Лабораторний практикум з дисципліни „Технології програмування”, яка є складовою частиною професійної підготовки фахівців за напрямом 6.170101 «Безпека інформаційних і комунікаційних систем». Ці методичні рекомендації розроблені для виконання лабораторних робіт зі вказаної дисципліни студентами інституту комп’ютерних інформаційних технологій НАУ.

В результаті виконання лабораторних робіт у 5 та 6 семестрах студенти набувають навички: роботи в середовищі C++ Builder 6, а саме засвоюють методику роботи: із виключеннями, рядками різних типів, з покажчиками та списками покажчиків, зі звичайними масивами, зі структурами і класами, процесами і потоками, взаємодії із сервером MySQL, створення найпростіших запитів мовою SQL, створення таблиць та баз даних, введення, редагування, виведення та видалення даних, розглядають складні питання SQL-програмування: влаштовані функції, повнотекстовий пошук, транзакції, тимчасові таблиці.

В результаті виконання лабораторних робіт у 7 семестрі студенти ознайомлюються з переліком та порядком роботи засобів системи програмування мови Асемблер, засобів DOS, BIOS, засвоюють правила трансляції, редагування та виконання програм, набувають навичок у програмування та роботі з мікропроцесором комп’ютера.

З кожної роботи складається звіт, який надається викладачеві на наступному занятті під час захисту роботи.

Звіт з лабораторних робіт повинен бути оформлений відповідно до вимог ЄСКД. У звіті вказуються найменування та мета роботи, наводяться початкові дані індивідуального варіанта завдання. До звіту додаються програми та результати, отримані в результаті досягнення мети роботи.

Для оволодіння курсом “Технології програмування” студентам необхідно мати базові знання з програмування мовою C++, Асемблер та навички роботи із середовищем C++

Builder.

Знання та вміння, отримані під час вивчення даної дисципліни, будуть використані як базові для опанування переважної більшості дисциплін професійної та практичної підготовки фахівця напряму 6.170101 „Безпека інформаційних і комунікаційних систем”.

Порядок оформлення звіту з лабораторної роботи

Сторінки звіту слід нумерувати арабськими цифрами, додержуючись наскрізної нумерації по всьому звіту лабораторної роботи. Номер сторінки проставляють у правому нижньому куті сторінки без крапки в кінці. Номер сторінки на титульному аркуші не проставляється.

Текст звіту слід друкувати шрифтом Times New Roman, розмір 14 пт, міжрядковий відступ 1,5, розміри полів: ліве – 30 мм; верхнє – 20 мм; нижнє – 25 мм; праве – 15 мм.

Абзацний відступ повинен бути однаковим упродовж усього тексту звіту й дорівнювати 1,25 см.

Звіт повинен містити: тему й мету роботи; суть звіту; висновки.

Суть звіту – це викладання відомостей про предмет лабораторної роботи, які є необхідними й достатніми для розкриття сутності даної роботи (опис методів роботи) та її результатів.

Висновки розміщують безпосередньо після викладання суті звіту, починаючи з нової сторінки. У висновках наводять оцінки одержаних результатів роботи. Текст висновків може поділятись на пункти.

Лабораторна робота 1

ОСОБЛИВОСТІ РОБОТИ В ІНТЕГРОВАНОМУ

СЕРЕДОВИЩІ РОЗРОБКИ C++ BUILDER 6

Мета

– ознайомити з правилами побудови програм в C++ Builder 6;

– ознайомити з елементарними функціями та операторами в C++ Builder 6;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Вивчити загальну структуру ІСР в C++ Builder 6.
2. Ознайомитися з процесом створення програм в C++ Builder 6.
3. Вивчити бібліотеки та їх компоненти, а також властивості різноманітних панелей C++ Builder 6.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

При побудові застосування використовують компоненти вікна редагування Edit, а результат виводиться в панель Panel.

Для відкриття нового застосування виконати команду File | New, у каскадному меню, що відкрилося, вибрати розділ Application.

Перенести на нього зі сторінки бібліотеки Additional два вікна редагування з приєднаними до них мітками LabeledEdit, а зі сторінки бібліотеки Standart – панель Panel, кнопку Button і мітку Label для напису, розмістивши усі об’єкти один під одним.

Змінити написи в мітках компонентів LabeledEdit на написі «Число 1», «Число 2», «Результат». Для цього, вибрати символ «+» у властивості EditLabel цих компонентів і змінити напис у властивості Caption списків властивостей міток, що розкрилися. Задати для міток жирний шрифт, для чого виділити мітку, у вікні Інспектора оператора Об’єктів розкрити подвійним натисканням властивість Font (шрифт), потім також подвійним натисканням розкрити під властивість Style (стиль) і встановити в true властивість fsBold (жирний).

Виділити на формі компонент – кнопку Button1, перейти в інспектор Об’єктів і змінити її властивість Caption (напис) на «Розрахунок».

Змінити зовнішній вигляд панелей: BevelInner і BevelOuter, які визначають вид (втоплений – bvLowered або опуклий – bvRaised) основного поля і рамки панелі. Наприклад, для

встановлення BevelInner = bvLowered BevelOuter = bvRaised. Таким чином, необхідна форма створена.

Приклад обробника натискання кнопки, оператор якого має вигляд:

Panel1 -> Caption = LabeledEdit1 -> Text + " * " +

LabeledEdit2 -> Text + " = " +

FloatToStr (StrToFloat(LabeledEdit1 -> Text) *

StrToFloat (LabeledEdit2 -> Text));

Властивості Caption компонента Panel1 значення виразу, вказаного в правій частині оператора. Цей вираз повинен має тип текс-твого рядка. Починається рядок із тексту, уведеного користувачем у вікно редагування LabeledEdit1, – текст зберігатиметься у властивості Text. До даного тексту додано символи «*». Знак «+» у виразах для рядків означає конкатенацію – зчеплення двох рядків символів. Аналогічним чином до рядка додається текст другого вікна редагування і символи «=». Після цього у рядку розміщається результат множення двох цілих чисел, який має числовий тип. Щоб вставити його в текст, слід спочатку перетворити число в рядок Це перетворення виконує функція FloatToStr(.) Ще одна операція – добуток текстових рядків, які слід попередньо перетворити у числа. Для перетворення заданих текстових рядків у числові значення використовується функція StrToFloat(), що переводить символьне зображення числа в його значення типу дійсного числа. Знак «*», вказаний між двома функціями StrToFloat(), означає операцію множення.

Зберегти результат можна, використовуючи команди File | Save All, або комбінації відповідних швидких кнопок.

Відкомпілювати застосування і виконати його. Переконатися, що воно правильне і швидко працює.

З наданих викладачем індивідуальних завдань вибрати свій варіант, що відповідає номеру за списком в журналі.

Контрольні питання

1. Як створити нову програму в C++ Builder?
2. Які компоненти входять до бібліотеки Standart?
3. Назвіть функції перетворення типів.
4. Укажіть мету застосування компоненту Label.
5. Укажіть основні властивості компоненту Edit.
6. За що відповідає властивість Caption?

Лабораторна робота 2

ОСНОВИ РОБОТИ З КОМПОНЕНТАМИ В C++ BUILDER 6

Мета

– надати практичні навички у роботі з основними елементами C++ Builder 6;

– ознайомити з особливостями функціонування елементів C++ Builder 6;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Ознайомитися з панеллю компонентів Standart.
2. Ознайомитися з процесом введення/виведення даних в C++ Builder 6.
3. Реалізувати представлені алгоритми.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Процес створення програми в C++Builder 6 складається з двох кроків: перший – створення форми програми (діалогове вікно), другий – використання функції обробки подій. Форма використання створюється шляхом додавання в неї компонентів і подальшого їх налаштування.

У формі практично будь-якого застосування є компоненти для забезпечення інтерфейсу між програмою і користувачем, які називають базовими. До базових компонент відносять:

ü Label – поле виведення тексту;

ü Edit – поле редагування тексту;

ü Button – командна кнопка;

ü CheckBox – незалежна кнопка вибору;

ü RadioButton – залежна кнопка вибору;

ü ListBox – список вибору;

ü ComboBox – комбінований список вибору.

Компоненти містяться на панелі інструментів вкладки Standart. Вид компонента, його розмір і поведінка визначають значення властивостей (характеристик) компонента. Інспектор властивостей знаходиться в лівому нижньому кутку екрану. Основну роботу в програмі виконують функції обробки подій, вкладка Events.

Початкову інформацію програма може отримати з полів редагування (компонент Edit), списку вибору (компонент ListBox) або комбінованого списку (компонент ComboBox). Для введення значень логічного типу можна використати Компоненти CheckBox і RadoiButton.

Програма може вивести результат у поле виведення тексту (компонент Label) або у вікно повідомлення (функції ShowMessage, MessageDlg).

Для перетворення, наприклад, редагування тексту, що знаходиться в полі, в ціле число слід використати функцію StrToInt, а в дробове – функцію StrToFloat. Для перетворення цілого, наприклад, значення змінної, в рядок використовується функція IntToStr, а для перетворення дробового – функція FloatToStr або FloatToStrF.

Контрольні питання

1. Які компоненти входять до панелі Standart?
2. Яким чином реалізується введення даних в C++ Builder 6?
3. Яким чином реалізується виведення даних в C++ Builder 6?
4. Укажіть мету застосування компоненту CheckBox.
5. Укажіть мету застосування компоненту ListBox.

Лабораторна робота 3

ОСОБЛИВОСТІ РОБОТИ З НАКОПИЧУВАЧАМИ ІНФОРМАЦІЇ В ІНТЕГРОВАНОМУ СЕРЕДОВИЩІ РОЗРОБКИ C++ BUILDER 6

Мета

– ознайомити з можливостями роботи з накопичувачами інформації в середовищі Borland C++ Builder 6;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Навчитися підключати накопичувачі інформаці в C++ Builder 6.
2. Навчитися обробляти та отримувати доступ до інформації на накопичувачах.
3. Реалізувати представлені алгоритми.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Основні функції для роботи з накопичувачами інформації:

Get Logical Drives: DWORD – функція повертає бітову маску, в якій зберігається інформація про доступні накопичувачі – наявність або відсутність накопичувача.

Get Logical DriveStrings(nBufferLength: DWORD; lpBuffer: PAnsiChar) : DWORD – функція записує спеціальну змінну lpBuffer імена присутніх в системі накопичувачів. Якщо функція виконалася без помилок, то вона повертає довжину в символах буфера, не включаючи завершуючого нульового символу. У разі помилки – повертає нуль.

Get Drive Type (lpRootPathName: PChar): UINT – функція повертає тип носія – змінний, фіксований, CD-ROM, диск RAM, або мережевий диск.

Get Disk Free SpaceEx (lp Directory Name: PChar; &lp Free Bytes Available To Caller, &lp Total Number Of Bytes, &lp Total Number Of Free Bytes: ULARGE_INTEGER): BOOL – функція отримує інформацію про доступний на накопичувачі дисковий простір: загальний об’єм простору, загальний об’єм вільного простору і загальний об’єм вільного простору, доступного для користувача. Якщо функція виконалася без помилки, то вона поверне ненульове значення.

Get Volume Information (lp Root Path Name: PChar; lp Volume Name Buffer: PChar; nVolume Name Size: DWORD; lpVolume Serial Number: PDWORD; &lp Maximum Component Length, &lpFile System Flags: DWORD; &lpFile System Name Buffer: PChar; &nFile System Name Size: DWORD): BOOL – повертає інформацію про файлову систему і про накопичувач інформації. Якщо функція виконалася без помилки, то вона поверне ненульове значення.

Контрольні питання

1. За допомогою яких функцій можливо підключити накопичувачі в C++ Builder 6?
2. Які типи накопичувачів є в C++ Builder 6?
3. Укажіть особливості роботи з різними типами накопичувачів в C++ Builder 6.
4. Яким чином реалізується передача інформації від накопичувача?
5. Яким чином реалізується передача інформації до накопичувача?
6. Як видалити накопичувач?

Лабораторна робота 4

МАТРИЦІ ДАНИХ В C++ BUILDER 6

Мета

– надати практичні навички створювати та оброблювати масиви засобами мови C++ Builder 6;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Ознайомитися з процесом створення масивів в C++ Builder 6.
2. Вивчити правила формування циклів в C++ Builder 6.
3. Реалізувати представлені алгоритми.

4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Ознайомимось на прикладі з операторами, які обчислюють максимальне значення і суму елементів, розташованих в масиві цілих чисел Data розмірністю 10:

```
int Max, Sum;

Max = Sum = Data [0];

for (int i = 1; i < 10; i ++)

{

if (Data [i] > Max) Max = Data [i];

Sum + = Data [i];

}
```

Перший вираз у структурі for уводить цілую змінну i, що є лічильником циклів та ініціалізує її значенням 1; другий вираз перевіряє умову завершення циклу, який у даному випадку повинен завершитися, коли змінна i, використовувана в тілі циклу як індекс масиву, набуде значення, більшого 9; третій вираз структури for після кожного виконання циклу збільшує значення i на 1 за допомогою операції інкремента.

В даному випадку змінна і оголошена в заголовку структури for. Зону її дії визначає тільки ця структура. Після завершення циклів змінна і видаляється з пам’яті.

Контрольні питання

1. Укажіть способи опису масивів даних.
2. Які оператори циклів використовує мова С++?
3. Укажіть особливості циклу for.
4. Що визначає процес зміни лічильника циклу?
5. Який тип може мати лічильник циклу?

Лабораторна робота 5

ДИНАМІЧНІ МАТРИЦІ ДАНИХ В С++ BUILDER 6

Мета

– надати практичні навички створювати і опрацьовувати двовимірні масиви засобами С++;

– підготовити студентів до самостійного використання засобів програмування мовою С++ Builder 6.

Завдання

1. Ознайомитися з процесом створення двовимірних масивів в C++ Builder.
2. Вивчити прийоми роботи з оператором new в C++ Builder 6.
3. Реалізувати представлені алгоритми.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Розглянути приклад операторів, які створюють масив цілих чисел Data:

```
int *Data;
```

```
Data = new int[(StringGrid1 ->RowCount * StringGrid1 ->ColCount)];
```

Аналіз приклада: StringGrid1 ->RowCount і StringGrid1 ->ColCount – динамічні змінні, які є властивостями компонента StringGrid, що розташований на вкладці Additional панелі компонентів C++ Builder. Для уведення двовимірного масиву використано StringGrid.

Для отримання двовимірного масиву використовується функція rand(), яка випадковим чином генерує число в межах від 0 до 32767.

```
for (int i=0; i<StringGrid1 ->RowCount ; i++)
```

```
for (int j=0; j<StringGrid1 ->ColCount; j++)
```

```
StringGrid1 ->Cells[i][j] =  IntToStr(rand() % 10);
```

Для отримання числа в межах від 0 до 9 береться залишок від ділення згенерованого оператором rand() числа на 10.

Для заповнення масиву використано властивість компонента StringGrid, StringGrid1 ->Cells[i][j], чим забезпечено звертання до комірок таблиці; кількість рядків і стовпців в компоненті StringGrid визначено послідовністю: StringGrid1 ->RowCount і StringGrid1 ->ColCount.

Для сортування двовимірного масиву можна використати декілька методів, проте для спрощення операції зручніше попередньо провести перетворення двовимірного масиву в одновимірний наступним чином:

```
int Dim=0;

for (int i=0; i<StringGrid1 ->RowCount; i++)

for (int j=0; j<StringGrid1 ->ColCount; j++)

{

Data[Dim] = StrToInt(StringGrid1 ->Cells[i][j]);

Dim++;

}
```

Для сортування масивів існують наступні способи:

Сортування відбором (selection sort) виконується наступними кроками: перебором знаходиться найменший елемент; він міняється місцями з елементом, що стоїть на нульовому місці; визначається найменший серед тих, що залишилися; міняється місцями з елементом, що стоїть на першому місці і так далі. Цикл закінчується, коли будуть вибрані усі елементи.

