МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

С. С. ЖУКОВСЬКИЙ, Т. А. ВАКАЛЮК

Об’єктно-орієнтоване програмування мовою C++

навчально-методичний посібник
для студентів напряму 6.040302 Інформатика*

Житомир 2016

УДК 004.42+004.432.2(076.5)
ББК 73р
Ж86

Затверджено Вченою радою Житомирського державного університету імені Івана Франка протокол № 4 від 01.11.2016 р.

Рецензенти:

Шевчук Л.Д. – кандидат педагогічних наук, доцент, заступник завідувача кафедри математики, інформатики та методики навчання ДВНЗ «Переяслав-Хмельницький державний педагогічний університет імені Григорія Сковороди»;

Медведєва М.О. – кандидат педагогічних наук, доцент, доцент кафедри вищої математики та методики навчання математики Уманського державного педагогічного університету імені Павла Тичини;

Мінтій І.С. – кандидат педагогічних наук, доцент кафедри інформатики та прикладної математики Криворізького режавного педагогічного університету.

Ж86 **Жуковський С.С., Вакалюк Т.А.**
Об’єктно-орінтоване програмування мовою С++. Навчально-методичний посібник для студентів напряму 6.040302 Інформатика*. – Житомир: Вид-во ЖДУ, 2016. – 100 с.

Посібник призначений для використання студентами під керівництвом викладача на лекціях, практичних та лабораторних заняттях. Посібник містить лекційний курс та лабораторний практикум із об’єктно0орієнтованого програмування мовою С++. Викладений матеріал відповідає діючій програмі з програмування для спеціальності «Інформатика»

Для студентів фізико-математичних спеціальностей вищих педагогічних закладів, вчителів інформатики загальноосвітніх шкіл.

УДК 004.42+004.432.2(076.5)
ББК 73р

ЗМІСТ

ТЕОРЕТИЧНІ ОСНОВИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ МОВОЮ С++ 4

Структури 4
Робота з текстовими та двійковими файлами 7
Використання списків 16
Перевантаження операцій 27
Успадкування класів 31
Виключні ситуації 36
Клас String 40
Контейнери 45

ЛАБОРАТОРНИЙ ПРАКТИКУМ 51

Лабораторна робота №1. Розробка програм з використанням структур .. 51
Лабораторна робота №2. Розробка програм з використанням текстових та двійкових файлів 57
Лабораторна робота №3. Розробка програм з використанням списків 58
Лабораторна робота №4. Розробка програм з використанням класів 64
Лабораторна робота №5. Перевантаження операцій 70
Лабораторна робота №6. Множинне успадкування мовою С++ 71
Лабораторна робота №7. Виключні ситуації 78
Лабораторна робота №8. Клас String 82
Лабораторна робота №9. Контейнери 86

ТЕСТОВІ ЗАВДАННЯ 88

РЕКОМЕНДОВАНА ЛІТЕРАТУРА 97

ТЕОРЕТИЧНІ ОСНОВИ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ МОВОЮ С++

Структури

Мета: ознайомити студентів з основними поняттями структур.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до майбутньої професійної діяльності.

Наочність: схематичні зображення.

Запитання для самоаналізу та самоперевірки:

1. Який синтаксис опису структури?
2. Як розподіляється пам’ять у структурі?
3. Як можна забезпечити доступ до елемента структури?
4. Як можна забезпечити доступ до елемента структури через вказівник?
5. Як описати змінну структурованого типу?
6. Як описати в структурі змінну структурованого типу?
7. Що таке вкладені структури?
8. Як визначити розмір структури?
9. Що таке покажчики на структуру?
10. Що таке масив структур?
11. Що таке бітові поля?
12. Що таке об’єднання?

Текст лекції.

Структура – це спеціальний тип даних створений програмістом, що складається з декількох відомих типів даних, що називаються полями.

```
struct <назва структури>
{
<тип> <назва поля>;
…
<тип> <назва поля>;
}<назва змінних і вказівників>;
```

Опис структури закінчується символом «;»

```
struct Student
{
char Name[20];
char SurName[20];
int Year;
};   // крапка з комою обов’язкова.
```

Розмір структури дорівнює сумі розмірів її полів. В даному прикладі 20+20+4=24 байти.

Для визначення обсягу структури можна використати функцію

sizeof(<назва типу структури або змінна>).

Оголошення змінної або вказівника структурованого типу даних аналогічне до оголошення звичайної змінної.

```
Student a, gr, *t1;
```

Оголосити змінні і структуру можна так:

```
struct Student
{
char Name[20];
char SurName[20];
int Year;
} a,b,*t; // оголошення змінних
```

В мові С можна оголосити змінні не називаючи структури:

```
struct     //відсутня назва
{
char Name[20];
char SurName[20];
int Year;
} a,b,*t;
```

Структура може містити поля типу структура:

```
struct Student
{
char Name[20];
char SurName[20];
struct {
int year,month, day;
}birthday;
} a,b,*t;
```

або

```
struct  BIRTHDAY {
int year,month, day;
};

struct Student
{
char Name[20];
char SurName[20];
BIRTHDAY birthday;
} a,b,*t;
```

Доступ до поля змінної типу структури відбувається за допомогою крапки.

Strcpy(a.Name,”Ivan”);

Доступ до поля вказівника типу структури відбувається за допомогою «->».

t->birthday.year=1991;

Бітові поля

Під час оголошення структури можна задавати обсяг пам’яті, який займають змінні певного поля. Для цього після назви поля ставиться знак двокрапка та зазначається ціле число, або константа – кількість бітів. Такі поля називаються бітовими.

Приклад демонструє використання бітових полів.

```
#include<iostream.h>
struct aa
{
short a:3;
short b:6;
};

int main()
{
   cout<<sizeof(aa);
   aa x;
   x.a=0;
   x.b=0;
   cout<<endl<<x.a<<' '<<x.b<<endl;
  for(int i=1;i<=100;i++)
  {x.a++; x.b++;
  cout<<i<<' '<<x.a<<' '<<x.b<<endl;
  }
   system("pause");
return 0;
}
```

Об'єднання — це місце в пам'яті, яке використовується для зберігання змінних, різних типів. Об'єднання дає можливість інтерпретувати один і той же набір бітів не менше, чим двома різними способами. Оголошення об'єднання (розпочинається з ключового слова union) схоже на оголошення структури і в загальному вигляді виглядає так:

```
union тег {
 тип ім'я-члена;
 тип ім'я-члена;
```

```
  тип ім'я-члена;
  …
} змінні-цього-об'єднання;
```

Наприклад:

```
union u_type {
  int i;
  char ch;
};
```

Об'єднання часто використовуються тоді, коли треба виконати специфічне перетворення типів, тому що дані, що зберігаються в об'єднаннях, можна означати абсолютно різними способами. Наприклад, використовуючи об'єднання, можна маніпулювати байтами, що становлять значення типу double, і робити так, щоб міняти його точність або виконувати яке-небудь незвичайне округлення.

Бітові поля.

Мова С має можливість працювати з окремими бітами. Це дає можливість. Ця можливість корисна, для збереження інформації, якщо обмежена пам’ять, деякі пристрої передають інформацію закодувавши її в один байт.

Метод використання бітових полів для доступу до бітів оснований на структурах. Бітове поле – це особий тип структури, що визначає яку довжину має кожен елемент.

Стандартний вигляд оголошення бітових полів наступний:

struct ім’я структури {
тип ім’я1: довжина;
тип ім’я2: довжина;
...
тип ім’яN: довжина;
}

Робота з текстовими та двійковими файлами

Мета: ознайомити студентів з основними поняттями розробки та описання програм з використанням текстових та двійкових файлів.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до майбутньої професійної діяльності.

Наочність: схематичні зображення.

Запитання для самоаналізу та самоперевірки:

1. Що таке текстовий (бінарний) файл?
2. Як оголосити файлову змінну (stdio.h, fstream)?
3. Як відкрити файл для читання, запису, дозапису (stdio.h, fstream)?
4. Як записати дані у текстовий файл (stdio.h, fstream)?
5. Як зчитати дані з текстового файлу (stdio.h, fstream)?

6. Як записати дані у двійковий файл ?
7. Як зчитати дані з двійкового файлу ?
8. Як встановити позицію запису, зчитування даних з файлу (у файл)?
9. Як визначити позицію зчитування(запису)?
10. Як закрити текстовий, (бінарний) файл?
11. Як визначити кінець файлу?

Текст лекції.

Робота з файлами <stdio.h>:

Оголошення – ***FILE *file***;

Відкриття – ***FILE *fopen(char *name, char *mode);***

Перевірка досягнення кінця файлу – int feof(FILE *file);

Закриття файлу – ***int fclose(FILE *file);*** спеціальна структура, оголошена у файлі ***<stdio.h>***, яка використовується час роботи з файлами. Для работи з файлом потрібно оголосити змінну ***FILE *<ім’я>***.

Функція fopen використовується для відкриття файлу. Перший параметр задає файлу. Другий параметр mode задає тип доступу до файлу.

Mode	Дія
"r"	Відкриття для читання. Якщо файл не існує або не знайдено, функція fopen повертає признак про помилку (NULL).
"w"	Відкриття для запису. Якщо файліснує його зміст видаляється. Якщо файл не існує, він створюється.
"a"	Відкриття для добавлення. Якщо файл не існує то він створюється.
"r+"	Відкривається файл для читання та запису. Файл повинен існувати.
"w+"	Відкриття пустого файлу для читання та запису. Якщо файл існує його зміст видаляється.
"a+"	Відкриття файлу для читання та добавлення. Якщо файл не існує файл створюється.
"rb"	Відкриття двійковий файлу без дозволу на модифікацію, файл відкривається лише для читання.
"wb"	Створення нового двійкового файлу тільки для запису, якщо файл із вказаним ім'ям вже існує, то він перезапишеться.

Текстовий режим

час введення/виведення в текстовому режимі відбувається перетворення зовнішнім представленням значення та внутрішнім (машинним) представленням цього значення.

