніки керування кольоровими палітрами [15, 17];

– вивід зображення крапок та ліній. Приклад програми, яка установить білий колір для фону та виведе на екран відрізок прямої червоного кольору. Вивід зображення крапок на різному фоні [15, 17];

– графічні примітиви, які можна відобразити на екрані за допомогою окремих процедур. Вивід зображення кривих ліній. Малювання кіл, дуг, сектору кола, зафарбованого та не зафарбованого еліпса, еліптичного сектора. Приклад програми, яка зображує концентричні кола випадково

вибраними кольорами, діаметр яких спочатку збільшується, а згодом – зменшується [16, 17];

– вивід зображення зафарбованих і незафарбованих багатокутників. Побудова зображення «вікна». Побудова зображення паралелепіпеда. Малювання багатокутників. Заливка замкнутих областей [16, 17];

– побудова графіків функцій перетворення координат і об’єктів. Спрайт, його створення, етапи роботи. Анімаційні ефекти. Фрактальні зображення [16].

Лекція № 22. ГРАФІКА У КОНСОЛЬНОМУ ВІКНІ. ГРАФІЧНА СИСТЕМА КООРДИНАТ. ГРАФІЧНІ ДРАЙВЕРИ ТА РЕЖИМИ СУЧАСНИХ ВІДЕОКАРТ ТА ЇХ ХАРАКТЕРИСТИКИ І МОЖЛИВОСТІ. ПРИКЛАД НЕСКЛАДНОЇ ПРОГРАМИ, ЯКА ВИКОРИСТОВУЄ ГРАФІЧНИЙ РЕЖИМ

Консольне вікно Windows є інтерфейсом, який надає Windows для введення і виведення даних додатків, які працюють у символьному режимі, тобто таким, що не мають власного графічного інтерфейсу.

Так, програми, які створюються на початковому етапі навчання, зазвичай виконуються в командному вікні. Вони використовують стандартні «текстові» функції введення/виведення printf (...), scanf (...), getchar (...) тощо або стандартні потоки cout і cin введення/виведення C ++. Однак бібліотека функцій не обмежується тільки ними.

Є можливість використовувати функції Windows API для керування консольним введенням – виведенням.

Відеокарта – електронний пристрій, частина комп'ютера, призначена для генерації та оброблення зображень з подальшим їх виведенням на екран периферійного пристрою.

Відеокарта зазвичай є платою розширення і вставляється у слот розширення, універсальний (PCI-Express, PCI, ISA, VLB, EISA, MCA) або спеціалізований (AGP), проте відеокарта може бути вбудованою у материнську плату як у вигляді окремого елемента, так і як складової частини північного мосту чипсету або центрального процесора. Відповідно вставлювана називається дискретною, а вбудована – інтегрованою.

Сучасні відеокарти не обмежуються лише звичайним виведенням зображень, вони мають убудований графічний мікропроцесор, котрий може здійснювати додаткове їх оброблення, звільняючи від цих задач центральний процесор. Також процесор і відеокарта працюють разом і є залежними один від одного. Останнім часом разом зі зростанням обчислювальних потужностей графічних процесорів має місце тенденція використовувати обчислювальні можливості графічного процесора для вирішення неграфічних задач .

Контрольні запитання

1. Що таке консольне вікно?
2. Які функції можна використовувати для керування консольним введенням-виведенням?
3. Що таке відеокарта?
4. Чим відрізняються дискретна й інтегрована відеокарти?
5. Що робить убудований графічний мікропроцесор?

Лекція № 23. КООРДИНАТНА ПЛОЩИНА. ПОНЯТТЯ ПІКСЕЛЯ. КОЛІР ФОНУ. ВИКОРИСТАННЯ КОЛЬОРІВ І СТИЛІВ. ЗАГАЛЬНІ ПОЛОЖЕННЯ. КЕРУВАННЯ КОЛЬОРОМ І СТИЛЯМИ. ЗАЛИВКА. ВИКОРИСТАННЯ ВІДТІНКІВ, ЩО ПОТРЕБУЄ ДОВОЛІ СКЛАДНОЇ ТЕХНІКИ КЕРУВАННЯ КОЛЬОРОВИМИ ПАЛІТРАМИ

Піксель – це неподільне зерно зображення, тобто мінімальний елемент зображення на екрані, який може бути зґенерований комп'ютером. Кожний піксель має свій колір. Сукупність пікселів різного кольору утворює зображення, тобто растрове зображення в пам'яті комп'ютера – це набір даних про колір пікселів, упорядкованих за рядками. Для опису розташування пікселів використовують різноманітні системи координат, найчастіше – систему цілих координат (x, y) із координатами (0,0) у лівому верхньому куті екрана, де x – номер рядка, y – номер пікселя в рядку.