Сортування вставками (insertion sort) – сортований масив переглядається в порядку зростання номерів і кожен елемент вставляється у вже переглянуту частину масиву так, щоб зберегти порядок. На кожному кроці сортування частина масиву вже впорядкована, тому для пошуку місця вставки можна використати метод половинного ділення.

Бульбашкове сортування (bubble sort) – найкоротший і один з найповільніших алгоритмів сортування.

Сортування злиттям (merge sort) виконується наступними кроками: початковий масив ділиться навпіл; кожна половина упорядковується; потім проводиться злиття. Для сортування половин використовується та ж функція, що і для сортування усього масиву, функція злиття виділена для економії місця в стеку.

Приклад програми бульбашкового сортування одновимірного масиву Data[Dim]:

```
int  temp=0;

for (int i=0; i<Dim; i++)

for (int i=0; i<Dim; i++)
```

```
if (Data[i]<=Data[i+1])

{       temp = Data[i+1];

Data[i+1] = Data[i];

Data[i] = temp;      }
```

Упорядкованому масиву слід повернути вихідний вид, тобто перетворити його у двовимірний, послідовністю кроків:

```
for (int i=0; i<StringGrid1 ->RowCount; i++)

for (int j=0; j<StringGrid1 ->ColCount; j++)

{      StringGrid1 ->Cells[j][i] = IntToStr(Data[Dim]);

Dim++;    }
```

Слід пам’ятати, що під розташування масиву було виділено пам’ять, яку необхідно звільнити командою delete(Data);

Контрольні питання

1. Дайте визначення динамічного масиву.

2. Якими операторами задається розмір пам’яті для динамічної матриці?
3. Які види сортувань матриць існує?
4. Поясніть принцип роботи методу «Сортування злиттям».

Лабораторна робота 6

ПОБУДОВА КЛАСІВ ТА СТРУКТУР В C++ BUILDER 6

Мета

– ознайомити з поняттями «класи» та «структури даних»;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Ознайомитися з процесом створення класів і структур в C++ Builder 6.
2. Вивчити прийоми роботи з оператором this в C++ Builder 6.
3. Реалізувати представлені алгоритми.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Для створення класу в C++ Builder 6 необхідно створити файл *.h, де міститиметься

оголошення класу.

Лістинг файла test.h

```
#include <vcl.h>

namespace DataBase; оголошення простору імен

{

class DB; оголошення класу

{

Private; зона видимості

struct Data; оголошення структури

{

AnsiString Name;

int Age;
```

float Hight;

int Indef; };

Data *Card;

public; зона видимості

DB(int Count); конструктор класу

virtual ~DB(); деструкція класу

void setData(AnsiString Name, int Age, float Hight, int Indef, int ID);

AnsiString getDataName(int ID){return this ->Card[ID].Name;};

int getDataAge(int ID){return this ->Card[ID].Age;};

float getDataHight(int ID){return this ->Card[ID].Hight;};

int getDataIndef(int ID){return this ->Card[ID].Indef;};

```
};


};
```

На початку оголошено власний простір імен, в якому створюватиметься клас, для створення якого використано службове слово class; у середині класу в зоні видимості private (члени якої не видимі за межами класу і можуть бути змінені лише за допомогою покажчика this) задано структуру за допомогою ключового слова struct; далі створено конструктор і деструкція класу, причому ім’я конструктора класу має співпадати з ім’ям класу; для звернення до захищеної частини класу використано оператор this.

Для опису конструктора, деструкції і неописаних функції необхідно створити файл *.cpp з таким же ім’ям, як і файл, що містить опис класу, тобто test.cpp.

Лістинг файла test.cpp

```
#include "Test.h"

namespace DataBase

{       DB::DB(int Count)

{                   this ->Card = new Data[Count];              }

DB::~DB()
```

```
{                       delete(this ->Card);                }

void DB::setData(AnsiString Name, int Age, float Hight, int Indef, int ID)

{                       this ->Card[ID].Name = Name;

this ->Card[ID].Age = Age;

this ->Card[ID].Hight = Hight;

this ->Card[ID].Indef = Indef;                }  }
```

Для звернення до функцій класу використано оператор :: після імені класу, перед ім'ям класу вказано тип, який повертає функція. В обов'язковому порядку мають бути описані конструктор і деструкція класу.

Очевидно, клас досить простий і усі його методи спрямовані на заповнення і виведення структури, оголошеної в класі. Оскільки створювався масив структур розмірністю Count, то для звернення до елементів структури використано змінну ID.

Лістинг основної програми:

```
#include <vcl.h>

#pragma hdrstop
```

```
#include "Unit 1.h"

#include "Test.h"

#pragma package(smart_init)

#pragma resource "*.dfm"

TForm1 *Form1;

using namespace DataBase;

DB *test;

__fastcall TForm1::TForm1(TComponent* Owner)

: TForm(Owner)

{   }

void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{  test = new DB(StrToInt(Edit12 ->Text));  }

void __fastcall TForm1::Button2Click(TObject *Sender)

{ test ->setData(Edit1 ->Text, StrToInt(Edit2 ->Text), StrToFloat(Edit3 ->Text), StrToInt(Edit4
->Text), StrToInt(Edit5 ->Text));  }

void __fastcall TForm1::Button3Click(TObject *Sender)

{Panel1 ->Caption = test ->getDataName(StrToInt(Edit6 ->Text));

Panel2 ->Caption = IntToStr(test ->getDataAge(StrToInt(Edit6 ->Text)));

Panel3 ->Caption = FloatToStr(test ->getDataHight(StrToInt(Edit6 ->Text)));

Panel4 ->Caption = IntToStr(test ->getDataIndef(StrToInt(Edit6 ->Text)));  }
```

Основна програма містить файл test.h, після оголошення простору імен створюється об’єкт класу DB *test, натисканням кнопки «створити» запускається конструктор класу і відбувається створення масиву структур розмірністю, яку задано в компоненті Edit. Кнопка «додати» виконує функцію setData, в яку передаються параметри з компонентів Edit. Кнопка «показати» виконує функції getData, які виводять значення структури в компоненти Panel.

Контрольні питання

1. Дайте визначення класу в мові С++.
2. Що таке компонентна функція і де вона використовується?
3. Дайте визначення конструктора класу.
4. Для чого використовується деконструктор класу?
5. Для чого використовується оператор this в С++ Builder 6.
6. Поясніть значення терміну «зони видимості» даних класу.

Лабораторна робота 7

НАСЛІДУВАННЯ КЛАССІВ В С++ BUILDER 6

Мета

– ознайомити з поняттями «наслідування» та «ієрархія класів»;

– надати практичні навички описувати похідні класи;

– підготовити студентів до самостійного використання засобів програмування мовою С++ Builder 6.

Завдання

1. Ознайомитися з процесом спадкоємства класів в С++ Builder 6.
2. Вивчити прийоми роботи з поняттям «Спадкоємство класів» в С++ Builder 6.

3. Реалізувати представлені алгоритми.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Класи можуть знаходитися відносно спадкоємства, при якому формується ієрархія об’єктів, що відповідає заздалегідь передбаченій ієрархії класів. Ієрархія класів дозволяє визначати нові класи на основі вже наявних. Наявні класи зазвичай називають базовими, а нові класи, що формуються на основі базових класів – похідними (породженими), або спадкоємцями. Похідні класи «отримують спадок» – дані і методи своїх базових класів, а також можуть поповнюватися власними компонентами. Успадковані компоненти не переміщаються в похідний клас, а залишаються у базових класах.

У визначенні і описі похідного класу наводиться список базових класів, з яких він безпосередньо наслідує дані і методи. Між ім’ям класу, що вводиться, і списком базових класів розміщається двокрапка.

Наприклад:

class S: X, Y, Z { . . .};

При такому визначенні клас S породжений класами X, Y, Z, звідки він наслідує компоненти. Спадкоємство компонента не виконується, якщо його ім’я буде використано в якості імені компонента у визначенні похідного класу S.

Як приклад створюється базовий клас «liniya1», з якого наслідує похідний клас «liniya 2», в якому лінія переміщається в нове місце на екрані, збільшується розмір, але її нахил залишається таким, як у базовому класі.

class liniya1

```
{

public:

int x1;

int y1;

int x2;

int y2;

int Ccolor;

narisovat_liniya_1(int a,int b,int c,int d, int color);

};

class liniya2: public liniya1

{
```

```
public:

narisovat_liniya_2(void);

};

liniya1::narisovat_liniya_1(int a,int b,int c,int d, int color)

{

x1=a;

y1=b;

x2=c;

y2=d;

Ccolor=color;

setcolor(Ccolor);

line(x1,y1,x2,y2);
```

}

При виконанні роботи слід використати графічні примітиви, за допомогою яких будуються необхідні фігури. У алгоритмі має бути чітко визначено використання функції спадкоємства класу графічного примітиву.

Контрольні питання

1. Для чого використовується механізм спадкоємства в класах?
2. Наведіть визначення множинного спадкоємства.
3. Наведіть визначення термінів «предок» і «спадкоємець» в спадкоємстві.
4. Наведіть визначення поняття віртуального базового класу.
5. Наведіть визначення абстрактного класу і віртуальної функції.

Лабораторна робота 8

ПОБУДОВА ВИКЛЮЧЕНЬ В C++ BUILDER 6

Мета

– надати практичні навички створювати виключення та застосовувати оператори try, throw, catch мови C++;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Ознайомитися з процесом створення виключень в C++ Builder 6.
2. Вивчити прийоми роботи з операторами try, throw, catch в C++ Builder 6.
3. Реалізувати представлені алгоритми.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Приклад програми конструкції виключень.

try; вказує на те, що далі піде блок виключень.

{ throw 1; ключове слово, що власне і створює виключення}

catch (int a); заголовок блоку виявлення виключень.

{ MessageDlg (“Exception – 1”, mtError, TMsgDlgButtons() << mbOK,); повідомлення про визначення виключення};

Аналіз наведеної програми:

1. try – ключове слово, що вказує на початок блоку виключень, використовується тільки в C++; мовою C використовується __try, яке можна так само використати в C++. Блок виключень береться у фігурні дужки, як це наведено у прикладі, і, за необхідності, в ньому створюється виключення.

2. throw – ключове слово, що створює виключення, тобто ініціалізується тимчасовий об’єкт певного типу, за допомогою якого необхідно створити виключення. У прикладі використано тип int, відповідно створюється тимчасовий об’єкт типу int, що містить дані.

3. catch – ключове слово, що вказує на початок блоку обробки виключень, де може розмістити будь-які реакції на визначені виключення; у параметрах блоку слід указати тип даних, використаний при створенні виключень; в наведеному вище прикладі створено виключення типу int – цей тип указано в дужках.

4. «Project XXX.exe raised exception» – повідомлення про створення виключення програмою, яка обробляється дебаггером. Дане повідомлення виникає також у випадках: Operating System Exception (Виключення операційної системи); Language Exception (Виключення мови).

5. Дебаггер знаходиться за адресою: Tools ->Debugger Options, де на вкладці Language Exceptions можна виявити звіти про знайдені виключення.

6. Для створення власного типу даних TCustomException, що відповідатиме за виключення, слід натиснути Add і вписати в рядок діалогу; OK.

Приклад програми зміни коду.

class TCustomException{}; тип даних користувача

TCustomException NewEx; об’єкт класу типу TCustomException

try; початок блоку створення виключень

{ throw NewEx; створення виключення }

catch (TCustomException); початок блоку виявлення виключень

{MessageDlg (“Exception!”!!,mtError, TMsgDlgButtons() << mbOK, 0); повідомлення};

Оскільки класи є типами даних, можна для зручності створити власний TcustomException – об’єкт NewEx.

Приклад програми.

;клас для виключень з параметрами.

;не забувати реєструвати його в опціях дебаггера.

class Tex

{public:

int f Code;

Tex(int eCode) {fCode=eCode;};}; код, вбудований в клавішу Button1

try

{ throw TEx(1301);}

catch(TEx Ex) }

Контрольні питання

1. Як визначити особливу ситуацію?
2. Як можна виділити в тексті програми контрольований блок?
3. Для чого використовуються оператори thorow і catch?
4. Коли використовується скорочена форма оператору thorow?
5. Наведіть опис конструктора і деконструктора у виключенні.

Лабораторна робота 9

ОСОБЛИВОСТІ РОБОТИ З БАЗАМИ ДАНИХ C++ BUILDER 6

Мета

– ознайомити з можливостями C++ Builder 6 щодо створення та роботи з базами даних;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Ознайомитися з можливостями C++ Builder 6 для роботи з базами даних.
2. Ознайомитися з процесом створення баз даних в C++ Builder 6.
3. Реалізувати представлені алгоритми.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Використовуючи Borland C++ Builder 6, можна створити застосування, працюючи як з розрахованими на одного користувача базами даних (БД), так і з серверними системами управління базами даних (СУБД), такими як Oracle, Sybase, Informix, Interbase, MS SQL Server, DB2, а також з ODBC-джерелами. C++ Builder має достатні можливості для створення застосувань, що використовують БД.

Набір даних в C++ Builder 6 – це об’єкт, що складається з набору записів, кожний з яких, у свою чергу, складається з полів та покажчика поточного запису. Набір даних може мати повну відповідність з реально існуючою таблицею або бути результатом запиту, він може бути частиною таблиці або об’єднувати між собою декілька таблиць.

Набір даних в C++ Builder 6 є спадкоємцем абстрактного класу TdataSet. Абстрактним є клас, від якого можна породжувати інші класи, але не можна створити екземпляр об’єкту цього класу. Наприклад, класи TQuery, TTable і TStoredProc, що містяться на сторінці палітри компонентів Data Access, – спадкоємці TDBDataSet, який, у свою чергу, є спадкоємцем TDataSet. TDataSet містить абстракції, необхідні для безпосереднього управління таблицями або запитами, забезпечуючи засоби для того, щоб відкрити таблицю або виконати запит і переміщатися між рядками.

Borland C++ Builder 6 має широкі можливості доступу до БД. Оскільки бази даних призначені не лише для зберігання, але і для вибору і обробки інформації, одним з найважливіших аспектів їх використання є створення запитів до них. Запит в C++ Builder 6 – це об’єкт, що є набором даних. Зазвичай для створення запиту використовується компонент TQuery – спадкоємець абстрактного класу TDataSet.

Контрольні питання

1. Назвіть компоненти, які відповідають за роботу з базами даних в C++ Builder.
2. Назвіть основні методи роботи з базами даних в C++ Builder 6.
3. Назвіть типи баз даних, з якими можна працювати в C++ Builder 6.
4. Назвіть вкладки, на яких розташовані компоненти для роботи з базами даних в C++ Builder 6.
5. Визначте зв’язки між компонентами для роботи з базами даних.
6. Які бази даних існують?

Лабораторна робота 10

ОСНОВИ РОБОТИ З БАЗАМИ ДАНИХ В C++ BUILDER 6

Мета

– ознайомити з процесом створення баз даних в C++ Builder 6;

– ознайомити з переліком та порядком роботи з компонентами DataControl в C++ Builder 6;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Дослідити можливості Database Desktop.
2. Створити таблицю для збереження даних.
3. Створити базу даних (БД) на основі таблиці в C++ Builder 6.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

1. Створити таблицю для збереження даних, відкривши Database Desktop. Послідовністю команд файл-створити-таблиця, створити таблицю за допомогою конструктора Paradox, тип таблиці – Paradox 7. Кожне поле таблиці має ім’я і тип даних, а також довжину (при необхідності):

– ім’я (тип: Alpha, розмір: 255);

– вік (тип: Long Integer);

– дата (тип: Date).