Значення	Функція
Введення одного символу	int getc (FILE *file);
Виведення одного символу	int putc (int c, FILE *file);
Введення	int fscanf (FILE *file, char *format, ...);
Виведення	int fprintf(FILE *file, char *format, ...);
Введення рядка	char* fgets (char *line, intmaxline, FILE *file);
Виведення рядка	int fputs (char *line, FILE *file);

Приклад зчитування/запису одного елементу з/в двійковий файл.

```
#include<stdio.h>
#include<string.h>
#include<iostream>
using namespace std;
struct user
{
        char Name[20];
        int year;
};
int main()
{
        FILE *fb;
        fb = fopen("binfileStruct.bin","wb");
        user T;
        T.year=2000;      strcpy(T.Name,"Ivan");
        fwrite(&T,sizeof(user), 1,fb);
        fclose(fb);
     user B;
        T.year=0;
        fb = fopen("binfileStruct.bin","rb");
        fread(&B,sizeof(user), 1,fb);
       printf (“%s %d\n”, B.Name, B.year);
        fclose(fb);
       return 0;
}
```

Приклад зчитування/запису масиву елементів з/в двійковий файл.

```
#include<stdio.h>
#include<string.h>
#include<iostream>
using namespace std;
struct user
{
```

```
        char Name[20];
        int year;
};
int main()
{
        FILE *fb,*tf;
        user mas[10];
        int n;
   tf =fopen("text.txt","r");
   fscanf(tf,"%d",&n);
   for(int i=0;i<n;i++)
   fscanf(tf,"%s %d",&mas[i].Name,&mas[i].year);
        for(int i=0;i<n;i++)
     printf ("%s %d\n", mas[i].Name, mas[i].year);
        fb = fopen("binfileStruct.bin","wb");
        fwrite(mas,sizeof(user), n,fb);
        fclose(fb);
      user B[10];
        fb = fopen("binfileStruct.bin","rb");
        fread(B,sizeof(user), n,fb);
      cout<<"*******************************\n";
        for(int i=0;i<n;i++)
     cout<<B[i].Name<<' '<<B[i].year<<endl;
        fclose(fb);
    return 0;
}
```

Двійковий режим

У двійковому режимі перетворень не відбувається, внутрішнє представлення значення записується у файл. Для відкриття файлу в двійковому режимі необхідно в параметр mode функції fopen добавити символ b.

Введення з двійкового файлу

– ***unsigned fread (void *buf, intbytes, int num, FILE *file);***

Виведення у двійковий файл

– ***unsignedfwrite (void *buf, intbytes, int num, FILE *file);*** fread зчитує з файлу file в змінну buf num елементів, кожний розміром bytes байт. Функція fwrite записує у файл file buf num елементів, кожен розміром bytes байт. повертають кількість прочитаних/записаних елементів.

В двій ковому режимі можливий прямий домтуп до файлу:

int fseek(FILE *file, long nbytes, int origin)

Данна функція вказівник у файлі ***file*** на ***nbytes*** байт з позиції, що визначається параметром ***origin***. При цьому параметр origin може набувати наступних значень:

SEEK_SET 0 – початок файлу;
SEEK_CUR 1 – поточна позиція вказівника;
SEEK_END 2 – кінець файлу.

Функція ***long ftell (FILE *file)*** повертає поточну позицію вказівника в файлі file.

fseek(file, 0, SEEK_END);
n = ftell(file); Визначає розмір // приклад введення та запису

Файлові потоки (fstream.h)

Для використання файлових потоків необхідно підключити бібліотеку ***<fstream>***. Існує три різновиди файлових потоків: ***fstream***, ***ifstream*** і ***ofstream***. Різниця між ними полягає в тому, що потік ***fstream*** по замовчуванні відкривається для введення і виведення, потік по замовчуванні відкривається для введення, а потік ofstream по замовчуванні відкривається для виведення. Змінити поведінку по замовчуванні, а також задати інші режими відкриття файлу можна за допомогою наступних констант:

ios_base :: app	-відкриття файлу для додавання;
ios_base :: binary	-відкриття двійкового, а не текстового файлу;
ios_base :: in	-відкриття файлу для читання;
ios_base :: out	-відкриття файлу для запису;
ios_base :: trunc	-видалення вмісту файлу при відкритті.

Режими відкриття файлу комбінуються з допомогою операції порозрядного АБО (|).

Для відкриття файлу можна задати ім'я файлу безпосередньо в конструкторі потоку або скористатися функцією ***open***.

fstream fs ("f1.txt");	// Відкриття файлу для читання і запису
ifstream ifs ("f2.txt");	// Відкриття файлу для читання
ofstream ofs ("f3.txt");	// Відкриття файлу для запису
fstream fs ("f1.txt", ios_base :: in | ios_base :: out | ios_base :: trunk);	// Відкриття файлу для читання і запису з видаленням вмісту файлу
ifstream ifs ("f2.txt", ios_base :: in | ios_base :: binary);	// Відкриття двійкового файлу для читання
ofstream ofs;	// Створюємо потік, не пов'язаний з файлом
ofs.open ("f3.txt");	// Відкриваємо файл для запису

Якщо ви плануєте використовувати файл тільки для читання або тільки для запису безпечніше скористатися відповідним файловим потоком.

Для перевірки відкриття файлу існує функція ***is_open***.

Потоки автоматично закриваються при завершенні програми. Однак при необхідності можна закрити потік функцією ***close*** і потім знову відкрити його, зв'язавши з іншим файлом.

Для роботи з текстовими потоками використовуються операції << і >>. Також можливе використання маніпуляторів для форматування введення / виведення значень.

```
int x;
fstream f;
 f.open ("in.txt", ios_base :: in); // Відкриваємо файл для читання
if (! f.is_open ()) // Перевіряємо відкриття файлу
 {cout << "Неможливо відкрити файл 'in.txt' \ n"; return; }
f >> x; // Читання змінної x з файлу
if (f.fail ()) // Перевірка помилок читання
 {cout << "Помилка читання з файлу 'in.txt' \ n"; return; }
f.close (); // Закриваємо файл
f.open ("out.txt", ios_base :: out); // Знову відкриваємо файл, тепер для запису
if (! f.is_open ()) // Перевіряємо відкриття файлу
 {cout << "Неможливо відкрити файл 'out.txt' \ n"; return; }
f << hex << x << endl; // Виводимо значення змінної x в 16-річній системі
f.close (); // Закриваємо файл
```

Для того щоб перевірити, чи досягнуто кінець файлу, використовується функція eof.

```
int n;
ifstream f ("in.txt");

if (! f.is_open ())
 {Cout << "Неможливо відкрити файл 'in.txt' \ n"; return; }
while (! f.eof ()) // Поки не досягнуть кінець файла
 {f >> n; cout << n << endl; }
```

Для організації прямого доступу до файлу використовуються функції ***seekg / seekp itellg / tellp***. Відмінність між функціями полягає в тому, що функції, з ім'ям, що закінчується символом ***'g',*** використовуються для роботи з потоками введення, а функції, з ім'ям, що закінчується символом ***'p',*** - для роботи з потоками виводу.

Функції ***seekg / seekp*** переміщують внутрішній покажчик файлу на задану позицію. Позиції відповідають байтам, нумерація починається з ***0***. Існує два різновиди функцій - з одним параметром і з двома параметрами.

Один цілочисельний параметр задає абсолютну позицію у файлі. Два параметра задають зсув (ціле число) і точку відліку. Цей параметр може приймати наступні значення:

ios_base :: beg	- початок файлу;
ios_base :: cur	- поточна позиція покажчика;
ios_base :: end	- кінець файлу.

Функції **tellg / tellp** не мають параметрів. Вони повертають поточну позицію вказівника у файлі.

Функції **seekg / seekp і tellg / tellp** працюють як з текстовими, так і з двійковими потоками. У будь-якому випадку бажано або знати структуру файлу, або працювати з файлами, всі записи в яких мають однакову довжину. В іншому випадку можливе переміщення покажчика на позицію, яка не є початком запису.

Для роботи з двійковими файлами використовуються функції ***read*** і ***write.*** Як параметри функції отримують покажчик (типу ***char **** для функції ***read*** і типу const ***char **** для функції ***write***), який задає адресу початку масиву для введення / виведення, і ціле число, що задає кількість байт для введення / виведення.

Приклад 1. Введення масиву з текстового файлу

```
// Функція введення одновимірного масиву.
// Якщо введення був здійснений без помилок, повертається 1, в іншому випадку - 0.
// X - вводиться масив,
// N - покажчик на змінну, що містить кількість елементів масиву,
// Fname - ім'я файлу для введення.

int ArrayInput (double x [], int * n, char * fname)
 { int i;
   FILE * file;

   if ((file = fopen (fname, "r")) == NULL)
    {printf ("Неможливо відкрити файл '% s' \ n", fname);
     return 0;
     }
   if (fscanf (file, "% d", n) <1)
    {
     printf ("Помилка читання з файлу '% s' \ n", fname);
     fclose (file);
     return 0;
     }
   if (* n <0 || * n> NMAX)
```

```
 {printf ("Кількість ел-тів мас. Повинно бути від 1 до % d! (Файл
'% s') \ n", NMAX, fname);
  fclose (file);
  return 0;
  }
 for (i = 0; i <* n; i ++)
  if (fscanf (file, "% lf", & x [i]) <1)
   {printf ("Помилка читання з файлу '% s' \ n", fname);
    fclose (file);
    return 0;
    }
 fclose (file);
 return 1;
 }
```

Приклад 2. Виведення масиву в двійковий файл

```
// Виведення масиву в двійковий файл.
// Якщо висновок був здійснений без помилок, повертається 1, в
іншому випадку - 0.
// X - виведений масив,
// N - змінна, що містить кількість елементів масиву,
// Fname - ім'я файлу для виводу.

int BinOutput (double x [], int n, char * fname)
 {
 FILE * file;
  if ((file = fopen (fname, "wb")) == NULL)
   {printf ("Неможливо відкрити файл '% s' \ n", fname);
    return 0;
    }
  if (fwrite (x, n * sizeof (double), 1, file) <1)
   {printf ("Помилка запису в файл '% s' \ n", fname);
    fclose (file);
    return 0;
    }
  fclose (file);
  return 1;
 }
```

Приклад 3. Введення масиву з двійкового файлу

```
// Введення масиву з двійкового файлу.
// Якщо введення був здійснений без помилок, повертається 1, в
іншому випадку - 0.
// X - вводиться масив,
// N - покажчик на змінну, що містить кількість елементів масиву,
// Fname - ім'я файлу для введення.
```

```
int BinInput (double x [], int * n, char * fname)
 {FILE * file;

  if ((file = fopen (fname, "rb")) == NULL)
   {printf ("Неможливо відкрити файл '% s' \ n", fname);
    return 0;
   }
  for (* n = 0;; (* n) ++)
   if (fread (x + (* n), sizeof (double), 1, file) <1)
    if (feof (file))
     break;
    else
     {
      printf ("Помилка читання з файлу '% s' \ n", fname);
       fclose (file);
       return 0;
      }
  fclose (file);
  return 1;
 }

 // Інший спосіб
int BinInput (double x [], int * n, char * fname)
 {FILE * file;

  if ((file = fopen (fname, "rb")) == NULL)
   {printf ("Неможливо відкрити файл '% s' \ n", fname);
    return 0;
   }
  fseek (file, 0, SEEK_END);
  * N = ftell (file) / sizeof (double);
  fseek (file, 0, SEEK_SET);
  if (fread (x, sizeof (double), * n, file) <* n)
   {printf ("Помилка читання з файлу '% s' \ n", fname);
    fclose (file);
    return 0;
   }
  fclose (file);
  return 1;
 }
```

Приклад 4. Потоковий ввод масиву із двійкового файлу

// Введення масиву з двійкового файлу.