Для створення кольорового зображення в конструкцію монітора входять три електронні гармати („червона", „зелена", „синя"). Керування вертикальною і горизонтальною розгортками у них спільне, а керування яскравістю кольору – роздільне.

Колір – це один із факторів світлового випромінювання. Наведені характеристики дії кольорів на психіку людини є основою для поділу кольорів на холодні й теплі. Холодні кольори (зелений, синій) заспокоюють, полегшують напруження очей, теплі (червоний, жовтий, оранжевий) потрібно розглядати як активні, динамічні. Холодні кольори створюють відчуття збільшеного простору (предмети ніби віддаляються), теплі – навпаки, зменшують зоровий простір. Отже, перша вимога при роботі з кольором – це чіткість і зручність сприйняття зображення, що забезпечується оптимальним підбором його основних характеристик.

Контрольні запитання

1. Які стилі використовуються для формування тексту?
2. Що таке піксель?
3. Які електронні пристрої входять у конструкцію монітора?

4. Які правила оптимального поєднання кольорів Ви знаєте?
5. Назвіть найвідоміші API для роботи з графікою.

Лекція № 24. ВИВІД ЗОБРАЖЕННЯ КРАПОК І ЛІНІЙ. ПРИКЛАД ПРОГРАМИ, ЯКА УСТАНОВИТЬ БІЛИЙ КОЛІР ДЛЯ ФОНУ ТА ВИВЕДЕ НА ЕКРАН ВІДРІЗОК ПРЯМОЇ ЧЕРВОНОГО КОЛЬОРУ. ВИВІД ЗОБРАЖЕННЯ КРАПОК НА РІЗНОМУ ФОНІ

Розглянемо растрові алгоритми для відрізків прямої лінії. Припустимо, що задані координати (x1, y1 - x2, y2) кінців відрізка прямої. Для виведення лінії необхідно зафарбувати в певний колір усі пікселі вздовж лінії. Для того щоб зафарбувати кожен піксель, необхідно знати його координати.

Найбільш просто намалювати відрізок горизонтальної лінії:

for (x = xl; x <= x2; x ++)// SetPixel(x, yl, Color.Black);

Обчислення поточних координат пікселя тут виконується як приріст по x (необхідно, щоб x1 <= x2), а вивід пікселя забезпечується функцією SetPixel(). Оскільки в мові C, C ++ для назви функції не можна використовувати кирилицю, то будемо далі використовувати її як коментар.

Аналогічно малюється відрізок вертикалі:

for (y = yl; y <= y2; y ++) // SetPixel(xl, y, Color.Black);

Як бачимо, в циклі виконуються найпростіші операції над цілими числами – приріст на одиницю і перевірка на "<=". Тому операція малювання відрізка виконується швидко і просто. Її використовують як базову операцію для інших операцій, наприклад, в алгоритмах заповнення площини полігонів.

Функція LineTo малює лінії від поточної позиції до координати, заданої параметрами x, y або point з використанням поточного пера:

BOOL LineTo (int x, int y); або BOOL LineTo (POINT point);

Змінити поточну позицію можна за допомогою функції:

CPoint MoveTo (int x, int y); або CPoint MoveTo (POINT point);

наприклад: MoveTo (10,20); LineTo (100,50);

Контрольні запитання

1. Що треба для того, щоб зафарбувати кожен піксель?
2. Як зобразити крапку за допомогою графіки?
3. Як зафарбувати тон за допомогою графіки?
4. Які операції виконуються в циклі for (y = yl; y <= y2; y ++)?
5. Як вивести лінію?