Для збереження файла в директорії програми слід натиснути кнопку «зберегти».

2. У меню Database програми C++ Builder обрати Explore. Використовуючи команду «створити», створити нову базу даних, тип якої – стандартний. Для задання шляху до даних натиснути на трикрапку і вибрати папку, де збережена вихідна таблиця.

3. Натисканням кнопки «активувати» розкрити БД і перетягти вихідну таблицю на форму застосування. C++ автоматично створить три нові компоненти: DataSource, Table, DBGrid. Перший відповідає за зв’язок з БД і передачу інформації з таблиць у компонент Table, який у свою чергу редагує і безпосередньо оброблює дані у таблицях. Компонент DBGrid дозволяє відображати і редагувати дані з таблиць.

Для створення кнопок «старт» і «стоп», які вмикають і вимикають БД, слід перетягти на форму компонент DBNavigator (Вкладка DataControl), який полегшує роботу з БД. У властивостях DataSource компонента DBNavigator слід обрати DataSource1. Для додавання запису слід натискати «+», а для видалення – «-».

Контрольні питання

1. Дайте визначення терміну «база даних».
2. Назвіть три основні властивості (ознаки) БД.
3. Дайте визначення терміну «СУБД».
4. Назвіть призначення компонент DataSource, Table, DBGrid.
5. Укажіть мету використання компоненту DBNavigator.

Лабораторна робота 11

ПОБУДОВА ЗВ'ЯЗКІВ МІЖ ТАБЛИЦЯМИ БАЗ ДАНИХ

Мета

– засвоїти правила зв'язування таблиць між собою та програмування при роботі з базами даних в C++ Builder 6;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. У таблиці, створеній в лабораторній роботі 10 (табл. 11.1) додати один стовпець,

якій вибирається відповідно до варіанту завдання.

2. Створити іншу таблицю (табл. 11.2), структура якої вибирається відповідно до варіанту завдання.

3. Створити в C++ Builder 6 новий додаток.

4. Розмістити на формі додатка компоненти для відображення двох наявних таблиць (тільки для читання) і зв'язати ці таблиці між собою. Показати дію зв'язку.

5. Переглянувши усі записи табл. 11.1, обчислити функцію, яка вибирається відповідно до варіанту завдання.

6. Організувати додавання (при натисненні на кнопку «Зберегти») нових записів у табл. 11.1. Ці значення вводяться у поля компонент, не пов'язаних з базою даних.

7. Створити (програмно) нову таблицю (табл. 11.3), в яку переписати з таблиці 11.1 усі записи, що задовольняють умові, вибираній відповідно до варіанту завдання. Відобразити на формі вміст табл. 11.3.

8. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

1. Програмування роботи з базами даних. Стан набору даних.

Набір даних може знаходитися в одному з наступних основних станів:

dsInactive – набір даних закритий, дані недоступні;

dsBrowse – дані можуть спостерігатися, але не можуть змінюватися (використовується за умовчання);

dsEdit – поточний запис може редагуватися;

dsInsert – може вставлятися новий запис;

dsSetKey – доступний режим пошуку запису і операція задання діапазону змін SetRange.

2. Пересилання запису в базу даних.

Під час редагування поточного запису зміни здійснюються у буфері, а не в самій БД. Пересилання запису у БД проводиться тільки при використанні методу Post.

3. Доступ до полів.

Поля відображаються об’єктами класу TField і похідних від нього класів TStringField, TSmallintField, TBooleanField, тощо. Ці об’єкти можуть створюватися трьома способами:

- автоматично генеруватися для кожного компонента набору даних (Table та ін.).
- створюватися в процесі проектування за допомогою Редактора Полів.
- створюватися програмно в процесі виконання додатку.

Доступ до об’єктів полів можливий трьома способами:

- за порядковим індексом об’єкта;
- за іменем поля;

- за іменем об’єкту.

4. Методи навігації.

Набори даних мають ряд методів, що дозволяють здійснювати навігацію – переміщення таблицею:

Fibrst – переміщення до першого запису;

Last – переміщення до останнього запису;

Next – переміщення до наступного запису;

Prior – переміщення до попереднього запису;

MoveTo(int i) – переміщення до кінця (при i > 0) або до початку (при i < 0) на i записів.

5. Пошук записів.

Одна з найважливіших для користувача операцій з БД – пошук записів за деяким ключем. Існує декілька методик пошуку записів : SetKey, FindKey, Lookup і Locate.

Для застосування методики SetKey таблиця заздалегідь має бути індексована за полем, за яким буде проводитися пошук. Для встановлення таблиці в стан пошуку використовується метод SetKey – dsSetKey. У стані dsSetKey набір даних сприймає подальший оператор надання значення полю не як привласнення, а як задання ключа

пошуку.

Для методів FindKey і FindNearest зручно застосовувати макрос OPENARRAY, що створює тимчасовий відкритий масив і визначальний його розмір:

OPENARRAY(TVarRec(список ключів))

Останній метод пошуку – Lookup. Цей метод визначається таким чином:

System:: Variant fastcall Lookup(const System:: AnsiString KeyFields, const System:: Variant SKeyValues, const System:: AnsiString ResultFields).

Перші два параметри аналогічні методу Locate. Третій параметр – рядок, що перераховує поля, значення яких повертаються у вигляді масиву Variant. Якщо не знайдено відповідного запису, функція повертає false.

6. Методи установки діапазону допустимих значень.

Метод SetRangeStart переводить набір даних в стан dsSetKey, і наступний оператор надання значення полю сприймається як задання для поля нижньої межі діапазону можливих значень. Метод SctRangeEnd діє аналогічно, але подальший оператор привласнення сприймається як задання верхньої межі діапазону. Після того, як межі діапазону встановлені, можна виконати метод ApplyRange. При цьому почнуть використовуватися встановлені межі діапазону і доступними для перегляду та редагування будуть тільки ті записи, в яких значення вказаного поля знаходяться усередині діапазону.

Контрольні питання

1. Назвіть способи зв’язку таблиць.
2. Назвіть стани набору даних.
3. Укажіть основні методи для роботи зі станами набору даних.
4. Розкрийте поняття «кешування змін».
5. Вкажіть способи доступу до об’єктів полів.
6. Назвіть методи для здійснення навігації таблицею.
7. Назвіть основні методики пошуку записів за деяким ключем.
8. Назвіть декілька методів, що дозволяють задавати діапазон допустимих значень під час виконання додатка.

Лабораторна робота 12

КОМПОНЕНТ QUERY І ЙОГО МЕТОДИ. SQL-ЗАПИТИ. ОПЕРАТОР SELECT

Мета

– засвоїти правила роботи з компонентом Query і створення SQL-запитів, що використовують оператор Select;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Створити в C++ Builder 6 новий додаток.
2. Розмістити на формі додатка компоненти для створення і відображення SQL-запитів.
3. Сформувати (та представити результати їх виконання) SQL-запити для виконання функцій, обраних відповідно до варіанту завдання. В якості початкової таблиці використати таблицю, зроблену в лабораторній роботі 10, з урахуванням її модифікації у лабораторній роботі 11.
4. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Для ознайомлення з Query необхідно відкрити в C++Builder 6 новий додаток і розмістити на форму компоненти Query, DataSource, DBGrid. У властивості DataSet компонента DataSourcel задати Queryl, а у властивості DataSource компонента DBGridl – DataSourcel. Таким чином, створено послідовність: набір даних (Queryl), джерело даних (DataSourcel), компонент візуалізації і управління даними (DBGridl).

Основна властивість компонента Query – SQL, що має тип TStrings. Це список рядків, що містять запити SQL. Налаштування компонента Query в процесі проектування можна проводити вручну, як це робиться з усіма компонентами, або за допомогою спеціального Візуального Будівника Запитів. Його виклик проводиться натисканням правої кнопки миші на компоненті Query і вибором із спливаючого меню розділу SQL Builder.

Перш ніж приступати до ручного налаштування, у властивості DataBaseName компонента Query потрібно задати БД, з якою здійснюватиметься зв’язок. БД задається вибором зі спливаючого списку псевдонімів або вказівкою повного шляху до каталогу або файлу (залежно від СУБД, що використовується).

Передусім, потрібно занести у властивість SQL запит, що містить ім’я таблиці, з якою слід працювати.

Перед детальним налаштуванням компонента Query слід сформувати в його властивості SQL запит, в якому вказується таблиця і перераховуються параметри, якщо вони використовуються в додатку. Запит, що заноситься в SQL на початку проектування, носить службовий характер. Надалі можна його програмно замінити на будь-який інший запит.

Після визначення таблиці, яка опрацьовуватиметься, можна буде настроїти поля в Query.

Після створення відповідного запиту можна встановити властивість Active компонента Query в true. Якщо усе виконано правильно, то в компоненті DBGridl відобразиться інформація із таблиць, що були вказані у запиті.

Додаток можна запустити на виконання, переглянути його функціонування, але редагування додатку неможливе. Це пов’язано з тим, що запит Select повертає таблицю тільки для читання. Втім, в простому додатку це легко виправити. Досить встановити в компоненті Queryl властивість RequestLive в true, що дозволяє повертати як результат запиту – змінюваний, «живий» набір даних, замість таблиці тільки для читання. Точніше, установка RequestLive в true робить спробу повернути «живий» набір даних. Успішною ця спроба буде тільки при дотриманні ряду умов, зокрема:

- набір даних формується зверненням тільки до однієї таблиці;
- набір даних не впорядкований (у запиті не використовується ORDER BY);
- у наборі даних не використовуються сукупні характеристики типу Sum, Count тощо;
- набір даних не кешується (властивість CashedUpdates рівна false).

Для управління відображенням даних в компоненті Query є вже відомий Редактор Полів (Field Editor). Викликати його можна або подвійним натисканням на Query, або натисканням правої кнопки миші на Query і вибором Fields Editor із спливаючого меню. У Редакторові Полів можна: додати імена отримуваних полів (натискання правою кнопкою миші і вибір розділу меню Add); задати заголовки полів, що відрізняються від їх імен; зробити якісь поля невидимими (Visible), не редагованими (ReadOnly); в логічних полях можна задати слова, що висвічуються (так;ні); задати формат висвічування чисел; створити обчислювані поля, поля перегляду; задати діапазони значень тощо.

Контрольні питання

1. Укажіть особливості компонента Query.
2. Дайте порівняльну характеристику компонент Query і Table.
3. Назвіть основну властивість компонента Query.
4. Який тип має властивість SQL?

5. Назвіть основні методи компонента Query.
6. Укажіть особливості оператора Select.
7. Запишіть синтаксичну форму оператора Select.
8. Як повернути усі поля таблиці через запит Select?
9. Розкажіть про сукупні характеристики, що використовуються у запитах.

Лабораторна робота 13

SQL-ЗАПИТИ. ОБ'ЄДНЕННЯ ТАБЛИЦЬ

Мета

– засвоїти правила об'єднання таблиць засобами SQL-запитів та підзапитів;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Створити таблицю, що містить поля, згідно із варіантом завдання.
2. Написати запити, які будуть виводити результати, що вказані у варіантах завдання.
3. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

13.1. Об'єднання таблиць.

Одна з найважливіших особливостей запитів SQL – їх здатність визначати зв'язки між численними таблицями і виводити інформацію з них. Цей вид операції називається об'єднанням, яке є одним з видів операцій в реляційних базах даних. При об'єднанні

таблиці представлені списком після FROM, відділяються комами. Предикат запиту може посилатися до будь-якого стовпця будь-якої пов’язаної таблиці і може використовуватися для зв’язку між ними. Зазвичай предикат порівнює значення в стовпцях різних таблиць, щоб визначити, чи задовольняє WHERE встановленій умові.

13.2. Створення об’єднання.

Приклад програми створення відповідності між продавцями і замовниками з певного міста.

```
SELECT Customers.cname, Salespeople.sname Salespeople.city FROM Salespeople, Customers WHERE Salespeople.city = Customers.city; SELECT Customers.cname, Salespeople.sname, Salespeople.city FROM Salespeople, Customers WHERE Salespeople.city = Customers.city
```

SQL в об’єднанні в основному досліджує кожну комбінацію рядків двох або більше можливих таблиць і перевіряє ці комбінації за їх предикатами. Якщо комбінація проводить значення, яке робить предикат вірним, і якщо поле city з рядків таблиць Замовника дорівнює London, то Peel – це те значення, яке комбінація вибере для виводу. Те ж саме, буде потім виконано для кожного продавця в таблиці Продавців (у деяких з яких не було ніяких замовників в цих містах).

Контрольні питання

1. Дайте визначення повного імені таблиці.
2. Дайте визначення повного імені стовпців.
3. Які види об’єднання таблиць існують?
4. Які запити опрацьовують об’єднанні таблиці?.
5. За допомогою яких SQL-запитів можна об’єднати таблиці?

Лабораторна робота 14

SQL-ЗАПИТИ. ПІДПОРЯДКОВАНІ ЗАПИТИ

Мета

– надати практичні навички створення підпорядкованих запитів;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Створити таблицю, що містить поля, згідно із варіантом завдання.
2. Написати підпорядковані запити, які будуть виводити результати, що вказані у варіантах завдання.
3. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

При використанні підзапитів в предикатах, заснованих на реляційних операціях (рівняннях або нерівностях), слід переконатися, що використано підзапит, який видаватиме тільки один рядок виведення. Підзапити, які не проводять ніякого виведення (чи нульовий вивід), змушують розглядати предикат як невідомий. Проте невідомий предикат має той самий ефект, що і невірний: ніякі рядки не вибираються основним запитом.

DISTINCT з підзапитами.

В деяких випадках можна використовувати DISTINCT, щоб змусити підзапит генерувати поодиноке значення.

Приклад програми для знаходження всіх порядків кредитування для тих продавців, які обслуговують Hoffman 'a (cnum = 2001).

```
SELECT *  FROM Orders  WHERE snum = (SELECT DISTINCT snum        FROM Orders
          WHERE cnum = 2001);          SELECT *                        FROM  Orders
```

```
WHERE snum = (SELECT DISTINCT snum
FROM Orders                    WHERE cnum = 2001);
```

Використання агрегатних функцій в підзапитах

Тип функцій, який автоматично може проводити поодиноке значення для будь-якого числа рядків – агрегатна функція.

Будь-який запит, що використовує поодиноку функцію агрегату без пропозиції GROUP BY, вибиратиме поодиноке значення для використання в основному предикаті.

Використання підзапитів, які видають багато рядків за допомогою оператора IN

Можна використовувати підзапити, які виводять будь-яке число рядків, застосовуючи спеціальний оператор IN (оператори BETWEEN, LIKE і IS NULL не можуть використовуватися з підзапитами). IN визначає набір значень, одне з яких повинне співпадати з іншим терміном рівняння предиката в запиті, щоб предикат був вірним.

Коли використовується IN з підзапитом, SQL просто формує цей набір з виведення підзапиту.

Використання виразів у під запитах

Можливо використовувати вираз, заснований на стовпці, а не просто сам стовпець, в пропозиції SELECT підзапиту. Це може бути виконано або за допомогою реляційних операторів, або з IN.

Приклад програми використання реляційного оператора.

```
SELECT *          FROM Customers          WHERE cnum =          (SELECT
```

snum + 1000 FROM Salespeople WHERE sname = Serres);

Підзапити в пропозиції HAVING

Можна використовувати підзапити усередині пропозиції HAVING, які мають власні агрегатні функції за умови, що вони не використовують декілька значень, або застосовувати GROUP BY чи HAVING.

Приклад програми використання підзапиту.