// Якщо введення був здійснений без помилок, повертається 1, в іншому випадку - 0.

// X - вводиться масив,

```
// N - покажчик на змінну, що містить кількість елементів масиву,
// Fname - ім'я файлу для введення.

int BinInput (double x [], int * n, char * fname)
 {
  ifstream f (fname, ios_base :: in | ios_base :: binary);
  if (! f.is_open ())
   {
     cout << "Неможливо відкрити файл '" << fname << "' \ n";
     return 0;
    }
  f.seekg (0, ios_base :: end);
  * n = f.tellg () / sizeof (double);
  f.seekg (0, ios_base :: beg);
  f.read (reinterpret_cast <char *> (x), * n * sizeof (double));
  if (f.fail ())
   {cout << "Помилка читання з файлу '" << fname << "' \ n";
    return 0;
    }
  return 1;
 }
```

Використання списків

Мета: Набути уміння та навички розробки та описання програм з використанням списків, текстових та двійкових файлів.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до майбутньої професійної діяльності.

Наочність: схематичні зображення.

Запитання для самоаналізу та самоперевірки:

1. Який синтаксис опису структури?
1. Як розподіляється пам’ять у структурі?
2. Як можна забезпечити доступ до елемента структури?
3. Що таке список?
4. Як реалізувати список мовою Сі?
5. Як добавити елемент в кінець списку, в середину списку, на початок списку?
6. Що таке стек?
7. Як реалізувати стек мовою Сі?
8. Що таке черга?
9. Як реалізувати чергу мовою Сі?
10. Яка перевага списку над масивом?
11. Що таке файл?
12. З якими файлами можна працювати в С++?

13. Як відкрити текстові файли для читання, запису?
14. Як відкрити двійкові (бінарні) файли для читання, запису?
15. Які переваги двійкових файлів перед текстовими і навпаки?
16. Як закрити файл?
17. Як звернутись до певного елемента бінарного файла?
18. Як перейти на початок бінарного файлу?
19. Як перейти на кінець бінарного файлу?

Текст лекції.

Приклад виконання лабораторної роботи

```
#include<fstream>
#include<iostream>
#include<string.h>
#include<math.h>
using namespace std;
// створення структури
struct abonent
{
  char name[20];
  char init[20];
  int nomer;
  char adress[20];
  abonent *next;
};
// опис вказівників на змінні
abonent *element, *pershij, *poper, *novij, **start, **last, *k;

int i=0, b;
// опис функцій
void StvSpisok();// створення списку
void output(); // виведення списку на екран
void output_1(); // запис списку у текстовий файл
void vstavka();    // вставка елементу в список
void output_bin_file();  // запис елементів списку у бінарний файл
void input_bin_file();   // зчитування з бінарного файлу

int main()
{
    StvSpisok();
    output();
    output_1();
    vstavka();
    output();
    output_bin_file();
```

```
    input_bin_file();
    system("pause");
    return 0;
}

void  StvSpisok()//--------створення списку введення даних з текстового
файлу----
  {
    ifstream  inf("Phone.dat");
    element = new (abonent);
    pershij = element;
    int i=0;
   do
     {
     poper=element;
     inf>>element->name;
     inf>>element->init;
     inf>>element->nomer;
     inf>>element->adress;
     element->next = new (abonent);
     element = element->next;
     }
    while(poper->nomer!=0);
    poper->next=NULL;  // закриваємо список
    inf.close();
     }

void output()//-----------виведення списку на екран------------------------------
{
   element = pershij;
   cout<<"SPISOK ABONENTIV\n";

    while(element->next!=NULL)

    {
      cout<<"**************\n";
      cout << element->name<<endl;
      cout << element->init<<endl;
      cout<< element->nomer<<endl;
      cout << element->adress<<endl;
      element = element->next;
    }
  cout<<"_____________________________________________________"<<endl;
 system("pause");
  }
```

```
void output_1()//----------запис списку у текстовий файл-------------------------
-
{
    ofstream outf("output_3.txt");
    element=pershij;
    cout<<"out abonent in output_3.txt \n";
    outf<<"abonentu:\n";
    int i=0;
    while(element->next!=NULL) // поки не кінець списку
    {
      outf<<element->name<<endl;
      outf<<element->init<<endl;
      outf<<element->nomer<<endl;
      outf<<element->adress<<endl;
      element=element->next;
     //  cout<<i<<endl;i++;
    }
    system("pause");
}

void vstavka()//---------------вставка елемента в список----------------------
{
 novij=new(abonent);
 cout<<"novij abonent\n";// введення нового елементу з клавіатури
     cin>>novij->name;
     cin>>novij->init;
     cin>>novij->nomer;
     cin>>novij->adress;
     // вставка на початок списку
     element=pershij;
     poper=element;
     if(strcmp(pershij->name,novij->name)>=0)//
       { cout<<"Insert Pershij   \n";
        pershij=novij;
        novij->next=element;

       }
     else
 {    //Вставка в середину списка
  int k=0;
  bool f=true;
  while(element->next!=NULL)//
  {
    if(stricmp(element->name,novij->name)>=0)
```

```
       {cout<<"insert \n";
        poper->next=novij;
        novij->next=element;
         f=false;
         break;
       }
       poper=element;     //insert
       element=element->next;// insert

    }
    if(f) {// вставка в кінець списку
         cout<<poper->name<<endl;
          poper->next=novij; novij->next=element;
         cout<<"insrt end\n";
         }
}
 system("pause");
}

void output_bin_file()//---------------запис в бінарний файл-----------------
{
  FILE *f;
  f=fopen("Phone_2.bin","wb");
  element=pershij;
  cout<<sizeof(abonent)<<endl;
   while(element->next!=NULL)
  {
    if (element->nomer==332277)
    {
    fwrite(&element,sizeof(abonent),1,f);
    }
    element=element->next;
  }
  strcpy(element->name,"0");      strcpy(element->init,"0");
  element->nomer=0;               strcpy(element->adress,"0");
   fwrite(&element,sizeof(abonent),1,f);
  fclose(f);
}

void input_bin_file()//-----------запис у бінарний файл--------------------
{
  FILE *f;
  f=fopen("Phone_2.bin","rb");
  cout<<"read binary file\n";
  element=new abonent;
```

```
  pershij=element;
  do
  {
     fread(&element,sizeof(abonent),1,f);
     cout << element->name<<endl;
     cout << element->init<<endl;
     cout<< element->nomer<<endl;
     cout << element->adress<<endl;
    element->next=new abonent;
    poper=element;
    element=element->next;
  }
  while(poper->nomer!=0);
  fclose(f);
}
```

Примітка. Текстовий файл повинен закінчуватися нулями..

Поняття класу, констуктор , деструктор

Мета: Набути уміння та навички розробки та описання програм з використанням класів, методів класу, конструкторів, деструкторів.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до майбутньої професійної діяльності.

Наочність: схематичні зображення.

Запитання для самоаналізу та самоперевірки:

1. Що таке клас в С++?
2. Який синтаксис опису класу?
3. Що таке поля, методи класу?
4. Які є специфікатори доступу класу, їхнє призначення?
5. Як можна забезпечити доступ до елементів класу?
6. Що таке конструктор? Які правила створення та роботи конструктора ?
7. Що таке конструктор позамовчунні, конструктор копіювання, конструктор перетворення?
8. Що таке деструктор? Які правила створення та роботи деструктора ?
9. Які існують ініціалізації елементів у конструкторах?
10. Який порядок виклику конструкторів та деструкторів?

В основі об'єктно-орієнтованої мови програмування лежать два основних поняття: клас та об'єкт. Об'єкт – це базове поняття в ООП, це конкретна реалізація, екземпляр класу. Об'єкт складається з трьох частин: стан (змінні стану), методи (операції), ім'я об'єкта. Клас - це група даних і методів або функцій для роботи з цими даними, це шаблон. Об'єкти однаковими наборами змінних стану і методів, утворюють клас. Якщо об'єкти мають реалізацію з конкретного світу, то класи є абстракціями. Трохи більш складні об'єкти можуть взагалі не містити даних, а представляти процес і містити тільки функції, які реалізують цей процес. Для формування реального об'єкта необхідно мати шаблон, по прикладу якого і будується даний об'єкт.

Поняттю "об'єкт" зіставляють ряд доповнюючих один одного визначень:

· це сукупність змінних стану і пов'язаних з ними методів (операцій). Ці методи визначають як об'єкт взаємодіє з оточуючим його світом;

· це конкретна реалізація, екземпляр класу. У програмуванні відносини об'єкта і класу можна порівняти з описом змінної, де сама змінна (об'єкт) є екземпляром якого-небудь типу даних (класу);

· це відчутна реальність, що характеризується чітко визначеною поведінкою;

· особливий упізнаваний предмет, блок або сутність (реальний чи абстрактний), що має важливе функціональне призначення в даній предметній області.

Є й інша класифікація методів об'єкта, коли виділяють функції управління, реалізації, доступу та допоміжні функції. Індивідуальність об'єкта - це властивість об'єкта, що відрізняє цей об'єкт від усіх інших об'єктів. Об'єкти можуть перебувати в певних відносинах один до одного, відносини можуть бути ієрархічними. Основні ієрархічні відносини називають відносини використання та включення.

Основні властивості об'єктів:

Зазвичай вважається, що без інкапсуляції неможливо уявити собі об’єктно-орієнтоване програмування.

Інкапсуляція це здатність об'єктів скривати деякі способи обробки даних (методи) та самі дані від навколишнього цифрового середовища. Історія розвитку методологій програмування спонукувана боротьбою зі складністю розробки програмного забезпечення. Складність великих програмних систем, у створенні яких бере участь відразу велика кількість розробників, зменшується, якщо на верхньому рівні не видно деталей реалізації нижніх рівнів. Власне, процедурний підхід був першим кроком на цьому шляху. Під інкапсуляцією (encapsulation, що можна перекласти по-різному, але на потрібні асоціації добре наводить слово "обволікання") розуміється приховування інформації про внутрішній устрій об'єкта, при якому робота з об'єктом може вестися тільки через його загальнодоступний (public) інтерфейс. Або це механізм програмування, об'єднуючий разом код і дані, якими він маніпулює, виключаючи як втручання ззовні, так і неправильне використання даних. В об'єктно-орієнтованій мові програмівання дані і код можуть бути об'єднані в абсолютно автономний чорний ящик. Усередині такого ящика знаходяться всі необхідні дані і код. Коли код і дані зв'язуються разом подібним чином, створюється об'єкт. Іншими словами, об'єкт - це елемент, що підтримує інкапсуляцію.