Лекція № 25. ГРАФІЧНІ ПРИМІТИВИ, ЯКІ МОЖНА ВІДОБРАЗИТИ НА ЕКРАНІ ЗА ДОПОМОГОЮ ОКРЕМИХ ПРОЦЕДУР. ВИВІД ЗОБРАЖЕННЯ КРИВИХ ЛІНІЙ. МАЛЮВАННЯ КІЛ, ДУГ, СЕКТОРА КОЛА, ЗАФАРБОВАНОГО ТА НЕЗАФАРБОВАНОГО ЕЛІПСА, ЕЛІПТИЧНОГО СЕКТОРА

Графічні примітиви – це елементи простої форми, з яких будують більш складні зображення. Будь-який малюнок або креслення можна розглядати як сукупність графічних примітивів: крапок, ліній, кіл, дуг та ін. Для того щоб на екрані з'явився малюнок, програма повинна забезпечити креслення кожного з графічних примітивів, з яких він складається.

Для малювання прямокутників поточним пером використовуються функції

BOOL CDC :: Rectangle (int upX, int upY, int lowX, int lowY);

BOOL CDC :: RoundRect (int upX, int upY, int lowX, int lowY, int curveX, int curveY);

Перша функція малює звичайний прямокутник, а друга - прямокутник з округленими кутами. Параметри curveX і curveY задають ширину та висоту еліпса, що визначають дугу для округлених кутів. Намальовані фігури автоматично заповнюються за допомогою поточного пензлика контексту пристрою. Наприклад:

dc.Rectangle (10,5,50,15);

dc.RoundRect (100,5,200,50,10,20);

Для малювання еліпсів поточним пером використовується функція

BOOL CDC :: Ellipse (int upX, int upY, int lowX, int lowY);

Координати (upX, upY) і (lowX, lowY) задають лівий верхній і правий нижній кути прямокутника, в який буде вписаний еліпс. Якщо потрібно намалювати коло, то задається описаний квадрат. Фігура заповнюється за допомогою поточної кисті контексту пристрою.

Для малювання сектора кола призначена функція Pie ():

BOOL CDC :: Pie (HOC hDC, int nLeftRect, int riTotRect, int nRightRect, int nBottomRect, int nXStart, int nYStart, int nXEnd, int nYEnd);

Параметри та приклади використання розглянуті у довідковій системі та на лекції.

Контрольні запитання

1. Як намалювати прямокутник у лівому верхньому куті вікна?
2. Що таке графічні примітиви?
3. Як можна змінити колір фону?
4. Як намалювати коло?
5. За допомогою яких функцій будується еліпс?

Лекція № 26. ВИВІД ЗОБРАЖЕННЯ ЗАФАРБОВАНИХ І НЕЗАФАРБОВАНИХ БАГАТОКУТНИКІВ. ПОБУДОВА ЗОБРАЖЕННЯ «ВІКНА». ПОБУДОВА ЗОБРАЖЕННЯ ПАРАЛЕЛЕПІПЕДА. МАЛЮВАННЯ БАГАТОКУТНИКІВ. ЗАЛИВКА ЗАМКНУТИХ ОБЛАСТЕЙ

Функції для створення та зафарбовування багатокутників:

– DrawPoly(n: word; P) – малює багатокутник, кількість сторін у якого – n, а інформація про вершини зберігається в нетипізованому параметрі P. Як P зручніше за усе використовувати масив із записів, кожен з яких містить поля x, y;

– FillPoly(n: word; P) – зафарбований багатокутник.

Виділення вікна потрібно при визначенні тієї частини сцени, що повинна бути виведена на екран дисплея. Нехай вікно обмежене лініями. По черзі для кожного багатокутника перевіряється розташування його вершин і ребер щодо границь вікна. Так, для багатокутника при відсіканні по границі змінюється множина вершин у порядку обходу за годинниковою стрілкою. Після закінчення перегляду стосовно всіх границь у вікні з'являються вершини, що залишилися в списку.

Установлює режим зафарбовування багатокутників функція

int CDC :: SetPolyFillMode (int nPolyFillMode);

де параметр nPolyFillMode відповідає за можливі режими зафарбовування.

Наступна функція забезпечує зафарбовування довільних областей з використанням поточного пензлика:

BOOL CDC :: ExtFloodFill (int x, int y, COLORREF crColor, UINT nFillType);

де параметр crColor визначає або колір кордону області, або колір самої області, що залежить від режиму визначення області зафарбовування, що задається параметром nFillType.