SELECT rating, COUNT (DISTINCT cnum) FROM Customers GROUP BY rating HAVING rating > (SELECT AVG (rating) FROM Customers WHERE city = "San Jose");

Контрольні питання

1. Опишіть роботу підзапиту.
2. Укажіть, які значення може виводити підзапит.
3. Опишіть роботу DISTINCT у підзапитах.
4. Опишіть використання агрегатних функцій у підзапитах.
5. Опишіть використання оператора IN.
6. Опишіть підзапити в пропозиціях HAVING.

Лабораторна робота 15

ВСТАНОВЛЕННЯ MYSQL SERVER

Мета

– ознайомити з правилами встановлення, налаштування та роботи у середовищі MySQL;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Встановити MySQL Server.
2. Створити базу даних згідно із варіантом завдання.
3. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Встановлення MySQL Server:

1. У вікні Setup Type відмітити вибіркову установку компонентів "Custom".
2. У вікні Custom Setup можна обрати додаткові компоненти і змінити директорію програми.
3. У вікні Ready to Install the program натиснути "Install".
4. Налаштувати MySQL сервер після натискання кнопки Finish у вікні Wizard Completed.
5. У вікні MySQL Server Instrance Configuration обрати деталізоване налаштування – "Detailed Configuration" – Next.
6. Відмітити пункт "Developer Machine" – Next.
7. Пункт "Multifunctional Database" дозволяє працювати як з таблицями типу InnoDB (з можливістю використання транзакцій), так і з високошвидкісною MyISAM (як правило для веб-розробок використовується саме цей тип таблиць).
8. Вибрати диск і директорії для зберігання таблиць типу InnoDB.

10. Вибрати максимально можливу кількість підключень до сервера MySQL. При виборі "Decision Support (DSS)/OLAP", максимальна кількість підключень буде обмежена двадцятьма, чого більш ніж достатньо при установці сервера на домашньому комп’ютері і відсутності великої кількості одночасних підключень.

11. Звернути увагу на налагодження налаштувань цього вікна. Відмітивши "Manual Selected Default Character Set / Collation" і вибравши із меню, яке з'явилось "cp1251", визначити, що спочатку для таблиць використовуватиметься кодування Cyrillic Windows (cp1251) – це означає коректну роботу з російською мовою в цьому кодуванні.

12. Якщо відмітити "Install As Windows Service", сервер запускатиметься у вигляді сервісу, що є рекомендованим способом запуску. Нижче, в списку контекстного меню, задається ім'я сервісу. Знявши прапорець з "Launch the MySQL Server automatically" запускається сервер вручну. Відмічання прапорцем "Include Bin Directory in Windows PATH" дозволить встановити видимість директорії "bin" для командного рядка.

13. Встановити пароль користувача "root": будь-який простий пароль замість порожнього поля унеможливить появу непередбачених ситуацій.

Контрольні питання

1. В чому полягає різниця між поколіннями MySQL Server?
2. Укажіть особливості установлення MySQL Server.
3. Укажіть особливості роботи з MySQL Server.
4. Укажіть особливості створення баз даних в MySQL Server.
5. Визначте користувачів MySQL Server.
6. Укажіть способи захисту баз даних в MySQL Server.

Лабораторна робота 16

РОБОТА З ТАБЛИЦЯМИ І БАЗАМИ ДАНИХ В MYSQL

Мета

– набути навичок роботи з таблицями та базами даних у середовищі MySQL;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Створити базу даних згідно із варіантом завдання.
2. Створити таблиці в базі даних за допомогою команди CREATE TABLE.
3. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

16.1. Створення бази даних за допомогою оператора CREATE DATABASE.

Для роботи з MySQL Server необхідно використовувати MySQL Command Line Client.

Синтаксис оператора CREATE DATABASE.

CREATE DATABASE [IF NOT EXISTS] db_name [CHARACTER SET charset] [COLLATE collation];

db_name – ім’я, яке буде присвоєно створюваній базі даних;

IF NOT EXISTS – якщо не вказати цей параметр, то при спробі створення бази даних із вже існуючим ім’ям, виникне помилка виконання команди.

CHARACTER SET, COLLATE – використовується для задання стандартного кодування таблиці і порядку сортування.

16.2. Створення таблиці у базі даних командою CREATE TABLE.

Синтаксис:

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name [(create_definition,...)]

[table_options] [select_statement]

tbl_name – задає ім’я таблиці, яка буде створена в поточній базі даних. Якщо ніяка база даних на момент виклику команди CREATE TABLE не була визначена поточною, то виникне помилка виконання команди.

TEMPORARY – використовується для створення тимчасової таблиці з ім’ям tbl_name протягом виконання тільки поточного сценарію. Після виконання сценарію створена таблиця видаляється.

IF NOT EXISTS – якщо вказано цей параметр і проводиться спроба створити таблицю з дублюючим ім’ям (тобто таблиця з таким ім’ям в поточній БД вже є), то таблиця не буде створена і повідомлення про помилку не з’явиться. Інакше, таблиця також створена не буде, але команда викличе помилку.

Create_definition – визначає внутрішню структуру створюваної таблиці (назви і типи полів,

ключі, індекси і так далі).

Col_name – задає ім’я стовпця в створюваній таблиці.

Type – задає тип даних для стовпця col_name.

[NOT NULL | NULL] – вказує, чи може даний стовпець містити значення NULL або ні.

DEFAULT default_value – задає значення за умовчання для цього стовпця. При вставці нового запису в таблицю командою INSERT, якщо значення для поля col_name явно вказано не було, встановлюється значення default_value.

AUTO_INCREMENT – при вставці нового запису в таблицю поле з цим атрибутом автоматично отримає числове значення, на 1 більше найбільшого значення для цього поля у нинішній момент часу. Ця можливість зазвичай використовується для генерування унікальних ідентифікаторів рядків. Стовпець, для якого застосовується атрибут AUTO_INCREMENT, повинен мати цілочисельний тип.

Щоб отримати значення ID останньому вставленому запису, можна скористатися командою MySQL SELECTLAST_INSERT_ID( ).

PRIMARY KEY – задає первинний ключ таблиці. У таблиці може бути заданий тільки один первинний ключ. Усі значення стовпця, поміченого як первинний ключ, не повинні містити значення NULL. Якщо при створенні таблиці первинний ключ явно вказаний не був, а застосування його вимагає, то БД MySQL автоматично встановлює перший стовпець з параметром UNIQUE, якщо в усіх значеннях цього стовпця ніде не зустрічається значення NULL.

У якості первинного ключа можна задати як один, так і декілька стовпців:

PRIMARY KEY(col_1, col_2, ...)

Тільки в цьому випадку жоден інший стовпець не може бути первинним, тобто не може бути описаний:

PRIMARY KEY(col_1), PRIMARY KEY(col_1, col_2)

Поля PRIMARY KEY є проіндексованими полями (детальнішу інформацію по індексах читайте далі в INDEX).

KEY – синонімом до INDEX.

INDEX – задає поля, які будуть проіндексовані.

UNIQUE – ключ вказує на те, що даний стовпець може мати тільки унікальні значення. Унікальними можна задати як один, так і декілька стовпців:

CREATE TABLE users (name VARCHAR (200) NOT NULL, address VARCHAR (255) NOT NULL, UNIQUE (name, address));

FULLTEXT – задає поля, до яких надалі може бути застосований повнотекстовий пошук.

Повнотекстовий пошук є засобом MySQL, спрямованим на пошук потрібної інформації у базі даних і виведення результатів відповідно до релевантності знайдених рядків відносно пошукового запиту.

FOREIGN KEY і CHECK – введені для сумісності при перенесенні коду з інших SQL-баз даних при запуску застосувань, що створюють таблиці з посиланнями.

table_options – задає додаткові параметри створюваної таблиці. Ця можливість з'явилася в MySQL починаючи з версії 3.23.

TYPE – задає тип створюваної таблиці.

COMMENT – коментар для цієї таблиці завдовжки 60 символів.

PASSWORD – шифрує файл '.frm' за допомогою пароля. Ця опція не функціонує в стандартній версії MySQL.

ROW_FORMAT – визначає, яким чином повинні зберігатися рядки. Нині ця опція працює тільки з таблицями MyISAM, які підтримують формати рядків DYNAMIC і FIXED.

UNION – опція застосовується, якщо необхідно використовувати сукупність ідентичних таблиць як одну таблицю. Вона працює тільки з таблицями MERGE. На даний момент для таблиць, що зіставляються з таблицею MERGE, необхідно мати привілеї SELECT, UPDATE і DELETE.

Контрольні питання

1. Запишіть синтаксис оператора CREATE DATABASE.
2. Запишіть синтаксис оператора CREATE TABLE.
3. Який оператор необхідно використовувати, для того щоб визначити налаштування вже існуючої бази даних?
4. Що задає параметр TEMPORARY при створенні таблиці?
5. Укажіть декілька можливих значень параметра type.

6. Для чого використовується AUTO_INCREMENT?
7. Вкажіть призначення PRIMARY KEY.
8. Для чого застосовується опція UNION?

Лабораторна робота 17 **НАПИСАННЯ ЗАПИТІВ У MYSQL**

Мета

– надати практичні навички створення запитів за існуючими таблицям баз даних в MySQL;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Вивести на екран інформацію про існуючу базу даних.
2. Навчитись обробляти дані з існуючих баз даних.
3. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Підключитися до MySQL-серверу, запущеному на даному комп'ютері, відкрити сеанс: MySQL – username – password=passwd

Тут usename – ім'я користувача, passwd – пароль.

При введенні команд слід пам'ятати про знак «;» у кінці, інакше MySQL виведе значок ->, вимагаючи продовження команди. Можна поставити розділовий знак «;» і після значка -> і знову натиснути Enter.

Перегляд існуючих БД: show databases – запит виведе список усіх баз, доступних користувачеві.

Вибрати потрібну БД командою use db_test, результатом – Database changed. Тако ж, цією командою можна замінити базу з поточної на будь-яку іншу.

Вибрати базу можна відразу при відкритті сеансу, додавши у кінці ім’я бази. Це виглядатиме так: mysql – u username –password=passwd db_test

Щоб дізнатися про таблиці, наявні у базі, слід запустити show tables;

Для перегляду опису таблиці: які в ній є поля, якого вони типу, які з полів проіндексовані – слід скористатися командою desc person, що виведе потрібну інформацію.

Перелік команд:

1. Визначити кількість записів в таблиці SELECT count(*) FROM table_name.
2. Вибрати всі записи з таблиці SELECT * FROM table_name.
3. Вибрати декілька записів з таблиці, наприклад 5. Це корисно, коли потрібно дізнатися, як приблизно виглядають дані в таблиці. SELECT * FROM table_name LIMIT 5.
4. Вибрати усі записи з person, упорядковані за зростанням номера person_no. SELECT * FROM person ORDER BY person_no.
5. Вибрати усі записи з person, упорядковані за спаданням номера person_no. SELECT * FROM person ORDER BY person_no DESC.
6. Вибрати декілька (12) записів з person, упорядкованих за зростанням номера person_no. SELECT * FROM person ORDER BY person_no LIMIT 12.
7. Вибрати усі записи з person, де поле name рівне Anna SELECT * FROM person WHERE name='Anna'.
8. Вибрати усі записи з person, де поле name починається з An SELECT * FROM person WHERE name LIKE 'An%'.
9. Вибрати усі записи з person, впорядковані по person_no, де поле name закінчується на na SELECT * FROM person WHERE name LIKE '%na' ORDER BY person_no.

10. Вибрати усі пари name, що зустрічаються, last_name з таблиці SELECT name, last_name FROM person.

11. Вибрати усі різні (унікальні) пари name, що зустрічаються, last_name з таблиці. Повтори не будуть представлені у результатах. SELECT DISTINCT name, last_name FROM person.

Контрольні питання

1. Яким чином можна переглянути список існуючих баз даних?
2. За допомогою якої команди виконується перехід до бази даних?
3. Яким чином можливо визначити кількість записів у таблиці?
4. Якою командою можливо вивести список таблиць в БД?
5. Яким чином можна дізнатись про кількість записів у таблиці?
6. Які фільтри існують для вибору записів з таблиці.

Лабораторна робота 18

ЗВ’ЯЗОК MYSQL SERVER І C++ BUILDER. РОБОТА З ТАБЛЦЯМИ І БАЗАМИ ДАНИХ В MYSQL. ODBC

Мета

– засвоїти правила зв’язування і настроювання MySQL, C++ Builder 6, ODBC Drivers;

– підготовити студентів до самостійного використання засобів програмування мовою C++ Builder 6.

Завдання

1. Для усіх варіантів створити програму в C++ Builder 6, в якій є присутнім компонент PageControl. Програма має:

- містити усі таблиці із лабораторної роботи 16;
- кожна таблиця має відображатися в окремому компоненті DBGrid;
- мають бути присутніми три різні форми для заповнення кожної з таблиць;
- має бути присутньою можливість відчистки таблиць*;
- має бути присутньою можливість додавання поля в таблиці*;
- мають бути присутніми обробники усіх можливих помилок;
- має бути присутнім сортування, групування, фільтрація по вибіркових полях (поля обираються самостійно)*;
- таблиці не повинні редагуватися через компонент DBGrid або компонент DBControl.

2. Скласти звіт з лабораторної роботи.

Примітка. Усі завдання, помічені символом «*» мають бути виконані за допомогою мови SQL.

Порядок виконання лабораторної роботи

18.1. Налаштування і установка ODBC.

– Відкрити папку ODBC.

– Запустити файл Install.bat.

– Перейти в Панель Управління -> Адміністрування і запустити застосування «Джерела Даних (ODBC)».

– Натиснути кнопку «додати» і вибрати драйвер MySQL.

– Натиснути кнопку «готово» і заповнити форму, що з’явилася.

Сервер: 127.0.0.1

Порт: 3306

Ім’я користувача і пароль, які задали при установці, а також назва і опис драйвера, можуть бути будь-якими. Натиснути кнопку «Ок». Повторити дії для вкладки «Системний».

Відкрити C++ Builder 6 і перетягнути на форму компоненти Query, DataSource, DBGrid, зв’язати їх. У полі DatabaseName компонента Query вибрати ім’я бази даних, визначене при створенні драйвера OBDC.

18.2. Сторінки із закладками.

Багатосторінкові вікна діалогу створюються за допомогою компонента, який відображається у вигляді вкладок. Вони спрощують призначений для користувача інтерфейс, замінюючи декілька зв’язаних за змістом форм.

Розглянемо приклад програми створення вкладок.

Крок 1. Відкрити новий проект і встановити для головної форми наступні властивості:

Name = ExamForm

Caption = Іспит

BorderStyle = bsDialog

Position = poScreenCenter

Крок 2. Розмістити на формі компонент PageControl.

Крок 3. Спочатку компонент PageControl не містить жодної вкладки. Для створення вкладки натиснути праву кнопку миші на компоненті і вибрати в контекстному меню команду New Page. Буде створена перша вкладка із заголовком TabSheet1. Кожна вкладка в компоненті PageControl представлена об’єктом класу TTabSheet. Властивості окремої вкладки встановлюються у вікні властивостей.

Крок 4. Перейти до вікна властивостей і замінити текст закладки, вписавши у властивості Caption нове значення.

Крок 5. Спочатку натисканням активізувати вкладку. Потім помістіти на неї групу взаємовиключних перемикачів – компонент RadioGroup. Заголовок групи міститиме умову питання, а текст перемикачів – можливі варіанти відповіді.

Крок 6. Для групи перемикачів установити відповідні розміри і встановити значення наступних властивостей:

Caption = The right expression is (вираз є правильним)

Items = sin 50°< cos 50°

sin 50°> cos 50°

sin 50° = cos 50°

ItemIndex = 0 (номер варіанту, що приймається за умовчання)

Tag = 1 (номер правильного варіанту, вважаючи від нуля).