Поліморфізм - також одна з трьох основних парадигм ООП. Якщо говорити коротко, поліморфізм - це здатність об'єкта використовувати методи похідного класу, який не існує на момент створення базового. Слово «поліморфізм» можна перекласти як «багато форм». Тобто це можливість використання одного і того ж імені операції або методу до об'єктів різних класів, при цьому дії, що здійснюються з об'єктами, можуть істотно різнитися. Тому можна сказати, що в одного і того ж слова багато форм.

І остання парадигма **спадкування** (inheritance) - стосується здатності мови дозволяти будувати нові визначення класів на основі визначень існуючих класів. По суті, спадкування дозволяє розширювати поведінку базового класу, наслідуючи основну функціональність у похідному підкласі.Спадкування являє собою процес, в ході якого один об'єкт набуває властивостей іншого об'єкта. Це дуже важливий процес, оскільки він забезпечує принцип ієрархічної класифікації. Якщо вдуматися, то велика частина знань піддається систематизації завдяки ієрархічної класифікації по низхідній.

В об'єктно-орієнтованому програмуванні, клас — це спеціальна конструкція, яка використовується для групування пов'язаних змінних та функцій. При цьому згідно з термінологією ООП глобальні змінні класу (члени-змінні) називаються полями даних (також властивостями або атрибутами), а члени-функції називаютьметодами класу. Створений та ініціалізований екземпляр класу називають об'єктом класу. На основі одного класу, можна створити безліч об'єктів, що відрізнятимуться один від одного своїм станом (значеннями полів).

Майже кожному члену класа можна встановити модифікатор доступу (за винятком статичних конструкторів та деяких інших речей). У більшості об'єктно-орієнтованих мов програмування підтримуються такі модифікатори доступу:

- ***private*** (закритий, внутрішній член класу) — звернення до члену допускаються лише з методів того класу, у якому цей член визначений. Будь-які спадкоємці класу вже не зможуть отримати доступ до цього члену. Спадкування за типом private забороняє доступ з дочірнього класу до всіх членів батьківського класу, включаючи навіть public-члени (C++);
- ***protected*** (захищений, внутрішній член ієрархії класів) — звернення до члена допускаються з методів того класу, у якому цей член визначений, а також з будь-яких методів його класів-спадкоємців. Спадкування за типом protected робить все public-члени батьківського класу protected-членами класу-спадкоємця (C++);
- ***public*** (відкритий член класу) — звернення до члена допускаються з будь-якого коду. Спадкування за типом public не міняє модифікаторів батьківського класу (C++);

Структура класу:

```
class ім’я класу{
    public:
        // елементи в цій секції доступні з будь-якої частини програми
       MyClass(); // конструктор
     ~MyClass(); // деструктор
    protected:
      // елементи в цій секції доступні з класу і його нащадків
    private:
     // елементи в цій секції доступні лише з класу;
    // це область доступу по замовчуванні
};
```

В класі під час створення полів, ми не можемо надати їм значень. Для цього створюють метод, конструктор, ім’я якого співпадає з іменем класу.

Конструктор – це метод класу, який призначений для ініціалізації полів класу (надання початкових значень). Вкликається конструктор під час створення об’єктукласу.

В класі може бути декілька конструктоів, які відрізняються параметрами.

Конструктор без параметрів – називать конструктор по замовчуванні.

Метод класу, який призначений для знищення об’єкту класу, називається ***деструктор.*** В більшості випадків деструктор створюють тоді, коли клас використовує динамічні поля. По зовершенні використання даного об’єкту потрібно очистити динамічні поля.

Властивості конструкторів тадеструкторів .

- Конструктори та деструкторзавжди оголошуємо в розділі public.
- Конструктор та деструкторне має типу, навіть void
- Ім’я конструктора співпадає з іменем класу;
- Ім’я деструктор співпадає з імене класу з приставкою ~ ;
- В класі можна створити декілька конструкторів, які відрізняються параметрами.
- В класі можна створити лише один деструктор

Приклад програми.
Файл main.cpp

```
#include <iostream>
#include <stdlib.h>
#include "vector.h"
using namespace std;
int main(int argc, char** argv) {
Vector a;
Vector b(5,7);
a.Out();
b.Out();
Vector c(b);
c.Out();

        return 0;
}
```

Файл vector.h

```
#ifndef VECTOR_H
#define VECTOR_H

class Vector
{
    int x; //  координата x
    int y; //  координата y
```

```
        public:
                Vector();
                Vector(int, int );
                Vector(Vector &);
        void SetX(int);
        void SetY(int);
        void Out();
        protected:
};

#endif
```

ФайлVector.cpp

```
#include "vector.h"
#include <iostream>
using namespace std;

Vector::Vector()
{
        x=0;
        y=0;
}

Vector::Vector(int X,int Y)
{
        x=X;
        y=Y;
}

Vector::Vector(Vector &V)
{
        x=V.x;
        y=V.y;
}

void Vector::SetX(int X)
{
        x=X;
}

void Vector::SetY(int Y)
{
        y=Y;
}
```

```
void Vector::Out()
{
        cout<<x<<' '<<y<<endl;
}
```

Перевантаження операцій

Мета: Набути уміння та навички розробки та описання програм з використанням перевантаженням операцій.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до майбутньої професійної діяльності.

Наочність: схематичні зображення.

Запитання для самоаналізу та самоперевірки:

1. Що таке перевантаження операцій?
2. Який синтаксис перевантаження?
3. Які операції неможна перевантажувати?
4. За допомогою якої операції виконується перевантаження операцій?
5. Коли необхідно перевантажувати операцію присвоєння?
6. Як передаються параметри під час перевантаження операції?

Текст лекції.

В С++ існує можливість доозначати дію стандартних операторів (додавання, віднімання, множення і т.п. для випадку, коли операндами є об’єкти класів, створених користувачем). Така процедура називається ***перевантаженням операторів***

Перевантаження операторів реалізується шляхом створення відповідної операторної функції.

Операторна функція може бути:

1. Зовнішньою (але, як правило, дружньою) до класу

2 Методом

Оголошення зовнішньої операторної функції

тип_результату ***operator*** (аргумент);

```
{
        //програмний код Оператори:
}
```

Оператори перевантаження: ***+, -, *, /, %, =, +=, -=, *=, /=, %=, ++, --, !, [], ==, !=***

Перевантаження операторів

1. При перевантаженні ***бінарних операторів*** зовнішніми операторними функціями останні мають два аргументи – перший та другий операнди.

2. При перевантаженні ***унарних операторів*** зовнішніми функціями останні мають один аргумент – операнд.
Виключення – оператори інкременту та декременту (можуть мати два аргументи при перевантаженні постфіксної форми).
3. При перевантаженні ***бінарних операторів*** методами класу останні мають один аргумент – другий операнд. Першим операндом є об’єкт, з якого викорикається метод.
4. При перевантаженні ***унарних операторів*** методами класу останні не мають аргументів (операндом є об’єкт, з якого викликається метод).
Виключення – оператори інкременту та декременту (можуть мати один аргумент при перевантаженні постфіксної форми).

Перевантаження оператора додавання зовнішньою функцією.

```
class MComp
{
  public:
  double Re, Im;
};
MComp operator + ( MComp x,  MComp y )
{
  MComp z;
   z.Re=x.Re+y.Re;
   z.Im=x.Im+y.Im;
   return z;
}
int main()
{
MComp a,b,c;
a.Re=1;  a.Im=2;
b.Re=2;  b.Im=3;
c=a+b;
cout<<"c.Re = "<<c.Re<<"\n";
cout<<"c.Im = "<<c.Im<<"\n";
        return 0;
 }
```

Перевантаження оператора додавання методом класу

```
class MComp{
private:
double Re, Im;
public:
MComp operator+(MComp y)
{
MComp z;
z.Re=Re+y.Re; z.Im=Im+y.Im;
```

```
return z;
}
};
int main()
{
MComp a,b,c;
a.Re=1; a.Im=2; b.Re=2; b.Im=3;
c=a+b;
cout<<"c.Re = "<<c.Re<<"\n";
cout<<"c.Im = "<<c.Im<<"\n";
return 0;
}
```

Перевантаження операторної функції

```
class MComp{
public:
double Re, Im;};
MComp operator+(MComp x, MComp y){
MComp z;
z.Re=x.Re+y.Re; z.Im=x.Im+y.Im;
return z;}
об’єкт + об’єкт
;}
MComp operator+(double x, MComp y){
MComp z; число + об’єкт
z.Re=x+y.Re; z.Im=y.Im;
return z;}
об MComp operator+(MComp y, double x){
MComp z;
z Re=x+y Re; z Im=y Im;
об’єкт + число
z.y.z.y.return z;}
```

Перевантаженння операції присвоєння.

```
class MComp{
public:
double Re;
double Im; Результат
MComp operator=(MComp x){
Re=x.Re+1;
Im=x.Im-1;
return *this;}
};
int main(){
MComp a,b;
```

```
a.Re=1;
a.Im=2;
b=a; b=a >>> b≠a
cout<<"b.Re = "<<b.Re<<"\n";
cout<<"b.Im = "<<b.Im<<"\n";
}
return 0;}
```

Індексування об’єктів.

```
const int n=10;
class RealNums{
public:
int p[n];
RealNums(){
int k;
for(k=0;k<n; k++)
p[k]=rand()%n;
}
int &operator[](int i){
return p[i%n];
}
RealNums operator+(RealNums obj){
int i;
RealNums tmp;
 for(i=0;i<n;i++)
tmp[i]=p[i]+obj[i];
return tmp;
}

void show(){
int i;
fir(i=0;i<n;i++)
printf(“%3d”,p[i]);
printf(“\n”);
};
int main()
{
RelNums a,b;
a.show();
b.show();
(a+b).show();
for(int i=0;i<n;i++)
a[i]=b[i]-a[i];
a.show();
```

```
return 0;
}


}
```

РЕЗЮМЕ

1 Операторна 1. функція – функція, формальною назвою якої є перевантажуваний оператор. Операторна функція може бути як зовнішньою, так і методом класу.

2. Аргументами операторної функції є операнди відповідного виразу. У випадку зовнішньої операторної функції для бінарних операторів аргументи два, для унарних операторів - один. Якщо використовується метод класу, бінарним операторам відповідає один аргумент, унарним – жодного. Унарні оператори інкременту та декременту в цьому відношенні мають особливості.