Для установки режиму малювання або заливки використовується функція

int CDC :: SetROP2 (int nDrawMode);

де nDrawMode визначає режим растрових (піксельних) операцій.

Контрольні запитання

1. Як намалювати багатокутник за допомогою графіки?
2. Як зафарбувати багатокутник?
3. Як побудувати зображення «вікно»?
4. Як залити замкнуту область?
5. Як побудувати зображення паралелепіпеда?

Лекція № 27. ПОБУДОВА ГРАФІКІВ ФУНКЦІЙ. ПЕРЕТВОРЕННЯ КООРДИНАТ І ОБ'ЄКТІВ. СПРАЙТ, ЙОГО СТВОРЕННЯ, ЕТАПИ РОБОТИ. АНІМАЦІЙНІ ЕФЕКТИ. ФРАКТАЛЬНІ ЗОБРАЖЕННЯ

Комп'ютерна анімація — мистецтво створення рухомих зображень за допомогою комп'ютерів. Вона є підрозділом комп'ютерної графіки й анімації. На відміну від більш загального поняття «графіка», що належить як до нерухомих, так і до рухомих зображень, методи комп'ютерної анімації присвячено питанням тільки рухомих.

Фрактальна графіка — технологія створення зображень на основі фракталів. Фрактальна графіка базується на фрактальній геометрії.

Найвідомішими фрактальними об'єктами є дерева: від кожної гілки відходять менші, схожі на неї, від них – ще менші. За окремою гілкою математичними методами можна відслідкувати властивості всього дерева. Фрактальні властивості мають такі природні об'єкти, як сніжинка, що при збільшенні виявляється фракталом; за фрактальними алгоритмами ростуть кристали та рослини.

Поява нових елементів меншого розміру відбувається за простим алгоритмом. Очевидно, що описати подібні об'єкти можна всього лише декількома математичними рівняннями!

Трикутники можна добудовувати аналогічним чином до нескінченності. Ми можемо отримати об'єкт будь-якого рівня складності, використовуючи простий алгоритм. При цьому нічого, крім самих рівнянь, які займають декілька байтів, у пам'яті комп'ютера зберігати нічого не треба. Уся інформація, необхідна для відтворення цього фрактала, займає всього лише десятки байтів.

Контрольні запитання

1. Що таке комп'ютерна анімація?
2. Що таке фрактал?
3. Що таке фрактальна графіка?
4. Назвіть найвідоміші фрактальні об'єкти.
5. Які природні об'єкти мають фрактальні властивості?

ПРИКЛАД ТЕСТОВИХ ЗАДАЧ МОДУЛЯ 3

1. Яка функція побудує точку?
   - Line
   - Circle
   - SetPixel
   - Rectangle
2. Яка з наведених функцій записана без помилок?
   - Rectangle (x1, x2, y1, y2);

- Rectangle (x1, y1);
- Rectangle (x1, y1, x2, y2);
- Rectangle (x1, y1, x2);

3. У результаті виконання поданого фрагмента алгоритма що буде намальовано?

Circle (100,100,45);
Circle (100,200,30);
Circle (300,300,70);
Circle (350,100,40);
Circle (130,300,60);

- 5 кіл
- 5 ліній
- 5 прямокутників
- 5 точок

4. Оголосіть масив для здійснення трансформації з рис. 27.1.

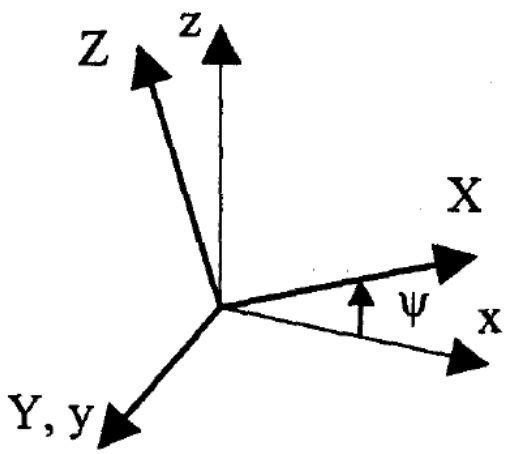

Рис. 27.1. Поворот осей на кут Ψ

5. Чому дорівнює одиниця виміру dpi? _____

6. Що називають структурою даних, яка містить інформацію про параметри і атрибути виведення графіки на пристрій (наприклад, дисплей або принтер)?