Крок 7. Помістити на форму кнопку (компонент Button). Ця кнопка служить для закриття вікна, її слід зробити такою:

Name = CloseButton

Caption = Close

Cancel = True

Крок 8. Створити для кнопки обробники події OnClick.

У деяких випадках необхідно відстежити переключення між сторінками. Для цього в компоненті PageControl передбачені події OnChanging і OnChange. Перша подія – це запит на переключення сторінки, а друга – повідомлення про те, що сторінка переключена.

Контрольні питання

1. Поясніть призначення технології ODBC.
2. Який компонент відповідає за роботу із закладками?
3. Назвіть характерні властивості компонента PageControl.
4. Наведіть опис процесу встановлення зв’язку між ODBC та C++ Builder 6.
5. Укажіть особливості роботи з ODBC в C++ Builder 6.
6. Укажіть обмеження в роботі з ODBC в C++ Builder 6.

Лабораторна робота 19

ДОСЛІДЖЕННЯ ЗАСОБІВ СИСТЕМИ ПРОГРАМУВАННЯ

Мета

– ознайомити з переліком та порядком роботи засобів системи програмування;

– засвоїти правила трансляції, редагування та виконання програм на мові Асемблеру;

– підготовити студентів до самостійного використання засобів

програмування мовою Асемблер.

Завдання

1. Створити локальну папку на робочому диску.
2. З робочого диска комп'ютера викладача скопіювати в створену папку архівний файл системи програмування, провести інсталяцію системи програмування (СП).
3. Вивчити параметри транслятора й редактори зв'язків.
4. За вказівкою викладача підготувати програми для налагодження.
5. Провести запуск кожної програми на трансляцію, редагування й виконання через командний рядок.
6. Провести аналіз результатів трансляції й редагування. Особливу увагу звернути на програму в машинних кодах.
7. Підготувати bat-файл для трансляції, редагування й виконання програми.
8. Налагодити програми у вигляді COM та EXE-файла.
9. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Програма, написана мовою Асемблер, так само, як і програма, написана будь-якою іншою мовою програмування, виконується не сама по собі, а за допомогою операційної системи (ОС). ОС виділяє область пам'яті для програми, завантажує її, передає їй керування й забезпечує взаємодію програми з пристроями введення-виведення, файловими системами й іншими програмами (крім тих випадків, коли ця програма сама є ОС або її частиною). Найпростіша й розповсюджена ОС для комп'ютерів, заснованих на процесорах Intel, - DOS (дискова операційна система).

Система DOS надає програмам повну свободу дій, не обмежуючи доступ до пам'яті й зовнішніх пристроїв, дозволяючи їм самим керувати процесором і розподілом пам'яті.

Для того, щоб програма виконувалась будь-якою ОС, вона повинна бути скомпільована у файл, який може бути виконаний. Основні два формати таких файлів в DOS – COM й EXE.

Файли типу COM містять тільки скомпільований код без якої-небудь додаткової інформації про програму. Код, дані й стек такої програми розміщуються в одному сегменті й не можуть перевищувати 64 кілобайта.

Файли типу EXE містять заголовок, у якому описуються: розмір файла; необхідний об'єм пам'яті; список команд у програмі, що використовують абсолютні адреси, які залежать від розміщення програми в пам'яті, тощо. Exe-файл може мати будь-який розмір. Формат EXE також використовується для виконуваних файлів у різних версіях DOS-розширювачів й Windows, але зі значними змінами.

Програма типу COM.

Наберіть у будь-якому текстовому редакторі, що може записувати файли як звичайний текст (наприклад, NOTEPAD в Windows), наступний текст:

```
;hello-1.asm

;Виводить на екран повідомлення “Hello World”

.model Tiny                ;модель пам’яті для COM

.code                      ;початок сегмента коду

org 100h                           ;початкове значення лічильника – 100h
```

```
Start: mov ah,9                         ;номер функції DOS в AH

mov dx,offset  message       ; адреса рядка в DX

int 21h                          ;виклик системної функції DOS

ret                               ;завершення COM-програми

Message Db   “Hello World”,0Dh,0Ah,’$’       ;рядок для виведення

end start                              ;кінець програми
```

і збережіть його як файл hello-1.asm.

Для перетворення програми у файл, що виконується, необхідно:

– викликати Асемблер, для компіляції її в об’єктний файл з ім’ям hello-l.obj:

```
ml /c hello-1.asm
```

– викликати редактор зв’язків і програму перетворення формату link16 hello-1.obj,,NUL,,,:

exe2bin hello-1.exe hello-1.com.

Файл HELLO-1.COM має розмір 23 байта. Якщо його виконати, на екрані з’явиться рядок «Hello World!» і програма завершиться.

Програма типу EXE

Exe-програми більш складні у виконанні, але для них відсутнє обмеження розміру в 64 кілобайта, так що всі великі програми використають саме цей формат.

```
;hello-2.asm

;виводить на екран повідомлення “Hello World!” і завершується;

;компіляція та редагування:

;ml /c hello-2.asm

;link16 hello-2.obj

.model  small           ;модель пам’яті, використовувана для
;EXE формату

.stack    100h          ;сегмент стека 256 байт
```

```
.code                   ;початок сегменту коду

start: mov      ax,DGROUP ;сегментна адреса рядка message

mov     ds,ax           ;міститься в DS

mov     dx,offset message

mov     ah,9

int     21h             ;функція DOS "виведення рядку"

mov     ax,4C00h

int     21h             ;функція DOS "завершити програму"

.data                   ;початок сегменту даних

message  db 'Hello World!',0Dh,0Ah,'$'

end     start
```

Контрольні питання

1. Укажіть склад системи програмування.
2. Укажіть призначення основних програм для трансляції, редагування, перетворення формату.
3. Який порядок оброблення програм на мові Асемблеру?
4. Укажіть параметри трансляції програм.
5. Укажіть особливості оформлення COM-програм, EXE-програм та програм з використанням засобів Windows.

Лабораторна робота 20

ОРГАНІЗАЦІЯ ВИВЕДЕННЯ ДАНИХ

Мета

– ознайомити з переліком та порядком роботи засобів DOS і BIOS для виведення даних на екран у текстовому режимі;

– ознайомити з порядком використання засобів прямої роботи з відеопам’яттю для виведення даних на екран у текстовому режимі;

– підготовити студентів до самостійного використання засобів програмування мовою Асемблер.

Завдання

1. Створити локальну папку на робочому диску.

2. З робочого диска комп'ютера викладача скопіювати в створену папку архівний файл системи програмування, провести інсталяцію СП.

Підготувати 4 програми з розділу «Порядок виконання лабораторної роботи» для налагодження.

1. Підготувати bat-файл для трансляції, редагування й виконання програми.
2. Налагодити програми у вигляді COM-файлів.
3. За вказівкою викладача скорегувати програми відповідно до варіантів індивідуальних завдань і налагодити їх.
4. Скласти звіт з лабораторної роботи.

Варіанти індивідуальних завдань:

1. У програмах № 1, 3 або 4 змінити число символів у рядку з 16-ти на 8.
2. У програмах № 1, 3 або 4 змінити спосіб організації циклу.
3. У програмах № 3,4 змінити атрибут символу на «чорний на білому тлі».
4. У програмі № 1, 3, 4 організувати виведення тільки перших 16-ти символів.

Порядок виконання лабораторної роботи

20.1. Засоби DOS

Існує декілька способів виведення тексту, DOS надає для цього кілька функцій.

Функція DOS 02h – записати символ в STDOUT з перевіркою на Ctrl-Break. Якщо в ході роботи цієї функції була натиснута комбінація клавіш Ctrl-Break, викликається переривання 23h, що за умовчанням здійснює вихід з програми.

Функція DOS 06h – записати символ в STDOUT без перевірки на Ctrl-Break. Не обробляє

керуючі символи (CR, LF, HT й BS виконують свої функції при виведенні на екран, але зберігаються у разі перенапрямлення виведення у файл) і не перевіряє натискання Ctrl-Break.

Функція DOS 09h – записати рядок в STDOUT з перевіркою на Ctrl-Break. Дія цієї функції повністю аналогічна дії функції 02h, але виводиться не один символ, а цілий рядок, як у програмах hello-1.asm й hello-2.asm.

Функція DOS 40h – записати у файл або пристрій. Призначена для запису у файл, але, якщо в регістр BX помістити число 1, функція 40h буде виводити дані на STDOUT, а якщо BX = 2 – на пристрій STDERR.

STDERR завжди виводить дані на екран і не спрямовує у файли. На цій функції засновані використовувані в C функції стандартного виведення – фактично функція C fputs() просто викликає це переривання, розміщуючи свій перший аргумент у BX, адресу рядка (другий аргумент) – в DS:DX і довжину – у CX.

Приклад програми типу COM, що виводить на екран всі ASCII-символи, 16 рядків по 16 символів у рядку.

```
;dosout1.asm

.model  tiny

.code

org     100h                    ;початок COM-файла
```

```
start:

mov     cx,256                  ;вивести 256 символів

mov     dl,0                    ;перший символ з кодом 00

mov     ah,02h                  ;номер функції DOS

;“виведення символу”

cloop:

int     21h                     ;виклик DOS

inc     dl                      ;збільшення DL на 1 –

;наступний символ

test    dl,0Fh                  ;якщо DL – не кратний 16

jnz     continue_loop           ;продовжити цикл,

push    dx                      ;інакше зберегти поточний
```

```
;символ

mov     dl,0Dh                  ;вивести CR

int     21h

mov     dl,0Ah                  ;вивести LF

int     21h

pop     dx                      ;відновити поточний символ

continue_loop:

loop    cloop                   ;продовжити цикл

ret                             ;завершення COM-файла

end   start
```

Приклад програми типу COM, що виводить рядок "This function can print $".

```
;dosout2.asm

;використовує виведення в STDERR

.model  tiny

.code

org     100h            ;початок COM-файла

start:

mov     ah,40h                  ;номер функції DOS

mov     bx,2            ;пристрій

mov     dx,offset message   ;DS:DX – адреса рядка

mov     cx,message_length           ;CX – довжина рядка

int     21h

ret                             ;завершення COM-файла
```

```
message       db                  'This function can print $'

message_length = $-message     ;довжина рядка = поточний адрес

;мінус адрес початку рядка

end     start
```

20.2. Засоби BIOS

Для більш “тонкої” роботи з екраном, використовуються відеофункції BIOS.

BIOS (базова система введення-виведення) – це набір програм, розміщених у постійній пам’яті комп’ютера, які виконують його завантаження відразу після увімкнення й забезпечують доступ до деяких пристроїв, зокрема до відеоадаптера.

Усі функції відеосервісу BIOS викликаються через переривання 10h.

Розглянемо функції, які можуть бути корисні для виведення текстів на екран.

Вибір відеорежиму.

BIOS надає можливість перемикання екрана в різні текстові й графічні режими. Режими відрізняються один від одного роздільною здатністю (для графічних) і кількістю рядків і

стовпчиків (для текстових), а також кількістю можливих кольорів.

Функція INT 10h, AH = 00 – встановити відеорежим.

Номера текстових режимів – 0,1,2,3 й 7.

0 й 1 – 16-кольорові режими 40x25 (з 25 рядками по 40 символів у рядку);

2 й 3 – 16-кольорові режими 80x25;

7 – монохромний режим 80x25.

Функція INT 10h, AH = 4Fh, AL = 02 – установити SuperVGA-видеорежим.

Керування положенням курсору.

Функція INT 10h, AH = 02 – встановити положення курсору.

Функція INT 10h, AH = 03 – зчитати положення й розмір курсору, повертається поточний стан курсору на обраній сторінці.

Функція INT40h, AH = 08 – зчитати символ й атрибут символу в поточній позиції курсору.

Функція INT 10h, AH = 0Ah – вивести символ з поточним атрибутом на екран.

Функція INT 10h, AH = 0Eh – вивести символ у режимі телетайпа.

Функція INT 10h, AH = 13h – вивести рядок символів із заданими атрибутами.

Приклад програми виведення засобами BIOS.

```
;biosout.asm

.model tiny

.code

org     100h            ; початок COM-файла

start:

mov     ax,0003h

int     10h             ; відеорежим 3 (очищення екрана й

;установка курсору в 0,0)
```

```
mov     dx,0            ; DH й DL будуть використатися для

;зберігання положення курсору ;початкове положення – 0,0

mov     si,256          ;SI – лічильник циклу

mov     al,0            ;перший символ з кодом 00h

mov     ah,9            ;номер відеофункції "виведення

;символу з атрибутом"

mov     cx,1             ;виводиться один символ за раз

mov     bl,00011111b        ;атрибут символу – білий на синьому

cloop:

int     10h             ;вивести символ на екран

push    ax              ;зберегти поточний символ і номер

;функції
```

```
mov     ah,2            ;номер відеофункції 2 – змінити

;положення курсору

inc     dl              ;збільшити поточну колонку на 1

int     10h             ;перемістити курсор

mov     ax,0920h        ;AH = 09, AL = 20h (ASCII-код пробілу)

int     10h             ;вивести пробіл

mov     ah,2            ;номер відеофункції 2

inc     dl              ;збільшити колонку на 1

int     10h             ;перемістити курсор

pop     ax              ;відновити номер функції в ah і

;поточний символ в al
```

```
inc     al              ;збільшити AL на 1 – наступний символ

test    al,0Fh          ;якщо AL не кратний 16,

jnz     continue_loop   ;продовжити цикл,

push    ax              ;інакше зберегти номер функції й

;поточний символ

mov     ah,2            ;номер відеофункції 2

inc     dh              ;збільшити номер рядка на 1

mov     dl,0            ;колонка = 0

int     10h             ;установити курсор на початок
;наступного рядка

pop     ax              ;відновити номер відеофункції й
;поточний символ

continue_loop:
```

```
dec     si              ;зменшити SI на 1, якщо він не                  ;став
нулем продовжити

jnz     cloop           ;CX використається усередині циклу,

;так що не можна використати команду

;LOOP для його організації

ret                     ;завершення COM-файла

end     start
```

20.3. Безпосередня робота з відеопам’яттю.

Графіка, текст, які зображені на моніторі, одночасно наявні і знаходяться у пам’яті відеоадаптера. Для того, щоб зображення з’явилося на моніторі, воно повинно бути записане в пам’ять відеоадаптера. Для цього виділяється спеціальна область пам’яті, яка починається з абсолютної адреси B800h:0000h (для текстових режимів) і закінчується на B800h:FFFFh.

Дані, які програми записують у цю область пам’яті, негайно пересилаються в пам’ять відеоадаптера. Будь-яка програма може вивести текст на екран командою пересилання даних, не вдаючись до спеціальних функцій DOS або BIOS.