3. За умовчанням для операторів інкременту та декременту перевантажується префіксна форма. Для перевантаження постфіксної форми використовують додатковий числовий аргумент.

4. Операторні функції можуть перевантажуватись.

Успадкування класів

Мета: отримання практичних навиків при використання множинного успадкування мовою С++.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до майбутньої професійної діяльності.

Наочність: схематичні зображення.

Запитання для самоаналізу та самоперевірки:

1. Що таке успадкування класів?
2. Який синтаксис успадкування класів?
3. Принцип керування доступом елементів класу при успадкуванні?
4. Як успадковуються конструктори, деструктори.
5. Як викликаються конструктори, деструктори успадкованих класів?

Рекомендована література:

1. Т.А.Павловська С/С++ програмирование на языке високого уровня.(ст..200-200)
2. Харви Дейтел, Пол_Дейтел Как програмировать на С++ (ст..551-596)

Текст лекції.

Для створення класів з доданою функціональністю вводять успадкування. Клас-нащадок має поля і функції-члени базового класу, але не має права звертатися до приватних (private) полів і функцій базового класу. У цьому і полягає різниця між приватними і захищеними членами.

Клас-нащадок може додавати свої поля і функції або перевизначати функції базового класу.

За умовчанням, конструктор нащадка без параметрів викликає конструктор базового класу, а потім конструктори доданих елементів. Деструктор працює в зворотному порядку. Інші конструктори доводиться визначати кожного разу наново. На щастя, це можна зробити викликом конструктора базового класу.

```
class ArrayWithAdd : public Array {

ArrayWithAdd(int n) : Array(n) {}

ArrayWithAdd() : Array() {}

ArrayWithAdd(const Array& a) : Array(a) {}

void Add(const Array& a);

};
```

Нащадок — це біль ш ніж базовий клас, тому він може використовуватися скрізь, де використовується базовий клас, але не навпаки.

Успадкування буває публічним, захищеним і власним. При публічному спадкуванні, публічні і захищені члени базового класу зберігають свій статус, а до приватних не можуть звертатися навіть функції-члени нащадка. Захищене спадкування відрізняється тим, що при нім публічні члени базового класу є захищеними членами нащадка. При приватному успадкуванні, до жодного члена базового класу навіть функції-члени нащадка права звертатися не мають. Як правило, публічне спадкування зустрічається значно частіше за інші.

Клас може бути нащадком декількох класів. Це називається **множинним спадкуванням.** Такий клас володіє полями і функціями-членами всіх його предків. Наприклад, клас FlyingCat може бути нащадком класів Cat і FlyingAnimal.

```
class Cat {

        ...
void Purr();


...
};
class FlyingAnimal {

        ...

void Fly();

...
};
class FlyingCat : public Cat, public FlyingAnimal {

...
PurrAndFly() {Purr(); Fly();}

...
};
```

Існує два типи успадкування. Це одиночне і множинне. Суть одиночного полягає в тому, що дочірній клас наслідує дані і властивості тільки одного батьківського класу. При множинному успадкування кількість батьківських класів повинна бути не меншою за один. При множинному успадкуванні можуть виникати проблеми із тим, що у батьківських класів є методи або атрибути із однаковими назвами. При їх використанні слід явно вказувати від якого з батьківських класів використовується метод або атрибут. Зараз множинне успадкування вже майже не використовується і всі нові мови програмування відмовились від цього принципу. Із популярних мов програмування множинне успадкування підтримує лише С++.

Керування доступом при успадкуванні

Повернемося до синтаксису успадкування. Специфікатори доступу - public, private, protected при призначенні типу успадкування можуть пропускатися (як, до речі, і було у нашому першому прикладі зі службовцями та менеджерами), при цьому керуються наступними правилами:

- якщо визначається class, то по замовчуванню похідний клас

приймається як private;

• якщо доступ не вказаний в успадкуванні при описі struct, то по замовчуванню він приймається як public;

Наступна таблиця містить визначення рівня доступу в середині похідного класу. У першій колонці - специфікатор доступу, що визначає успадкування між класами, у двох подальших - рівень доступу у базовому та похідному класах:

Рівні доступу у базових та похідних класах при успадкуванні

Тип успадкування class A: [ ] class B	Доступ у базовому класі A	Доступ у похідному класі B
public	private public protected	недоступно public protected
private	private public protected	недоступно private private
protected	private public protected	недоступно protected protected

З цієї таблиці, видно, які можливості надає механізм успадкування. Так, при відкритому успадкуванні public загальнодоступні та захищені елементи-дані зберігають свої рівні доступу надалі, і лише private-елементи виявляються недоступними вниз по ієрархії. Слід дотримуватися наступних правил успадкування методів у похідному класі:

1. Оскільки конструктори не успадковуються, похідні класи повинні мати власні конструктори. Тут можуть бути дві ситуації:

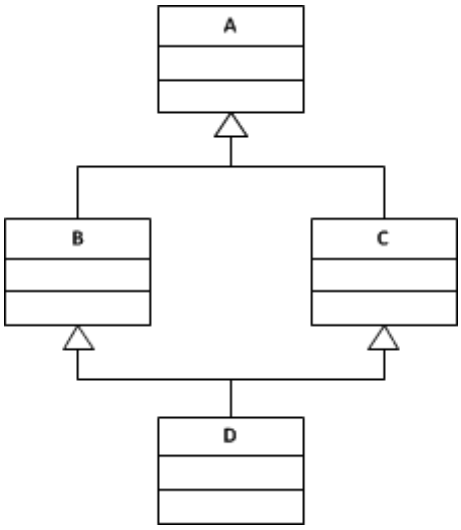
• якщо у конструкторі похідного класу відсутній явний виклик конструктора базового класу, автоматично викликається конструктор базового класу по замовчуванню (той, що не має параметрів). Для ієрархії декількох рівнів конструктори базових класів викликаються, починаючи з найвищого рівня.

• якщо конструктор базового класу потребує вказівку параметрів, він повинен бути явно викликаний в конструкторі похідного класу списком ініціалізації.

2. Оскільки деструктор не успадковуються та програмою не визначений деструктор у похідному класі, його буде згенеровано по замовчуванню і через нього викликано деструктори усіх базових класів. У класовій ієрархії

деструктори викликаються у порядку, зворотному до виклику конструкторів; спочатку деструктор поточного класу, а потім деструктор базового класу.

3. Похідний клас може перевизначати метод з одним і тим же ім'ям, що і у базовому класі, відповідно коректуючи його поведінку для себе. Аби запобігти неоднозначностям, рекомендовано перевизначати лише віртуальні методи класів.



Ситуація, зображена на малюнку можлива тільки для мови С++, оскільки на ній вказаний прояв множинного успадкування. Для того, щоб не виникало проблем, необхідно описувати успадкування класів B і C, як віртуальне від класу A. Тоді при створення об’єктів класу D не буде дублюватися клас A.

Виключні ситуації

Мета: Набути уміння та навички розробки та описання програм з використанням обробки виняткових(виключних) ситуацій.

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до майбутньої професійної діяльності.

Наочність: схематичні зображення.

Запитання для самоаналізу та самоперевірки:

1. Що таке виключні ситуації.
2. Синтаксис виключних ситуацій.
3. Як контролюються виключні ситуації
4. Як відбувається перехват виключних ситуацій

Текст лекції.

Одним з найбільш яскравих утілень принципу об'єктно-орієнтованого програмування є механізм обробки виняткових ситуацій у мові С++. У ході виконання програми можуть виявитися різні помилки. Вони можуть бути позв'язані з неправильним програмуванням (наприклад, вихід індексу масиву за межі припустимого чи переповнення пам'яті), а іноді їхня причина не залежить від програміста (скажемо, розрив зв'язку при мережевому з'єднанні). У кожній з цих ситуацій реакція програми непередбачена. Іноді вона завершує виконання, і лише після закінчення деякого інтервалу часу починають позначатися наслідку помилки, а частіше програма негайно припиняє роботу, піддаючи ризику дані, що знаходяться в пам'яті чи у файлі. Якщо не передбачити акуратне завершення роботи, використовуючи обробку виняткових ситуацій, результати можуть виявитися неприємними. В подальшому ми будемо називати винятковою ситуацією будь-яку подію, що вимагає особливої обробки. При цьому зовсім неважливо, чи є ця подія фатальною чи простою помилкою. Перевірка умов, що описують виняткову ситуацію, і реакція на її виникнення називається обробкою виняткової ситуації. Ця задача покладається на оброблювача виняткової ситуації.

Механізм обробки виняткових ситуацій

Обробка виняткових ситуацій у мові С++ є об’єктно-орієнтованою. Це значить, що виняткова ситуація є об'єктом, що генерується при виникненні незвичайних умов, передбачених програмістом, і передається оброблювачу, що неї перехоплює. Об'єктом, що описує природу виняткової ситуації, може бути будь-як сутність — літерал, рядок, об'єкт класу, число і т.д. Не слід думати, що виняткова ситуація обов'язково повинна бути об'єктом якого-небудь класу.