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Овсянник, В. Н. Язык С++ не для чайников : учеб. пособие / В. Н. Овсянник. – Харьков : Нац. аэрокосм. ун-т им. Н. Е. Жуковского "Харьков. авиац. ин-т", 2005. – 215 с.
2. Ковалюк, Т. В. Алгоритмізація та програмування : підручник / Т. В. Ковалюк. – Львів : Магнолія 2006, 2013. — 400 с.
3. Шаховська, Н. Б. Алгоритми і структури даних: посібник / Н. Б. Шаховська, Р. О. Голощук ; за ред. В. В. Пасічника. – Львів : Магнолія, 2010. – 215 с.
4. Ковалюк, Т. В. Основи програмування : підруч. для студ. з грифом МОН / Т. В. Ковалюк. – Київ : Видавнича група BHV, 2005. – 384 с.
5. Овсянник, В. Н. Использование Microsoft Visual Studio для разработки приложений Windows : учеб. пособие по лаб. практ. / В. Н. Овсянник. – Харьков : Нац. аэрокосм. ун-т им. Н. Е. Жуковского "Харьков. авиац. ин-т", 2014. – 84 с.
6. Долинский, М. С. Решение сложных и олимпиадных задач по программированию : учеб. пособие / М. С. Долинский. – СПб. : Питер, 2006. – 366 с.
7. Соколова, Е. В. Основы программирования и алгоритмические языки : учеб. пособие по лаб. практ. В 2 ч. – Ч. 1 / Е. В. Соколова, П. А. Лучшев, Н. Г. Мокляк. – Харьков : Нац. аэрокосм. ун-т им. Н. Е. Жуковского "Харьков. авиац. ин-т", 2006. – 111 с.
8. Соколова, Е. В. Основы программирования и алгоритмические языки : учеб. пособие по лаб. практ. В 2 ч. – Ч. 2 / Е. В. Соколова, П. А. Лучшев, И. Б. Туркин. – Харьков : Нац. аэрокосм. ун-т им. Н. Е. Жуковского "Харьков. авиац. ин-т", 2007. – 107 с .
9. Страуструп, Б. Язык программирования С++ : пер. с англ. / Б. Страуструп. – Издание специальное. – М. : Бином-Пресс, 2005. – 1104 с.
10. Шилдт, Г. Полный справочник по С++ : пер. с англ. / Г. Шилдт. – 4-е изд. – М. : ИД "Вильямс", 2004. – 800 с.
11. Седжвик, Р. Фундаментальные алгоритмы на С++. В 4 ч. – Ч. 1–4 : Анализ. Структуры данных. Сортировка. Поиск / Р. Седжвик. – Киев : DiaSoft, 2001. – 688 с.
12. Прата, С. Язык программирования С : лекции и упражнения : учебник : пер. с англ. / С. Прата. – СПб. : ДиаСофтЮП, 2002. – 896 с
13. Мозговой, М. В. С++ мастер-класс. 85 нетривиальных проектов, решений и задач / М. В. Мозговой. – СПб. : Наука и техника, 2007. – 272 с.
14. Караванова, Т. П. Основи алгоритмізації та програмування / Т. П. Караванова. – Київ : ФОРУМ, 2002. – 289 с.
15. Порев, В. Н. Компьютерная графика / В. Н. Порев. – СПб. : БХВ-Петербург, 2002. – 432 с.
16. Ласло, М. Вычислительная геометрия и компьютерная графика на С++ : пер. с англ. / М. Ласло. – М. : Издательство БИНОМ, 1997. – 304 с.

17. Журавчак, Л. М. Програмування компютерної графіки та мультимедійні засоби : навч. посіб. / Л. М. Журавчак, О. М. Левченко. – Львів : Вид-во Львівської політехніки, 2019. – 276 с.

ЗМІСТ

Навчальне видання

Погудін Андрій Володимирович
Погудіна Ольга Костянтинівна

АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ

Редактор С. П. Гевло

Зв. план, 2020
Підписано до видання 17. 06. 2020
Ум. друк. арк. 2,3. Обл.-вид. арк. 2,56. Електронний ресурс

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції сер. ДК № 391 від 30.03.2001