Приклад програми виведення через пам’ять відеоадаптера.

```
;dirout.asm

;виводить на екран всі ASCII-символи без винятку,

;використовуючи пряме виведення на екран

.model  tiny

.code

.386                    ;буде використаний регістр EAX і

;команда STOSD

org     100h            ; початок COM-файла

start:

mov     ax,0003h

int     10h             ;відеорежим 3 (очищення екрана)

cld                     ;обробка рядків, підготовка даних для
```

```
;виведення на екран

mov    eax,1F201F00h        ;перший символ 00 з атрибутом 1Fh,

;потім пробіл (20h) з атрибутом 1Fh

mov       bx,0F20h       ;пробіл з атрибутом 0Fh

mov       cx,255                  ;число символів мінус 1

mov    di,offset ctable          ;ES:DI – початок таблиці

cloop:

stosd                        ;записати символ і пробіл у таблицю

;ctable

inc       al                 ;AL містить наступний символ

test      cx,0Fh                  ;якщо CX не кратний 16,
```

```
jnz         continue_loop         ;продовжити цикл,

push     cx                  ;інакше зберегти значення
;лічильника

mov       cx,80-32        ;число символів, що залишилися до

;кінця рядка

xchg     ax,bx

rep        stosw             ;заповнити залишок рядка пробілами

;з атрибутом 0F

xchg     bx,ax             ;відновити значення EAX

pop       cx                  ;відновити значення лічильника

continue_loop:

loop      cloop

stosd                        ;записати останній (256-й) символ і
```

```
;пробіл

;виведення на екран

mov     ax,0B800h    ;сегментний адрес відеопам’яті

mov     es,ax

xor     di,di              ;DI = 0, адрес початку відеопам’яті в

;ES:DI

mov     si,offset ctable        ;адрес таблиці в DS:SI

mov     cx,15*80+32 ;15 рядків по 80 символів, останній
;рядок – 32

rep     movsw                   ;скопіювати таблицю ctable
      ;у відеопам’ять

ret                             ;завершення COM-файла

ctable:                         ;дані для виведення на екран починаються
```

;відразу за кінцем файла. В EXE-файлі такі дані

;визначають у сегменті .data

end start

Контрольні питання

1. Наведіть способи виведення даних на екран.
2. Які функції DOS для виведення даних у текстовому режимі?
3. Які функції BIOS для виведення даних у текстовому режимі?
4. Укажіть параметри відеорежиму.
5. Наведіть засоби роботи з відеопам'яттю.

Лабораторна робота № 21

РОЗРОБЛЕННЯ ЗАСОБІВ ВВЕДЕННЯ ДАНИХ

Мета

– ознайомити з переліком та порядком роботи засобів DOS для введення даних з клавіатури;

– ознайомити з переліком та порядком роботи засобів BIOS для введення даних з клавіатури;

– підготовити студентів до самостійного використання засобів програмування мовою Асемблер.

Завдання

1. Створити локальну папку на робочому диску.
2. З робочого диска комп'ютера викладача скопіювати в створену папку архівний файл системи програмування, провести інсталяцію СП.
3. Підготувати 2 програми з розділу «Порядок виконання лабораторної роботи» для налагодження. Звернути увагу на порядок програмування циклів, адресації даних, способи виклику процедур, алгоритм перекладу введених символів у число, організацію буфера для введення даних.
4. Підготувати bat-файл для трансляції, редагування й виконання програми.
5. Налагодити програми у вигляді COM-файлів.
6. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

21.1 Засоби DOS

DOS надає набір функцій для зчитування даних із клавіатури, які використовують стандартний пристрій введення STDIN.

Функція DOS 0Ah – зчитати рядок символів з STDIN у буфер. Для виклику цієї функції треба підготувати буфер, перший байт якого містить максимальну кількість символів для введення (1-254), у другий байт записується довжина реально введеного рядка без урахування останнього CR.

Функція DOS 01h – зчитати символ з STDIN з відлунням, очікуванням і перевіркою на Ctrl-Break.

Функція DOS 08h – зчитати символ з STDIN без відлуння, з очікуванням і перевіркою на Ctrl-Break.

Функція DOS 07h – зчитати символ з STDIN без відлуння, з очікуванням і без перевірки на Ctrl-Break.

Функція DOS 06h – зчитати символ з STDIN без відлуння, без очікування й без перевірки на Ctrl-Break.

Функція DOS 0Bh – перевірити стан клавіатури.

Функція DOS 0Ch – очистити буфер і зчитати символ.

Функції посимвольного введення без відлуння можна використати для інтерактивного керування програмою.

Приклад програми, що виконує перетворення десяткового числа в шістнадцяткове.

```
;dosin1.asm

.model  tiny

.code
```

```
.286                    ;для команди  shr al,4

org     100h            ;початок COM-файла

start:

mov     dx,offset message1

mov     ah,9

int     21h             ;вивести запрошення до введення
;message1

mov     dx,offset buffer

mov     ah,0Ah

int     21h             ;записати рядок символів у буфер

mov     dx,offset crlf

mov     ah,9
```

```
int         21h             ;переведення рядка

;переведення числа в ASCII-форматі з буфера в бінарне число в AX

xor         di,di           ;DI = 0 – номер байта в буфері

xor         ax,ax           ;AX = 0 – поточне значення результату

mov       cl,blength

xor         ch,ch

xor         bx,bx

mov       si,cx             ;SI – довжина буфера

mov       cl,10             ;CL = 10 – множник для MUL

asc2hex:

mov       bl,byte ptr bcontents[di]

sub       bl,'0'            ;цифра = код цифри – код символа "0"
```

```
jb      asc_error    ;якщо код символу менше, ніж код "0"

cmp     bl,9         ;або більше, ніж "9"

ja      asc_error    ;вийти із програми з повідомленням про

;помилку,

mul     cx           ;інакше – помножити поточний

;результат на 10

add     ax,bx        ;додати до нього нову цифру

inc     di           ;збільшити лічильник

cmp     di,si        ;якщо лічильник + 1 менше числа

;символів

jb      asc2hex      ;продовжити (лічильник починається
```

```
;від 0)

;виведення на екран рядка message2

push    ax              ;зберегти результат перетворення

mov     ah,9

mov     dx,offset message2

int     21h

pop     ax

;виведення на екран числа з регістра AX

push    ax

xchg    ah,al           ;помістити в AL старший байт

call    print_al                ;вивести його на екран

pop     ax              ;відновити в AL молодший байт
```

```
call    print_al                ;вивести його на екран

ret                             ;завершення COM-файла

asc_error:

mov     dx,offset err_msg

mov     ah,9

int     21h             ;вивести повідомлення про помилку

ret                             ;завершити програму

;процедура print_al.

;виводить на екран число в регістрі AL у шістнадцятковому форматі,

;модифікує значення регістрів AX й DX

print_al:
```

```
mov       dh,al

and       dh,0Fh                    ;DH – молодші 4 біти

shr       al,4                ;AL – старші

call      print_nibble  ;вивести старшу цифру

mov       al,dh               ;тепер AL містить молодші 4 біти

print_nibble:               ;процедура ведення 4 біт

;(шістнадцяткове число)

cmp       al,10               ;три команди, що переводять

;цифру в AL

sbb       al,69h              ;у відповідний ASCII-код

das                           ;(див. опис команди DAS)

mov       dl,al               ;код символу в DL
```

```
mov     ah,2            ;номер функції DOS в AH

int     21h             ;виведення символу

ret                     ;RET працює 2 рази – один раз для

;повернення із процедури print_nibble,

;викликаної для старшої цифри, і другий

;раз – для повернення з print_al

message1    db      'Enter decimal number: $'

message2    db      'Hex number is: $'

err_msg     db      'Bad number entered'

crlf        db      0Dh,0Ah,'$'

buffer      db      6               ;максимальний розмір буфера введення
```

```
blength                 db      ?       ;розмір буфера після зчитування

bcontents:                      ;вміст буфера розташовується за

;кінцем COM-файла

end     start
```

21.2. Засоби BIOS

BIOS надає більше можливостей порівняно з DOS для зчитування даних і керування клавіатурою, використовуючи різні функції переривання 16h й операції з байтами стану клавіатури.

Функція INT 16h, AH = 0, 10h, 20h – зчитування символу з очікуванням.

Функція INT 16h, AH = 1, 11h, 21h – перевірка символу.

Функція INT 16h, AH = 05h – помістити символ у буфер клавіатури.

Функція INT 16h, AH = 02h, 12h, 22h – зчитати стан клавіатури.

Приклад програми, що заносить у буфер клавіатури команду DIR.

```
;ungetch.asm заносить у буфер клавіатури команду DIR, так, щоб

;виконалася відразу після завершення програми

.model  tiny

.code

org     100h            ;COM-файл

start:

mov     cl,'d'          ;CL = ASCII-код букви "d"

call    ungetch

mov     cl,'i'          ;ASCII-код букви "i"

call    ungetch

mov     cl,'r'          ;ASCII-код букви "r"

call    ungetch
```

```
mov     cl,0Dh                  ;переведення рядка

ungetch:

mov     ah,5            ;AH = номер функції

mov     ch,0            ;CH = 0 (скан-код неважливий)

int     16h             ;помістити символ у буфер

ret                     ;завершити програму

end     start
```

Безпосередній доступ до буфера клавіатури швидший, ніж виклик відповідних функцій BIOS, і для програм, що потребують максимальної швидкості, таких як ігри або демо-програми, використовують керування клавіатурою на рівні портів введення-виведення.

Контрольні питання

1. Наведіть засоби введення даних.
2. Які функції DOS використовують для введення даних?
3. Вкажіть особливості кодування даних при введенні з клавіатури.
4. Які функції BIOS використовують для введення даних?
5. За допомогою якої команди забезпечується доступ до буферу клавіатури?

Лабораторна робота 22

ОБРОБЛЕННЯ ДАНИХ З ПІДВИЩЕНОЮ ТОЧНІСТЮ

Мета

– ознайомити із прийомами програмування оброблення даних підвищеної точності;

– підготовити студентів до самостійного використання засобів програмування мовою Асемблер.

Завдання

1. Створити локальну папку на робочому диску.
2. З робочого диска комп'ютера викладача скопіювати в створену папку архівний файл системи програмування, провести інсталяцію СП.
3. Відповідно до варіанта завдання, зазначеного викладачем, підготувати для налагодження 2 програми (Програма 1 – обробка цілочисельних даних, Програма 2 – обробка даних з фіксованою комою). Особливу увагу звернути на особливості програмування різних операцій над даними підвищеної точності, адресацію даних, способи визначення даних.
4. Підготувати bat-файл для трансляції, редагування й виконання програми.
5. Налагодити програми у вигляді COM-файлів.
6. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Таблиця 22.1

Варіанти програми 1 «Цілочисельна арифметика»

Вид операції	Розмірність числа (біт)	Додавання
40	48	56
64		

72

80

88

104

112

120

Віднімання

40

48

56

64

72

80

88

104

112

120

Порівняння

40

48

56

64

72

80

88

104

112

120

Номер варіанта дорівнює двозначному номеру студента за списком групи, старша цифра вибирається із лівого стовпця (номер виду операції), молодша цифра відповідає варіанту розмірності даних.

Конкретні значення чисел вибираються студентами самостійно й повинні доводити правильність роботи програми. Вони можуть бути задані безпосередньо в програмі або введені з монітора.

Значення чисел можуть бути задані безпосередньо в програмі або введені з монітора.

Розроблена програма повинна видавати вихідні числа й результат на екран. Для цього доцільно використати процедуру PRINT_AL з лабораторної роботи № 21.

Таблиця 22.2

Варіанти програми 2 «Обчислення з фіксованою комою»

Вид операції	Формат числа (біт: біт)
Ділення	16:16
	8:8
Множення	16:16
	8:8
Додавання	16:16

8:8

Віднімання

16:16

8:8

Примітка. Номер варіанта – за вказівкою викладача.

22.1. Цілочисельна арифметика

Мови високого рівня зазвичай обмежені в наборі типів даних, які вони можуть оброблювати. Для зберігання цілих чисел застосовують окремі байти, слова або подвійні слова. Використовуючи типи даних будь-якого розміру (64 біта, 128 біт, 1024 біта) можливо визначити всі арифметичні операції з такими числами.

При додаванні використовують команди ADC та ADD (додавання з урахуванням переносу). Спочатку складають наймолодші байти, слова або подвійні слова командою ADD, а потім складають все інше – команда ADC, рухаючись від молодшого розряду числа до старшого.

При відніманні застосовують команди SUB та SBB (віднімання з урахуванням займу), які діють аналогічно.

Для порівняння застосовують команду CMP, яка еквівалентна команді віднімання. У

більшості випадків, для визначення результату порівняння досить зрівняти найстарші слова (байти або подвійні слова), і тільки, якщо вони в точності рівні, буде потрібно порівняння наступних слів.

Щоб помножити числа підвищеної точності використовують команду MUL. Алгоритм множення аналогічний правилу множення десяткових чисел у стовпчик.

Операцію ділення будь-якого числа на слово або подвійне слово можна виконати за допомогою команди DIV. Ділення будь-якого іншого числа аналогічно загальній операції ділення, досить додати потрібне число трійок команд mov/div/mov у початок алгоритму.

Приклад програми додавання двох цілих 96-бітових чисел.

```
;числа задані безпосередньо в програмі

;результат виводиться на екран

;pr5.asm

;додавання чисел підвищеної точності 96 біт

.model          tiny

.code

.386
```

```
org     100h                    ;початок COM-файла

start:

;видача повідомлення mess1

mov    ah,9                     ;номер функції DOS – в AH

mov    dx,offset mess1                  ;адреса рядка – в DX

int    21h                      ;виклик функції DOS

mov    dx,offset crlf

mov    ah,9

int    21h                      ;переведення рядка

;власне розрахунок – сума двох 96-бітних чисел bigval_1 й bigval_2

;результат зберігається в bigval_3
```

```
mov    eax,dword ptr bigval_1[8]

add    eax,dword ptr bigval_2[8]   ;скласти молодші слова

mov    dword ptr bigval_3[8],eax   ;зберегти результат

mov    eax,dword ptr bigval_1[4]

adc    eax,dword ptr bigval_2[4]   ;скласти середні слова

mov    dword ptr bigval_3[4],eax

mov    eax,dword ptr bigval_1

adc    eax,dword ptr bigval_2                ;скласти старші слова

mov    dword ptr bigval_3,eax

;друк результату

call    print                              ;друк EAX – старші

;байти
```

```
mov    eax,dword ptr bigval_3[4]   ;друк EAX – середні

;байти

call      print

mov    eax,dword ptr bigval_3[8]   ;друк EAX – молодші

;байти

call      print

mov    dx,offset crlf

mov    ah,9

int       21h                         ;переведення рядка

ret

mess1   db    'REZULTAT = $'
```

```
crlf      db      0Dh,0Ah,'$'

bigval_1    dd   1Ah,1Ah,1Fh                ;перше 96-бітне число

bigval_2    dd   6h,6h,1h                  ;друге 96-бітне число

bigval_3    dd   0h,0h,0h                  ;місце для зберігання

;результату

;процедура побайтної видачі вмісту регістра EAX

print:

mov    esi,eax                             ;збереження числа

shr        eax,24                          ;підготовка до друку старшого

;(четвертого) байта

call print_al                          ;виведення двох цифр числа

mov       eax,esi
```

```
shr         eax,16                          ;підготовка до видачі третього

;байта

call print_al                       ;виведення двох цифр числа

mov       eax,esi

shr         eax,8                    ;підготовка до видачі другого

;байта

call print_al                       ;виведення двох цифр числа

mov       eax,esi                           ;підготовка до видачі першого

;байта

call print_al                       ;виведення двох цифр числа

ret
```

```
;процедура print_al

;виводить на екран число в регістрі AL у шістнадцятковому

;форматі

print_al:

mov       dh,al

and       dh,0Fh                    ;DH – молодші чотири біти

shr       al,4                      ;AL – старші

call      print_nibble              ;вивести старшу цифру

mov       al,dh                     ;тепер AL містить молодші

;чотири біти

print_nibble:                       ;процедура виведення чотирьох

;біт (шістнадцяткової цифри)
```

```
cmp     al,10                   ;три команди, що переводять

;цифру в AL

sbb     al,69h                  ;у відповідний ASCII-код

das                              ;(див. опис команди DAS)

mov     dl,al                   ;код символу – в DL

mov     ah,2                    ;номер функції DOS в AH

int     21h                     ;виклик функції

ret                              ;повернення з процедури

end     start
```

22.2. Обчислення з фіксованою комою

Найпоширеніші формати для чисел з фіксованою комою – 8:8 й 16:16. У першому випадку, на цілу й на дробову частини числа відводиться по одному байту, а в другому – по одному слову. Операції із цими двома форматами можна виконувати, вміщуючи число в регістр (16-бітний – для формату 8:8 й 32-бітний – для формату 16:16).