В основі обробки виняткових ситуацій у мові С++ лежать три ключових слова: try, catch і throw. Якщо програміст підозрює, що

визначений фрагмент програми може спровокувати помилку, він повинний занурити цю частину коду в блок try. Необхідно мати на увазі, що зміст помилки (за винятком стандартних ситуацій) визначає сам програміст. Це значить, що програміст може задати будь-яку умову, що приведе до створення виняткової ситуації. Після цього необхідно вказати, у яких умовах варто генерувати виняткову ситуацію. Для цієї мети призначене ключове слово throw. І нарешті, виняткову ситуацію потрібно перехопити й обробити в блоці catch. Ось як виглядає ця конструкція.

```
try {
 // Тіло блоку
try if(умова)throw виняткова_ситуація
}
catch(тип1 аргумент)
{
// Тіло блоку catch
}
 catch(тип2 аргумент)
{
// Тіло блоку catch
}
. . .
catch(типN аргумент)
 {
// Тіло блоку catch
}
```

Розмір блоку try не обмежений. У нього можна занурити як один оператор, так і цілу програму. Один блок try можна зв'язати з довільною кількістю блоків ***catch***. Оскільки кожен блок catch відповідає окремому типу виняткової ситуації, програма сама визначить, який з них виконати. У цьому випадку інші блоки catch не виконуються. Кожен блок catch має аргумент, що приймає визначене значення. Цей аргумент може бути об'єктом будь-якого типу. Якщо програма виконана правильно й у блоці try не виникло жодної виняткової ситуації, усі блоки ***catch*** будуть зігноровані. Якщо в програмі виникла подія, що програміст вважає небажаним, оператор ***throw*** генерує виняткову ситуацію. Для цього оператор throw повинний знаходитися усередині блоку ***try*** або усередині функції, викликуваної усередині блоку ***try***. Якщо в програмі виникла виняткова ситуація, для якої не передбачені перехоплення й обробка, викликається стандартна функція ***terminate()***, що, у свою чергу, викликає функцію ***abort().*** Утім, іноді виняткова ситуація не є небезпечної. У цьому випадку можна виправити помилку (наприклад, привласнити нульовому знаменнику ненульове значення) і продовжити виконання програми. Розглянемо найпростіший приклад

Обробка виняткової ситуації

```
#include using namespace std;
int main()
{
   int n = 10, m = 0;
printf("Початок\n");
try {
 printf("У блоці try\n");
  if(m==0) throw "Divide by zero";
  else n=n/m;
printf("Подальша частина блоку не виконується!");
}
 catch (const char* s)
 {
    printf("%s\n",s);
  }
 printf("Кінець\n"); return 0;
}
```

Ця програма виводить на екран наступні рядки. Начало Усередині блоку try Розподіл на нуль Кінець Простежимо за потоком керування при виконанні цієї програми. Спочатку з'являються і ініціалізуються дві целочисельні перемінні (одна з них дорівнює нулю). Потім виводиться повідомлення про початок виконання програми, і потік керування входить у блок ***try.*** Після виводу рядка повідомлення про вхід у блок try, потік керування переходить до перевірки рівності ***m==0***. Оскільки ця рівність є істиною, генерується виняткова ситуація (у даному випадку — константний рядок). Керування негайно передається блоку ***catch,*** аргументом якого є константний символьний вказівник, ігноруючи всі інші оператори в блоці ***try***. У цій програмі блок catch не робить жодних спроб виправити помилку. Замість цього він просто видає повідомлення — рядок, отриманий як аргумент — і передає керування оператору, що слідує за блоком. На закінчення функція printf() виводить на екран рядок Кінець, і програма завершує свою роботу. Тип виняткової ситуації повинний збігатися з типом аргументу розділу catch. Поглянемо, что відбудеться, якщо цією умовою зневажити. Порушення угоди про тип виняткової ситуації

```
#include <iostream>
 using namespace std;
int main()
{
int n = 10, m = 0;
printf("Начало\n");
try
```

{ printf("У блоці try\n"); if(m==0) throw "Розподіл на нуль"; else n=n/m; printf("Подальша частина блоку не виконується!"); } catch (const char s) //

Помилка! Необходимо

const char* s! { printf("%s\n",s); } printf("Кінець\n"); return 0; }

У цій програмі ми зробили цілком “природну” помилку — забули поставити зірочку в оголошенні аргументу. Тепер блок catch очікує виняткову ситуацію, що представляє собою константний символ, а не вказівник. Ця помилка приводить до аварійного завершення роботи програми. Покажемо, що відбудеться, якщо виняткова ситуація генерується усередині функції, яка викликається в блоці try. Виняткова ситуація, згенерована усередині функції

```
 #include
using namespace std;
int Denominator(int);
int main()
{ int n = 10, m; printf("Початок\n");
```

try { printf("У блоці try\n"); m=Denominator(0); printf("Подальша частина блоку не виконується!"); } catch (const char* s) { printf("%s\n",s); } printf("Кінець\n"); return 0; } int Denominator(int i) { if(i==0)throw "Розподіл на нуль"; return i; } У цій програмі виняткова ситуація генерується у функції Denominator(), яка викликається в блоці try. Завдяки цьому результати роботи програми цілком збігаються з попередніми. Якщо блок try знаходиться усередині функції, обробка виняткової ситуації виконується при кожнім виклику. Розміщення блоку try усередині функції

```
#include <stdio.h>
using namespace std;
int Denominator(int);
int main()
{ int n = 10, m; printf("Початок\n");
m=Denominator(0);
n = Denominator(11);
printf("Кінець\n"); return 0;
}
int Denominator(int i)
{ printf("У функції Denominator\n");
try {
 printf("У блоці try\n");
if(i==0) throw("Розподіл на нуль!");
if(i>10) throw 10;
 printf("Подальша частина блоку не виконується!");
}
catch (const char* s)
{
```

```
printf("%s\n",s);
}
catch (int n)
{
 printf("Чисельник більше %d\n",n);
}
return i;
}
```

Клас String

Мета: Набути уміння та навички розробки та описання програм з використанням класу String..

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до майбутньої професійної діяльності.

Наочність: схематичні зображення.

Запитання для самоаналізу та самоперевірки:

1. Що таке клас string?
2. Які конструктори існують в класі string?
3. Які операції допустимі для об’єктів классу string?
4. Як реалізовуються методи обробки рядка, об’єкту string?
5. Які методи обробки рядка існують?
6. Які методи переводять з типу string в тип char?

Текст лекції.

Конструктори

Клас String має декілька конструкторів:

```
string();
            // створює пустий об’єкт класу
string(const char*);
             // створює об’єкт  ініціалізуючи його рядком.
```

Клас string має три операції присвоєння:

```
string& operator=(const string& str); //присвоєння string величини
string& operator=(const char* s); //присвоєння рядка типу char
string& operator=(char C); //присвоєння значення символу
string str1, str2=“ABCD”;
```

Приклад:

```
char s[]=“EFGH”,c=‘a’;
```

```
str1=str2;
str1=s;
str2=c;
```

Операції класу string:

= присвоєння	>= менше або рівно
+ додавання	> менше
== порівняння	[] індексація
!= нерівно	<< виведення
< менше	>> введення
> більше	+= додавання
<= більше або рівно	

Введення

getline(cin,s2); введення до кінця рядка
cin>>s; введення до пропуску

Методи класу string

size() - повертає розмір рядка;
int n=str.size();
length() - повертає розмір рядка;
int n=str. length();
insert(pos,str) – вставляє рядок str

s.at(1); Доступ до елемента рядка.

Присвоєння рядка

assign(str)
assign(string str,pos,n)
assign(char *str,n)

Добавлення до одного рядка іншого

append (str)
append (string str,pos,n)
append (char *str,n)

Вставка рядка.

insert(pos, str); // вставляє рядок str у рядоу що викликається, в позицію pos.
insert (pos, string str, pos,n)
// вставляє у рядок що викликається, частину рядка str, з позиції pos довжиною n символів

insert (pos, char *str, n);
insert(pos,str) –
вставляє рядок str починаючи з похиції pos у рядок, що викликається
string str(“ABCD”);
str.insert(3,”ZZ”);
// отриманий рядок буде “ABCZZD”

Видалення частини рядка

erase(pos, n) видалення n елементів починаючи з позиції n
erase(n) видалення N перших елементів

Заміна рядка або частини рядка
replase(pos,n,str);
замінює n елементів рядкапочинаючи з позиції pos, елементами рядка str.
string str(“abcd”);
str. replace(1,2,”123”);

replase(pos1, n1, str, string pos2, n2);
replase(pos1, n1, char *str, n2);

Повернення підрядка
substr(pos,n);
повертає підрядок починаючи з позиції pos довжиною n символів
string str1,str(“ABCD”);
str1 = str.substr(1,2); // повертає “BC”

swap(str);

c_str(char *s); переведення в массив char

Копіювання

copy(char *s, int len, int pos);
копіювання рядка s, довжиною len, починаючи з позиції pos.

Пошук підрядків

find(string str, int pos);
find(char c, int pos); -повертає позцію першого входження
string str1,str(“ABCD”);
int n = str.find(str,”BC”);

// повертає 1

rfind(string str, int pos);
rfind(char c, int pos);
Повертає позицію останнього входження
string str1,str(“ABCD”);
int n = str.find(”BC”,0);
// повертає 1

find_first_of(string str, int pos); // першого
find_last_of(string str, int pos); // останнього
Повертає позицію першого (останнього) входження будь-якого символу підрядка в рядку
string str1,str(“ABCD”);
int n = str. first_of(”VC”,0);
// повертає 2

swap(str); // обмін рядків
str1.swap(str2);
erase(pos,n);//видяляє з даного рядка n символів починаючи з pos.
clear(); **// очишчає рядок;**

Копіювання рядка типу string в масив типу char.
copy(char *s,n,pos);
копіює в символьний масив s n елементів починаючи з позиції pos.

find(str, pos); -пошук першого лівого підрядка str в діному рялку, повертає позицію початку знайденого підрядка

rfind(s,n); - пошук першого лівого символа в діному рялку, повертає позицію початку знайденого підрядка

rfind(str,n); - пошук першого правого підрядка str в діному рялку, повертає позицію початку знайденого підрядка

rfind(s,n); - пошук першого правого символа в діному рялку, повертає позицію початку знайденого підрядка

find_first_of(str, pos); пошук самого лівого входження будь-якого символу раяка в рядку, повертає позицію знайденого символу.

find_first_of(char, pos); пошук самого лівого входження будь-якого символу раяка в рядку, повертає позицію знайденого символу.

find_last_of(str, pos); пошук самого правого входження будь-якого символу раяка в рядку, повертає позицію знайденого символу.

find_last_of(char, pos); пошук самого правого входження будь-якого символу раяка в рядку, повертає позицію знайденого символу.

compare(str); - порівняння рядків
compare(pos, n, str); - порівняння рядків
compare(pos1, n1, str, pos2,n2x); - порівняння рядків +, - 0

bool empty(); істина якщо рядок пустий.

```
#include<iostream>
#include<cstring>
#include<string>
#include<windows.h>
using namespace std;
char* Rus( char* text);
int  main()
{
        int a;
 cout<<Rus("Введіть a")<<endl;
// Виведення російського тексту через //функцію Rus
 cin>>a;
 cout<<Rus("Значення a=")<<a<<endl;
string str1="ЖДУ",str2="імені Івана Франка",str3,str4;
char  c1[]="ЖДУ",c2[]="імені Івана Франка",c3[80],c4[80];
  str3=str1+" "+str2;
  //з'єднує декілька рядків
  strcpy(c3,str3.c_str());
  cout<<Rus(c3)<<endl;

  cout<<Rus("Довжина str1=")<<str1.size();
  cout<<endl<<Rus("Довжина str2=")<<str2.length();
  str3.insert(3," *********");// //str3.insert(p,str);вставляє в рядок
        // str3 рядок str починаючи з позиції p

  strcpy(c3,str3.c_str());
  cout<<endl<<Rus(c3)<<endl;
  str3.replace(3,5,"12345");// str3.replace(3,5,"12345");
                // Замінює n ел. починаючи з k
strcpy(c3,str3.c_str());
  cout<<endl<<Rus(c3)<<endl;
  str4=str3.substr(3,5);
//  str3.substr(p,n); повертає рядок //довжини  n починаючи з позиції p
```

```
  strcpy(c3,str4.c_str());
  cout<<endl<<Rus(c3)<<endl;
  int n=str3.find("i",0);//str3.find(str,pos);
//повертає першу ліву позицію підрядка
//str  в рядку str3 починаючи з позиції pos
  cout<<"left n= "<<n<<endl;
 n=str3.rfind("i",20);//str3.rfind(str,pos); //повертає першу праву позицію
                      // підрядка  str  в рядку str3 починаючи з позиції pos
  cout<<"right n= "<<n<<endl;
  str1.swap(str2);   // str1.swap(str2); міняє //місцями рядки str1 і str2
  strcpy(c3,str1.c_str());
  cout<<endl<<Rus(c3)<<endl;
  strcpy(c3,str2.c_str());
  cout<<endl<<Rus(c3)<<endl;
    str3.erase(3,10);   //str.erase(pos,n);видаляє з рядка str n //символів
починаючи з позиції pos
  strcpy(c3,str3.c_str());
  cout<<endl<<Rus(c3)<<endl;
 if(str1.compare(str2)>0)
 // str1.compare(str2) порівнює str1 і str2
 {
    cout<<endl<<"str1 >str2"<<endl;
 }
  else
  cout<<endl<<"str1<=str2"<<endl;
 return 0;
}
```