Додавання й віднімання для чисел з фіксованою комою не відрізняються від додавання й віднімання цілих чисел.

При множенні варто пам’ятати, що множення 16-бітних чисел дає 32-бітний результат, а множення 32-бітних чисел – 64-бітний результат.

При операції ділення, число, записане з фіксованою комою у форматі 16:16, можна представити як число, помножене на 216. Якщо розділити такі числа одне на одне відразу, отримаємо результат ділення цілих чисел: . Щоб результат мав потрібний нам вигляд , треба заздалегідь помножити ділене на 216.

Приклад програми ділення чисел з фіксованою комою.

```
; pr6.asm

;ділення чисел з фіксованою комою без знака у форматі16:16

;числа – однакові за форматом

.model          tiny

.code

.386
```

```
org     100h                    ;початок COM-файла

start:

;виведення повідомлення mess1

mov    ah,9                     ; номер функції DOS – в AH

mov    dx,offset mess1          ;адреса рядка – в DX

int    21h                      ;виклик функції DOS

mov    dx,offset crlf

mov    ah,9

int    21h                      ;переведення рядка

;власне розрахунок – ділене в EAX, дільник – в EBX

mov    eax,000B0000h            ;ділене дорівнює 11,0

mov    ebx,000A0000h            ;дільник дорівнює 10,0
```

```
xor     edx,edx                     ;очищення edx

ror     eax,16                      ;циклічний зсув на 16 розрядів,

;це еквівалентно множенню на 2

;в 16-му ступені

xchg   ax,dx                        ;EDX:EAX = EAX * 2 в 16-му

;ступені

div     ebx                         ;EAX = результат ділення, що

;дорівнює 1,1 або ;1,1999h,

;друк результату

call    print                       ;друк EAX

mov    dx,offset crlf
```

```
mov    ah,9

int       21h                        ;переведення рядка

ret

mess1            db       'REZULTAT = $'

crlf      db       0Dh,0Ah,'$'

;процедура побайтної видачі вмісту регістра EAX

print:

mov    esi,eax                        ;збереження числа

shr       eax,24                        ;підготовка до друку старшого

;(четвертого) байта

call print_al                     ;виведення двох цифр числа

mov    eax,esi
```

```
shr     eax,16                      ;підготовка до друку третього

;байта

call print_al                   ;виведення двох цифр числа

mov    eax,esi

shr     eax,8                    ;підготовка до друку другого

;байта

call print_al                   ;виведення двох цифр числа

mov    eax,esi                      ;підготовка до друку першого

;байта

call print_al                   ;виведення двох цифр числа

ret
```

```
;процедура print_al

;виводить на екран число в регістрі AL у шістнадцятковому форматі

print_al:

mov    dh,al

and    dh,0Fh                    ;DH – молодші чотири біти

shr    al,4                      ;AL – старші

call print_nibble                ;вивести старшу цифру



mov al,dh                        ;тепер AL містить молодші

;чотири біти

print_nibble:

;процедура виведення чотирьох біт (шістнадцяткової цифри)
```

```
cmp al,10                    ;три команди, що переводять

;цифру в AL

sbb al,69h                   ;у відповідний ASCII-код

das                           ;(див. опис команди DAS)

mov   dl,al                  ;код символу – в DL

mov   ah,2                   ;номер функції DOS в AH

;(виведення символу)

int     21h                  ;виклик функції

ret                           ;повернення із процедури

end   start
```

Контрольні питання

1. Опишіть алгоритм виконання операції складання для великих чисел.

2. Опишіть алгоритм виконання операції множення для великих чисел.

3. Опишіть алгоритм виконання операції віднімання для великих чисел.

4. Опишіть алгоритм виконання операції ділення для великих чисел.

Лабораторна робота 23

АНАЛІЗ ХАРАКТЕРИСТИК МІКРОПРОЦЕСОРІВ

ЗАСОБАМИ АСЕМБЛЕРУ

Мета

– засвоїти порядок використання команди CPUID;

– підготовити студентів до самостійного використання засобів програмування мовою Асемблер.

Завдання

1. Створити локальну папку на робочому диску.
2. З робочого диска комп'ютера викладача скопіювати в створену папку архівний файл системи програмування, провести інсталяцію СП.
3. Відповідно до варіанта завдання, зазначеного викладачем, підготувати для налагодження програму аналізу характеристик мікропроцесора, використовуючи команду CPUID. Особливу увагу звернути на особливості програмування різних режимів команди CPUID.
4. Підготувати bat-файл для трансляції, редагування й виконання програми.
5. Налагодити програми у вигляді COM-файлів.
6. Скласти звіт з лабораторної роботи.

Розроблена програма повинна видавати результат на екран. Для цього доцільно використати процедуру PRINT_AL з лабораторної роботи 21.

Порядок виконання лабораторної роботи

Варіанти лабораторної роботи:

1. Визначити виробника й версію процесора.
2. Визначити максимальний номер розширеної функції CPUID.
3. Розшифрувати EAX при EAX = 1.
4. Розшифрувати EDX при EAX = 1.
5. і т.д. при EAX = 2, 3.

Команда CPUID надає можливість отримати інформацію про виробника, тип і модифікацію процесора, про наявність і підтримку різних розширень.

Результат роботи CPUID залежить від значення регістра EAX. Якщо EAX – 0, CPUID повертає в EAX максимальне значення, з яким її можна викликати (2 для P6, 1 для P5), а регістри EBX:ECX:EDX містять 12-байтний рядок – ідентифікатор виробника (табл. 23.1).

Таблиця 23.1

Рядки виробників в CPUID

Виробник

Рядок в EBX ECX EDX

Intel

GenuineIntel

UMC

UMC UMC UMC

Cyrix

CyrixInstead

AMD

AuthenlicAMD

NeGen

NexGenDriven

Centaur Technology

CentaurHalls

Наприклад, для процесорів Intel регістр EBX містить «Genu» (756E6547h), ECX містить «Intel» (49656E69H), а EDX – «Intel» (6C6574GEh).

Якщо EAX = 1, CPUID повертає в EAX інформацію про версії процесора, а в EDX – інформацію про підтримувані розширення.

Деякі описані поняття належать до роботи процесора в захищеному режимі.

Приклад програми визначення виробника й версії процесора.

;pr7.asm

;використання команди CPUID

```
.model          tiny

.code

.686p

org     100h                    ;початок COM-файла

start:

;виведення повідомлення mess1

mov    ah,9                     ;номер функції DOS – в AH

mov    dx,offset mess1                  ;адреса рядка – в DX

int     21h                     ;виклик функції DOS

;визначення виробника

xor     eax,eax
```

```
cpuid

;друк EAX

call    print                   ;друк EAX

mov   dx,offset crlf

mov   ah,9

int     21h                     ;переведення рядка

;виведення повідомлення mess2

mov   ah,9                      ;номер функції DOS – в AH

mov   dx,offset mess2               ;адреса рядка – в DX

int     21h                     ;виклик функції DOS

xor     eax,eax

cpuid
```

```
;друк EBX

mov    dword ptr rez,ebx

mov    dx,offset rez

mov    ah,9

int    21h                    ;друк рядка rez

;друк EDX

xor    eax,eax

cpuid

mov    dword ptr rez,edx

mov    dx,offset rez

mov    ah,9
```

```
int     21h                 ;друк рядка rez

;друк ECX

xor     eax,eax

cpuid

mov    dword ptr rez,ecx

mov    dx,offset rez

mov    ah,9

int     21h                 ;друк рядка rez

mov    dx,offset crlf

mov    ah,9

int     21h                 ;переведення рядка

;визначення версії
```

```
;виведення повідомлення mess3

mov    ah,9                    ;номер функції DOS – в AH

mov    dx,offset mess3                ;адреса рядка – в DX

int    21h                     ;виклик функції DOS

mov    eax,1

cpuid

;друк EAX

call    print                  ;друк EAX

mov    dx,offset crlf

mov    ah,9

int    21h                     ;переведення рядка
```

```
ret

mess1   db    'EAX = $'

mess2   db    'EBX,ECX,EDX = $'

mess3   db    'VERSION CPU = $'

crlf    db      0Dh,0Ah,'$'

rez     db      "    $"

;процедура побайтного виведення вмісту регістра EAX

print:

mov     esi,eax                         ;збереження числа

shr     eax,24                          ;підготовка до друку старшого

;четвертого байта

call print_al                   ;виведення двох цифр числа
```

```
mov     eax,esi

shr     eax,16                  ;підготовка до друку третього

;байта

call print_al                   ;виведення двох цифр числа

mov     eax,esi

shr     eax,8                   ;підготовка до друку другого

;байта

call print_al                   ;виведення двох цифр числа

mov     eax,esi                 ;підготовка до друку першого

;байта

call print_al                   ;виведення двох цифр числа
```

```
ret

;процедура print_al виводить на екран число в регістрі AL

;у шістнадцятковому форматі

print_al:

mov       dh,al

and       dh,0Fh                    ;DH – молодші чотири біти

shr       al,4                      ;AL – старші чотири біти

call print_nibble                   ;вивести старшу цифру

mov       al,dh                     ;тепер AL містить молодші

;чотири біти

print_nibble:

;процедура виведення чотирьох біт (шістнадцяткове число)
```

```
cmp al,10                       ;три команди, що переводять

;число в AL

sbb     al,69h                  ;у відповідний ASCII-код

das                             ;(див. опис команди DAS)

mov     dl,al                   ;код символу – в DL

mov ah,2                        ;номер функції DOS в AH

int     21h                     ;виклик функції

ret                             ;повернення із процедури

end   start
```

Контрольні питання

1. Поясніть призначення команди CPUID.
2. Яким чином перевіряється виробник мікропроцесора?
3. Яким чином можливо визначити ідентифікатор виробника?
4. При якому значенні регістра EAX визначається версія процесора?
5. У якому регістрі міститься інформація про підтримувані розширення та при якому

значенні?

Лабораторна робота 24

ВИМІРЮВАННЯ ПРОДУКТИВНОСТІ МІКРОПРОЦЕСОРІВ

Мета

– ознайомити студентів із способами використання таймеру мікропроцесора;

– підготовити студентів до самостійного використання засобів програмування мовою Асемблер.

Завдання

1. Створити локальну папку на робочому диску.
2. З робочого диска комп'ютера викладача скопіювати в створену папку архівний файл системи програмування, провести інсталяцію СП.
3. Відповідно до варіанта завдання, зазначеного викладачем, підготувати для налагодження програму виміру продуктивності МП.
4. Підготувати bat-файл для трансляції, редагування й виконання програми.
5. Налагодити програми у вигляді COM-файлів.
6. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Варіанти лабораторної роботи: *

1-4. Визначити продуктивність процесора для команд регістрового множення, ділення, додавання, віднімання.

5. Визначити продуктивність процесора для команд запису в пам’ять .

6-9. Визначити продуктивність процесора для команд множення, ділення, додавання, віднімання формату регістр-пам’ять.

* Кожний з варіантів може бути реалізований шляхом прямого читання значень лічильника часу або через функцію «Визначити час».

Продуктивність мікропроцесора є комплексною характеристикою, що залежить передусім від особливостей апаратної реалізації його елементів, системи команд, способів роботи з пам’яттю тощо. Найпростіша характеристика продуктивності універсального процесора (FLOPS) показує кількість виконуваних ним операцій з даними із плаваючою крапкою за секунду. Для спеціалізованих МП використовуються інші характеристики, наприклад, продуктивність при виконанні швидкого перетворення Фур’є.

У рамках лабораторної роботи буде розглянутий спосіб виміру продуктивності МП на базі швидкості виконання команд певного типу: множення, ділення й т.д., тобто число виконаних команд за одиницю часу.

Принцип виміру полягає в наступному:

1. Фіксується показник таймера ЕОМ.
2. Виконується цикл з великою кількістю повторень, що містить у тілі команду заданого виду.
3. Фіксується показник таймера.
4. Для визначення продуктивності МП (кількість виконаних команд певного виду в одиницю часу) обчислюється різниця в показниках таймера.
5. Загальне число повторень циклу ділиться на розраховану в п.4 різницю.
6. На основі отриманого значення продуктивності й тактової частоти процесора

обчислюється число тактів виконання команди.

Приклад програми для виміру продуктивності комп’ютера**.**

```
;izm.asm

;у програмі організований цикл із кількістю повторень 70FFFFFh

;у тілі циклу – команда завантаження заданого числа в регістр і

;його ділення на інше задане число, перед і після циклу фіксуються

;показники таймера із запам’ятовуванням їх у відповідних полях

;далі отримані значення виводяться на екран

.model   tiny

.code

.386

org         100h                    ;початок COM-файла
```

```
start:

mov     eax,70FFFFFh            ;завантажити в eax кількість

;повторень циклу

mov     esi, eax                ;і перевантажити його в esi

;фіксування показників таймера

mov     ah,2Ch                  ;звертання до таймера

int     21h                     ;в DH – секунди

;в Dl – соті частки секунди

mov     byte ptr sec1,dh        ;запам’ятовування отриманих

mov     byte ptr ssec1,dl       ;значень.

;початок циклу
```

```
oop:  mov      eax,dword ptr seed  ;завантажити в регістр eax

;вихідне число

div dword ptr rand_m            ;розділити на число

dec esi                         ;зменшити лічильник циклу на

;одиницю

jnz        oop                  ;перейти на початок циклу, якщо

;лічильник не нуль

;фіксування показників таймеру

mov        ah,2Ch               ;звернення до таймера

int        21h                  ;в DH – секунди

;в Dl – соті частки секунди

mov byte ptr sec2,dh            ;запам’ятовування
```

```
mov byte ptr ssec2,dl           ;отриманих значень

;виведення результатів

mov     ah,9                    ;виведення повідомлення mess1

mov     dx,offset mess1

int     21h

mov     al,byte ptr sec1                ;виведення секунд і сотих часток

call print_al                   ;початку відліку

mov     al,byte ptr ssec1

call    print_al

mov     dx,offset crlf

mov     ah,9
```

```
int     21h                 ;переведення рядка

mov     ah,9                ;виведення повідомлення mess2

mov     dx,offset mess2

int     21h

mov     al,byte ptr sec2            ;виведення секунд і сотих часток

call    print_al                    ;закінчення відліку

mov     al,byte ptr ssec2

call    print_al

ret

sec1   db     ?            ;поле для зберігання секунд початку

;відліку

ssec1 db     ?             ;поле для зберігання сотих часток
```

```
;секунд початку відліку

sec2   db      ?               ;поле для зберігання секунд закінчення

;циклу

ssec2 db       ?               ;поле для зберігання сотих часток

;секунд закінчення циклу

mess1          db      'Nachalo - $'

mess2          db      'Okonchanie - $'

crlf      db      0Dh,0Ah,'$'

rand_a         dw      9621h

rand_m         dw      7FFFh

seed   dw      999
```

```
cycl    dw      10

;процедура print_al виводить на екран число в регістрі AL

print_al:

mov     dh,al

and     dh,0Fh                  ;DH – молодші чотири біти

shr     al,4                    ;AL – старші чотири біти

call    print_nibble            ;вивести старшу цифру

mov     al,dh                   ;тепер AL містить молодші

;чотири біти

print_nibble:                   ;процедура виведення

;чотирьох біт

cmp     al,10                   ;три команди, що переводять
```

```
;цифру в AL

sbb     al,69h              ;у відповідний ASCII-код

das                          ;(див. опис команди DAS)

mov     dl,al               ;код символу – в DL

mov     ah,2                ;номер функції DOS в AH

int     21h                 ;виклик функції

ret                          ;повернення із процедури

end   start
```

Контрольні питання

1. Як визначається продуктивність МП?
2. Вкажіть порядок вимірювання продуктивності МП.
3. Вкажіть засоби роботи із системним таймером.
4. За допомогою яких команд організовується цикл?
5. Опишіть виведення значень продуктивності на екран.