Контейнери

Мета: Набути уміння та навички розробки та описання програм з використанням контейнерів та бібліотеки <algorithm>

Професійна спрямованість: дана лекція є складовою частиною професійної підготовки вчителя інформатики до майбутньої професійної діяльності.

Наочність: схематичні зображення.

Запитання для самоаналізу та самоперевірки:

1. Що таке контейнер.
2. Які є контейнери. Їх призначення.
3. Для чого призначений вектор? Які методи він підтримує?
4. Для чого призначений черга? Які методи він підтримує?
5. Для чого призначений список? Які методи він підтримує?
6. В якому з контейнерів можна добавити, видалити елементи в кінець.

7. В якому з контейнерів можна добавити, видалити на початок?
8. В якому з контейнерів можна добавити, вставити елементи всередину.

Текст лекції.

Контейнери – це об’єкти, які містять в собі інші об’єкти.
Існує декілька видів контейнерів

Послідовні контейнери:
vector – динамічний масив;
list - список;
deque - двостороння черга;

Асоціативні контейнери:

map - словники
multimap – словники з дублікатами
set - множини
multiset – множини з дублікатами
bitset – мітові множини

Поля контейнерів

Поле	Значення
value_tupe	Тип елемента контейнера
size_tupe	Тип індексів, лічильників елементів
iterator	Ітератор
const_iterator	Константний ітератор
revers_iterator	Обернений ітератор
const_revers_iterator	Константний обернений ітератор
referense	Посилання на елемент
const_referense	Костантне посилання на елемент
key_tupe	Тип ключа(для асоціативних контейнерів)
key_compare	Тип критерію порвняння (для асоц. конт)

Ітератор – аналог вказівника на елемент.
Використовується для перегляду контейнера в прамому та в зворотньому напрямку.
Константні ітератори використовуються для елементів, які не міняються.

За допомогою ітераторів можна переглянути контейнери, незважаючи на типи даних, що використовуються для доступу до елементів.

Методи доступу до елементів

Метод	Значення
iterator begin()	Вказує на перший елемент
iterator end()	Вказує на елемент, що слідує за останнім
revers_iterator r_begin()	Вказує на перший елемент в оберненій послідовності
revers_iterator rend()	Вказує на елемент, наступним за останнім в оберненій послідовності

Методи визначення розміру контейнера

Метод	Значення
size()	Кількість елементів
max_size()	Максимальний розмір контейнера
empty()	Булевська функція, що показує, чи пустий контейнер

Порівняння контейнерів

Операція	**Метод**	**vector**	**deque**	**list**
Вставка в початок	**push_front**	-	+	+
Видалення з початку	**pop_front**	-	+	+
Вставка в кінець	**push_back**	+	+	+
Видалення з кінця	**pop_back**	+	+	+
Вставка в довільне місце	**insert**	(+)	(+)	+
Видалення з довільного місця	**Erase**	(+)	(+)	+
Довільний доступ до елементів	**[], at**	+	+	-

vector

Конструктори:
1) vector(); //конструктор по замовчуванню
2) vector(size_tupe n, const T& value=T());
//вектор довжиною n заповнюється
//одинавовим елементом value
3) vector(InputIter first, InputIterLast);
//вектор копіюється з іншого за ітераторами
//діапазону
4) vector(const vector <T>&x);
// конструктор копіювання

ч
vector <int> v1(10,1);
vector <int> v2(v1);
vector <int> v3(v2.begin(), vector.begin()+2);

Асоціативні контейнери

Асоціативні контейнери – забезпечують швидкий до даних за рахунок того, що вони побудовані на основі збалансованих деревах.

Існує 5 типів асоціативних контейнерів:

- словники (map) ;
- словники з дублікатами (multimap);
- множини (set);
- множини з дублікатами (multiset);
- бітові множини (bitset).

Словники часто називають асоціативними масивами або відображеннями.

Словник побудований на основі пари значень, перше з яких являє собою ключ для ідентифікації елемента, а другий – самий елемент.

Асоціативні контейнери описані у заготовочних файлах <map> і <set>.

Для збереження пари «ключ - елемент» використовується шаблон pair, описаний в заголовочному файлі <utility> . Шаблон pair має два параметри: first, second.

Для пари визначено дві операції порівняння = = рівно та < менше.

p1<p2 якщо p1.first<p2.first або p1.first==p2.first && p1.second<p2.second.

```
#include<iostream>
```

```
#include<utility>

using namespace std;

int main()
{
pair <int, double> p1(10,12.3),p2(p1);      //задається пара
p2=make_pair(20,12.3);                      //еквівалентно p2= pair<int, double> (20,12,3)
cout<<"p1 :"<<p1.first<<" "<<p1.second<<endl;
cout<<"p2 :"<<p2.first<<" "<<p2.second<<endl;
p2.first-=10;
if (p1==p2) cout<<"p1==p2\n";               //порівняння пари
p1.second-=1;
if (p1>p2) cout<<"p1>p2\n";
}
```

Словники (map)

У словнику (map) на відміну від словника з дублікатами (multimap), всі ключі повинні бути унікальні. Елементи в словнику зберігаються у відсортованому порядку

Шаблон словника містить три параметра: тип ключа, тип елемента і тип функціонального об’єкта

```
#include<fstream>
#include<iostream>
#include<string>
#include<map>
using namespace std;
typedef map <string,long,less <string> > map_sl;

int main()
{
map_sl m1;
ifstream in ("phonebook");
string str;
long num;
  while( in>>num,!in.eof()){
    in.get();
      getline(in,str);
      m1[str]=num;
      cout<<str<<"  " <<num<<endl;
 }
  m1["Petya K."]=2134522;
  map_sl::iterator i;
  cout<<"m1 :"<<endl;
```

```
  for(i=m1.begin(); i!=m1.end();i++)
       cout<<(*i).first<<" "<<(*i).second<<endl;
  i=m1.begin(); i++;
  cout<<"Drugiy element :";
  cout<<(*i).first<<"  "<<(*i).second<<endl;
  cout<<"Vasia :" << m1["Vasia"]<<endl;
 }
```

Для пошуку елементів у словнику визначені функції:

find() - повертає ітератор на знайдений елемент, у випадку його відсутності повертає end()

upper_bound(x) – повертає ітератор на перший елемент, ключ якого не менше x, у випадку його відсутності повертає end();

lower_bound(x)– повертає ітератор на перший елемент, ключ якого більший x, у випадку його відсутності повертає end().

Для вставлення та виділення ементів у словнику визначені функції:

insert() - вставлення елементів у словник;

erase() - видалення елементів зі словника;

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Лабораторна робота №1. Розробка програм з використанням структур

Мета: Набути уміння та навички розробки та описання програм обробки з використанням структур.

Програмне забезпечення: Середовище програмування DEV C++, Visual C++.

Контрольні питання.

1. Який синтаксис опису структури?
2. Як розподіляється пам’ять у структурі?
3. Як можна забезпечити доступ до елемента структури?
4. Як можна забезпечити доступ до елемента структури через вказівник?
5. Як описати змінну структурованого типу?
6. Як описати в структурі змінну структурованого типу?
7. Що таке вкладені структури?
8. Як визначити розмір структури?
9. Що таке покажчики на структуру?
10. Що таке масив структур?
11. Що таке бітові поля?
12. Що таке об’єднання?

Практична частина.

Варіант 1.

1. Описати структуру з ім’ям STUDENT, яка містить наступні поля:
   - Name – Прізвище та ініціали;
   - Year – рік народження:
   - Bal – оцінки з 4 предметів (масив з 4 елементів)
2. Написати програму, що використовує дану структуру і виконує наступні дії:
   - вводить з клавіатури масив даних Group, що складається з N змінних типу STUDENT;
   - Впорядковує записи за зростанням поля Year
   - Виводить на екран прізвища і рік народження студентів середній бал яких > 4.0;

Варіант 2.

1. Описати структуру з ім’ям SKLAD, яка містить наступні поля:
   - Name – Назва товару;

- Type – одиниця вимірювання:
- Quantity – кількість одиниць товару;
- Cost – ціна одиниці товару.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з клавіатури масив даних SHOP, що складається з N змінних типу SKLAD;
- Впорядковує записи по спаданню поля Name;
- Виводить на екран ціну та кількість товару, назва якого вводиться з клавіатури або виводить повідомлення про його відсутність.

Варіант 3.

1. Описати структуру з ім’ям TRAIN, яка містить наступні поля:

- Nazv – Назва
- Numer – номер поїзда;
- Date – дата відправлення (структура: day; month, year - день, місяць, рік);
- Time – ціна одиниці товару.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з клавіатури масив даних AVTOPARK що складається з N змінних типу TRAIN;
- Впорядковує записи по спаданню поля Numer;
- Виводить на екран всі рейси, які час відправлення яких після 15.00 по введеній даті.

Варіант 4.

1. Описати структуру з ім’ям ABONENT, яка містить наступні поля:

- Name – прізвище абонента;
- Init – ініціали абонента;
- Nomer – номер телефону;
- Adress –домашня адреса.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з клавіатури масив даних TELEFON, що складається з N змінних типу ABONENT;
- Впорядковує записи по зростанню поля Name;
- Виводить на екран прізвище, ініціали та домашню адресу за введеним номером телефону, або виводить повідомлення про його відсутність.

Варіант 5.