Лабораторна робота 25

РОЗРОБЛЕННЯ ЗАСОБІВ ШИФРУВАННЯ ДАНИХ

Мета

– ознайомити студентів із засобами генерації випадкових чисел;

– підготовити студентів до самостійного використання засобів програмування мовою Асемблер.

Завдання

1. Створити локальну папку на робочому диску.
2. З робочого диска комп'ютера викладача скопіювати в створену папку архівний файл системи програмування, провести інсталяцію СП.
3. Відповідно до варіанта завдання, зазначеного викладачем, підготувати для налагодження програму шифрування даних. Особливу увагу звернути на роботу генератора випадкових чисел і порядок визначення даних програми.
4. Підготувати bat-файл для трансляції, редагування й виконання програми.
5. Налагодити програми у вигляді COM-файлів.
6. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Наведений спосіб шифрування даних полягає в наступному:

1. У передавачі:

– текст, що підлягає шифруванню, розбивається на однакові по довжині блоки;

– за допомогою спеціального генератора формується набір псевдовипадкових чисел, кожне з яких складається з відповідним блоком вихідного тексту;

– інформація блоками передається у канал зв’язку.

2. У приймачі з кожного отриманого блоку віднімається точно таке ж випадкове число, у результаті виходить вихідний текст.

Розроблена програма повинна:

1. Вивести на екран текст, що підлягає шифруванню.
2. Одержати з використанням процедури генератора випадкових чисел, зберегти й вивести на екран (у шістнадцятковому вигляді) послідовність, що шифрує (гаму).
3. Зашифрувати, зберегти й вивести на екран зашифрований рядок.
4. Розшифрувати рядок і вивести його на екран.

Як варіант для шифрування використається символьний рядок: прізвище плюс ім’я плюс по-батькові студента.

Приклад програми шифрування та розшифрування повідомлення

```
;kodo.asm

;виводить на екран вихідний рядок тексту, генерує 32-бітне

;випадкове число потрібну кількість разів, шифрує ним вихідний

;рядок тексту по чотири байти, виводить зашифрований рядок на

;екран, декодує рядок, виводить на екран

.model   tiny

.code

.386

org         100h                    ;початок COM-файла

start:

;виведення на екран вихідного рядка

mov       h,9                       ;номер функції DOS – в AH
```

```
mov     dx,offset mess1             ;адреса вихідного рядка – в DX

int     21h                         ;виклик функції DOS

mov     dx,offset crlf

mov     ah,9

int     21h                         ;переведення рядка одержання й

;виведення гамми

call rand                           ;виклик процедури в eax – перше

;випадкове число

mov     dword ptr gamma,eax         ;зберегти для наступних

;викликів

call rand                           ;виклик процедури в eax – друге
```

```
;випадкове число

mov dword ptr gamma+4,eax          ;зберегти для наступних

;викликів

mov     ah,9                       ;номер функції DOS – в AH

mov     dx,offset gamma      ;адреса гамми рядка – в DX

int     21h                        ;виклик функції DOS

mov     dx,offset crlf

mov     ah,9

int     21h                        ;переведення рядка

;процес шифрування

mov     eax, dword ptr gamma+4

mov     ebx, dword ptr mess1+4
```

```
add     eax,ebx

mov     dword ptr code+4,eax

mov     eax, dword ptr gamma

mov     ebx, dword ptr mess1

adc     eax, ebx

mov     dword ptr code, eax

;виведення на екран зашифрованого рядка

mov     ah,9                    ;номер функції DOS – в AH

mov     dx,offset code                  ;адреса шифрованого рядка –

;в DX

int     21h                     ;виклик функції DOS
```

```
mov     dx,offset crlf

mov     ah,9

int     21h                     ;переведення рядка

;процес розшифрування

mov     eax, dword ptr gamma+4

mov     ebx, dword ptr code+4

sub     ebx,eax

mov     dword ptr code+4,ebx

mov     eax, dword ptr gamma

mov     ebx,dword ptr code

sbb     ebx,eax

mov     dword ptr code,ebx
```

```
;виведення на екран розшифрованого рядка

mov     ah,9                    ;номер функції DOS – в AH

mov     dx,offset code                  ;адреса шифрованого рядка –

;в DX

int     21h                     ;виклик функції DOS

mov     dx,offset crlf

mov     ah,9

int     21h                     ;переведення рядка

ret

mess1           db      'ELIZAROV$'

gamma           db      "$"
```

```
code            db      "$"

crlf            db      0Dh,0Ah,'$'

;процедура повертає в eax випадкове 32-бітне число (від 0 до 231-2)

rand    proc    near

push    edx

mov     eax, dword ptr seed ;зчитати останнє випадкове

;число

test    eax, eax                ;перевірити його, якщо це -1

js fetch_seed                   ;функція ще жодного разу не

;викликалася й треба створити

;початкове значення

randomize:
```

```
mul dword ptr rand_a                    ;помножити на число a

div     dword ptr rand_m    ;взяти залишок від ділення

;на 231-1

mov     eax, edx

mov     dword ptr seed,eax  ;зберегти для наступних

;викликів

pop     edx

ret

fetch_seed:

push    ds

push    0040h
```

```
pop       ds

mov       eax,dword ptr ds:006Ch              ;зчитати подвійне

;слово з області

pop       ds                          ;даних BIOS за адресою

;0040 006C – поточне число

jmp short randomize                   ;тактів таймера

rand_a  dd       69621

rand_m dd       7FFFFFFFh

seed      dd       -1

rand    endp

end       start
```

Контрольні питання

1. Вкажіть типи генераторів випадкових чисел.
2. Опишіть алгоритм генерації випадкових чисел.
3. Опишіть алгоритм виводу на екран початкового рядка.
4. Опишіть алгоритм виводу на екран розшифрованого рядка.
5. Опишіть алгоритм шифрування.

Лабораторна робота 26

АНАЛІЗ ЗАСОБІВ РОБОТИ З ФАЙЛАМИ

Мета

1. Ознайомити студентів зі способами роботи з файлами;
2. Підготовити студентів до самостійного використання засобів програмування мовою Асемблер.

Завдання

1. Створити локальну папку на робочому диску.
2. З робочого диска комп'ютера викладача скопіювати в створену папку архівний файл системи програмування, провести інсталяцію СП.
3. Відповідно до варіанта завдання, зазначеного викладачем, підготувати для налагодження програми для роботи з файлами.
4. Підготувати bat-файл для трансляції, редагування й виконання програми.
5. Налагодити програми у вигляді COM-файлів.
6. Скласти звіт з лабораторної роботи.

Порядок виконання лабораторної роботи

Основна функція DOS як операційної системи – організація доступу до дисків як до набору файлів і директорій.

Починаючи з версії MS DOS 2.0, разом з удосконаленням FAT (наприклад, появою вкладених директорій), з'явився набір UNIX-подібних функцій роботи з файлами, що використовують для опису файла тільки одне 16-бітне число, ідентифікатор файла або пристрою. Всі інші функції роботи з файлами використовують потім тільки це число.

Перші п'ять ідентифікаторів ініціалізуються системою в такий спосіб:

0: STDIN – стандартний пристрій введення (клавіатура);

1: STDOUT – стандартний пристрій виведення (екран);

2: STDERR – пристрій виведення повідомлень про помилки (екран);

3: AUX – послідовний порт (зазвичай COM1);

4: PRN – паралельний порт (зазвичай LPT1).

Отже, функції читання/запису (а також скидання буферів на диск) файлів можна застосовувати й до пристроїв.

Розглянемо перелік функцій для роботи з файлами:

Функція DOS 3Ch – створити файл, якщо файл існує, ця функція однаково відкриває його, привласнюючи йому нульову довжину. Щоб цього не відбулося, варто користуватися функцією 5Bh.

Функція DOS 3Dh – відкрити існуючий файл.

Функція DOS5Ah – створити й відкрити тимчасовий файл. Створює файл з унікальним ім'ям, що не є насправді тимчасовим. Такий файл варто спеціально видаляти, для чого його ім'я й записується в рядок в DS:DX. Для роботи з довгими іменами файлів використовуються додаткові функції, які викликаються також, як функція DOS 71h.

Функціям LFN 6Ch – створити або відкрити файл із довгим ім'ям. Якщо функції відкриття файлів повертають помилку «занадто багато відкритих файлів» (AX = = 4), варто збільшити кількість припустимих ідентифікаторів за допомогою функції 67h.

Функція DOS 67h – змінити максимальне число ідентифікаторів файлів.

Функція DOS 3Fh – читання з файла або пристрою. Якщо під час читання з файла кількість фактично зчитаних байтів в AX менше, ніж замовлене число в CX, то був досягнутий кінець файлу. Якщо потрібно зчитати (або записати) довільну ділянку файла, використовують функцію 42h (функція lseekв 3).

Функція DOS42k – перемістити показник читання/запису. Показник можна встановити за реальними межами файла: у негативне число, тоді наступна операція читання/запису викличе помилку; у позитивне число, більше довжини файла, тоді чергова операція запису збільшить розмір файла.

Функція DOS 40h – запис у файл або пристрій. Якщо при записі у файл указати CX = 0, він буде обрізаний за поточним значенням покажчика. Насправді відбувається запис у буфер DOS, дані з якого скидаються на диск під час закриття файла або, якщо їхня кількість перевищує розмір сектора диска. Для негайного очищення буфера можна використати функцію 68h (функція fflush у C).

Функція DOS 68h – скидання файлових буферів DOS на диск. Для критичних ділянок програм важливо використати більш ефективну функцію 0Dh.

Функція DOS 0Dh – скидання всіх файлових буферів на диск.

Функція DOS 3Eh – закрити файл. Якщо файл був відкритий для запису, всі файлові буфери скидаються на диск, установлюється час модифікації файла й записується його нова довжина.

Функція DOS 41h – видалення файла. Видалити файл можна тільки після того, як він буде закритий, інакше DOS продовжить виконання запису в не існуючий файл, що може привести до руйнування файлової системи.

Функція LFN 41h – видалення файлів з довгим ім'ям. Не дозволяє використовувати маски (символи * й ? в імені файла) для видалення відразу декількох файлів, хоча цього можна домогтися, викликаючи її через не задокументовану функцію 5D00h.

Функція DOS 4Eh – знайти перший файл. Виклик цієї функції заповнює даними область пам'яті DTA (область передачі даних), що починається за умовчанням зі зсуву 0080h від початку блоку даних PSP (при запуску COM- і EXE-програм сегменти DS й ES містять сегментну адресу початку PSP), але її можна перевизначити за допомогою функції 1Ah.

Функція DOS 1Ah – встановити область DTA. Після того як DTA заповнена даними, для продовження пошуку варто викликати функцію 4Fh, поки не буде повернута помилка.

Функція DOS 4Fh – знайти наступний файл. У випадку з довгими іменами файлів (LFN) застосовується набір із трьох під-функцій функції DOS 71h, які можна використати, тільки, якщо запущено IFSmg.

Функція LFN 4Eh – знайти перший файл із довгим ім’ям.

Функція LFN 4Fh – знайти наступний файл.

Приклад програми, яка використовує багато функцій роботи з фай-лами.

```
;fidoh.asm

;заміняють українську "Н" латинською "H" у всіх файлах з

;розширення .TXT у поточній директорії.

.model   tiny

.code

org        100h                    ;COM-файл

start:

mov      ah, 4Eh                         ;пошук першого файла

xor        cx,cx                   ;не системний, не директорія
```

```
mov     dx,offset filespek     ;маска для пошуку в DS:DX

file_open:

int     21h

jc      no_more_files                   ;якщо CF = 1 – файли скінчилися

mov     ax,3D02h                ;відкрити файл для читання й

;запису

mov     dx, 80h+1Eh             ;зсув DTA + зсув імені файла

int     21h                     ;від початку DTA.

jc      not_open                        ;якщо файл не відкривається –

;перейти до наступного

mov     bx, ax                  ;ідентифікатор файла в BX
```

```
mov     cx, 1                   ;зчитувати один байт

mov     dx, offset buffer       ;початок буфера в DX

read_next:

mov     ah, 3Fh                 ;читання файла

int     21h

jc      find_next               ;якщо помилка – перейти до

;наступного

dec     ax                      ;якщо AX = 0 – файл скінчився

js      find_next               ;перейти до наступного

cmp byte ptr buffer, 8Dh        ;якщо не зчитана російська "Н"

jne     read_next               ;зчитати наступний байт

mov     byte ptr buffer, 48h    ;інакше – записати в буфер
```

```
;латинську букву "H"

mov     ax, 4201h               ;перемістити покажчик файла від

;поточної

dec     cx                      ;позиції назад на 1

dec     cx                      ;CX = 0FFFFh

mov dx, cx                      ;DX = 0FFFFh

int     21h

mov     ah, 40h                         ;записати у файл

inc     cx

inc     cx                      ;один байт (CX = 1)

mov     dx, offset buffer               ;з буфера в DS:DX
```

```
int         21h

jmp short read_next             ;зчитати наступний байт

find_next:

mov         ah, 3Eh                         ;закрити попередній файл

int         21h

not_open:

mov         ah, 4Fh                         ;знайти наступний файл

mov         dx, 80h                         ;зсув DTA від початку PSP

jmp         short file_open

no_more_files:                              ;якщо файли скінчилися

ret                                 ;вийти із програми

filespec   db     "*.txt", 0        ;маска для пошуку
```

```
buffer label   byte              ;буфер для читання/запису –

end     start                    ;за кінцем програми
```

Контрольні питання

1. Розкрийте поняття «файла і типи файлів».
2. Які функції використовують для читання файла?
3. Які функції використовують для запису файла?
4. Опишіть процедуру створення і відкриття файла.
5. Опишіть процедуру читання файла і запису в файл.
6. Опишіть процедуру закриття і видалення файла.

Список літератури

1. *Артеменко Ю.Н.* MySQL. Справочник по языку / Ю.Н. Арте-менко. – М.: Диалектика , 2005. – 429 с.
2. *Архангельский А.Я.* Язык C++ в C++Builder / А.Я. Архан-гельский . – М.: Диалектика , 2008. – 944 с.
3. *Аткинсон Л.* MySQL. Библиотека профессионала / Л. Аткин-сон – М.: Диалектика , 2002. – 624 с.
4. *Зубков С.В.* Assembler. Для DOS, Windows и Unix / С.В. Зуб-ков. М.: ДМК, 1999. – 640 с.

5. *Культин Н.Б.* Самоучитель C++ Builder / Н.Б. Культин. – Спб.: БХВ – Петербург, 2004. – 464 с.

6. *Ульман Л.* SQL. Руководство по изучению языка / Л. Ульман. – Спб.: ДМК Пресс, 2004. – 352 с.

7. *Финогенов К.Г.* Использование языка Асемблера / К.Г. Фи-ногенов. – М.: Горячая линия. Телеком, 2004. – 438 с.

8. *Холингворт Д.* Borland C++ Builder 6. Руководство разработчика / Д. Холингворт, Б. Сворт. – М.: Издательский дом “Вильямс”, 2004. – 976 с.

9. *Швец В.А.* Справочная система по языку Ассемблера IBM PC / В.А. Швец – К.: Киев, 2002. – 548 с.