1. Описати структуру з ім’ям AEROFLOT, яка містить наступні поля:

- Nazv –назва пункту призначення;
- Numer – номер рейсу;

- Type –тип літака;
- Time –час відправлення.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з клавіатури масив даних ROZKLAD, що складається з N змінних типу AEROFLOT;
- Впорядковує записи по зростанню поля Type;
- Виводить на екран всі номери рейсів, та час відправлення літаків які відправляються в введений з клавіатури пункт призначення або повідомляє про відсутність таких рейсів.

Варіант 6.

1. Описати структуру з ім’ям DETAL, яка містить наступні поля:

- Name – назва деталі;
- Sort – сорт виробу;
- Date –дата виготовлення (структура: day; month, year - день, місяць, рік);
- Quant – кількість;
- Cost - ціна деталі.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з клавіатури масив даних ZAKAZ, що складається з N змінних типу DETAL;
- Впорядковує записи по спаданню поля Name;
- Виводить на екран всі деталі I сорту які виготовлені пізніше заданої дати, яка введена з клавіатури.

Варіант 7.

1. Описати структуру з ім’ям BOOK, яка містить наступні поля:

- Name – назва книги;
- Avtor – автори книги;
- Data –дата друку (структура: month, year - місяць, рік);
- Cost - ціна книги.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з клавіатури масив даних SHOP, що складається з N змінних типу BOOK;
- Впорядковує записи по зростанню поля Avtor;
- Виводить на екран всі книги які були надруковані в заданому році.

Варіант 8.

1. Описати структуру з ім’ям TOVAR, яка містить наступні поля:

Name – назва товару; Cost_Z – ціна закупки товару; Cost_P - ціна продажу товари.
Quantity –кількість одиниць товару; Pributok – прибуток.

2. Написати програму, що використовує дану структуру і виконує наступні дії:
   - вводить з клавіатури масив даних SHOP, що складається з N змінних типу TOVAR і обчислює прибуток по кожному товару;
   - Впорядковує записи по зростанню поля Pributok;
   - Виводить на екран всі товари, які мають найбільший прибуток.

Варіант 9.

1. Описати структуру з ім’ям VISTAVA, яка містить наступні поля:
   - Nazva – назва вистави;
   - Date – дата вистави (структура: day; month, year - день, місяць, рік);
   - Cost - ціна квитка.
   - Day_week – день неділі;

2. Написати програму, що використовує дану структуру і виконує наступні дії:
   - вводить з клавіатури масив даних THEATRE , що складається з N змінних типу VISTAVA;
   - Впорядковує записи по зростанню поля Day_week;
   - Виводить на екран всі вистави і дати іх проходження, які відбудуться в заданий день тижня.

Варіант 10.

1. Описати структуру з ім’ям WORKER, яка містить наступні поля:
   - Name – Ім’я працівника;
   - Surname – Прізвище працівника ;
   - Date – дата народження (структура: day; month, year - день, місяць, рік);
   - Pos - посада працівника.
   - Zarplata – заробітна плата працівника

2. Написати програму, що використовує дану структуру і виконує наступні дії:
   - вводить з клавіатури масив даних OFFICE, що складається з N змінних типу WORKER;
   - Впорядковує записи по віку працівників по спаданню;
   - Виводить на екран всіх працівників старших 30 років, які мають заробітну плату меншу введеної з клавіатури.

Варіант 11.

1. Описати структуру з ім’ям ZARPLATA, яка містить наступні поля:
   - Name – прізвище ім’я та по батькові працівника;
   - Date – дата народження (структура: day; month, year - день, місяць, рік);

- Work_day – кількість відпрацьованих днів;
- Stavka – ставка з урахуванням на 24 робочих дні;
- Naraxovano – нараховано заробітної плати з урахуванням ставки і кількості відпрацьованих днів;
- Do_viplati – сума виплати заробітної плати працівнику з вирахуванням 20% податку.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з клавіатури масив даних BUHGALTER, що складається з N змінних типу ZARPLATA і підраховує поля нараховано та до виплати ;
- Впорядковує записи по зростанню поля Name;
- Виводить на екран всіх працівників з їхньою заробітною платою, які відпрацювали менше 15 робочих днів.

Варіант 12.

1. Описати структуру з ім’ям INSTRUMENT, яка містить наступні поля:

- Name – назва музикального інструменту;
- Type – тип музикального інструменту;
- Year – рік виготовлення інструменту;
- Vlasnik – власник інструменту;
- Vartist – вартість музикального інструменту;

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з клавіатури масив даних ORKESTR, що складається з N змінних типу INSTRUMENT;
- Впорядковує записи по зростанню поля Vartist;
- Виводить на екран всі інструменти, вартість яких білша за 1000 грн..

Варіант 13.

1. Описати структуру з ім’ям COMPUTER, яка містить наступні поля:

- Procesor – процесор комп’ютера;
- Ram – обсяг оперативної пам’яті;
- HDD – структура що містить поля (Namt- виробник, V_Ram - обсяг, V- швидкість обертання диску);
- Monitor – ставка з урахуванням на 24 робочих дні;
- Keyboard – нараховано заробітної плати з урахуванням ставки і кількості відпрацьованих днів;
- Mouse – сума виплати заробітної плати працівнику з вирахуванням 20% податку.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з клавіатури масив даних CLASS, що складається з N змінних типу COMPUTER;

- Впорядковує записи по зростанню поля V_ram;
- Виводить на екран всіх комп’ютери за введеним процесором..

Варіант 14.

1. Описати структуру з ім’ям CUPURA, яка містить наступні поля:
   - Name – назва валюти;
   - Njminal – номінал;
   - Year – рік випуску куп’юри;
   - Kurs – курс валюти відносно української гривні;
2. Написати програму, що використовує дану структуру і виконує наступні дії:
   - вводить з клавіатури масив даних BANK, що складається з N змінних типу CUPURA;
   - Впорядковує записи по зростанню поля Name;
   - Виводить на екран всіх номінали вказаного року.

Лабораторна робота №2. Розробка програм з використанням текстових та двійкових файлів

Мета: Набути уміння та навички розробки та описання програм з використанням текстових та двійкових файлів.

Програмне забезпечення: Середовище програмування DEV C++, Visual C++ .

Контрольні питання.

1. Що таке текстовий (бінарний) файл?
2. Як оголосити файлову змінну (stdio.h, fstream)?
3. Як відкрити файл для читання, запису, дозапису (stdio.h, fstream)?
4. Як записати дані у текстовий файл (stdio.h, fstream)?
5. Як зчитати дані з текстового файлу (stdio.h, fstream)?
6. Як записати дані у двійковий файл ?
7. Як зчитати дані з двійкового файлу ?
8. Як істановити позицію запису, зчитування даних з файлу (у файл)?
9. Як визначити позицію зчитування(запису)?
10. Як закрити текстовий, (бінарний) файл?
11. Як визначити кінець файлу?

Завдання до лабораторної роботи.

Виконати лабораторну роботу №1, з використанням текстових та бінарних файлів.

1. Зчитати масив даних структуроваого типу відповідно варіанту з текстового файлу.
2. Відсортований масив вивести у текстовий файл з використанням бібліотеки stdio.h
4. Вивести масив структкри у бінарний файл.
5. Зчитати з бінарного файлу всі дані у масив структур.
6. Із зчитаного масиву виконати відбір відповідно варіанту і результат вивести у інший текстовий файл з використанням бібліотеки fstream.h

Лабораторна робота №3. Розробка програм з використанням списків

Мета: Набути уміння та навички розробки та описання програм з використанням списків, текстових та двійкових файлів.

Програмне забезпечення: DEV C++, Visual C++.

Контрольні питання.

1. Який синтаксис опису структури?
2. Як розподіляється пам’ять у структурі?
3. Як можна забезпечити доступ до елемента структури?
4. Що таке список?
5. Як реалізувати список мовою Сі?
6. Як добавити елемент в кінець списку, в середину списку, на початок списку?
7. Що таке стек?
8. Як реалізувати стек мовою Сі?
9. Що таке черга?
10. Як реалізувати чергу мовою Сі?
11. Яка перевага списку над масивом?
12. Що таке файл?
13. З якими файлами можна працювати в С++?
14. Як відкрити текстові файли для читання, запису?
15. Як відкрити двійкові (бінарні) файли для читання, запису?
16. Які переваги двійкових файлів перед текстовими і навпаки?
17. Як закрити файл?
18. Як звернутись до певного елемента бінарного файла?
19. Як перейти на початок бінарного файлу?
20. Як перейти на кінець бінарного файлу?

Зміст завдання

Варіант 1.

1. Описати структуру з ім’ям BOOK, яка містить наступні поля:

- Name – назва книги;
- Avtor – автори книги;
- Data –дата друку (структура: month, year - місяць, рік);
- Cost - ціна книги.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з текстового файлу SHOP.dat список книг, згідно структури, використовуючи елемент ***список***, і виводить його на екран;
- з даного списку видалити всі книги введеного з клавіатури року;
- добавити в кінець списку N нових книг;
- новостворений список записати в двійковий файл SHOP_NEW.DAT.

3. Написати програму яка зчитує з двійкового файлу SHOP_NEW.DAT, дані про 10 останніх книг. Записати у файл BOOK.DAT дані про книги відсортовані в алфавітному порядку по автору.

Варіант 2.

1. Описати структуру з ім’ям TOVAR, яка містить наступні поля:
- Name – назва товару;
- Cost_Z – ціна закупки товару;
- Cost_P - ціна продажу товари.
- Quantity –кількість одиниць товару;
- Pributok – прибуток.

2. Написати програму, що використовує дану структуру і виконує наступні дії:
- вводить з текстового файлу SKLAD.dat список книг, згідно структури, і виводить його на екран, використовуючи елемент ***список;***
- обчислює прибуток по кожному товару;
- добавляє на початок списку N нових товарів;
- видаляє товар, який не приносить прибуток;
- з новоствореного списку записати у двійковий файл SKLAD.sol всі товари, які мають найбільший прибуток.

3. Написати програму яка зчитує з двійкового файлу SKLAD_N.DAT, дані про всі товари, і записує в файл TOVAR.DAT товар, ціна якого перевищує середню ціну товару..

Варіант 3.

1. Описати структуру з ім’ям VISTAVA, яка містить наступні поля:
- Nazva – назва вистави;
- Date – дата вистави (структура: day; month, year - день, місяць, рік);
- Cost - ціна квитка.
- Day_week – день тижня;

2. Написати програму, що використовує дану структуру і виконує наступні дії:
- вводить з текстового файлу THEATRE.dat список вистав згідно структури VISTAVA (в файлі знаходиться список вистав відсортований по днях тижня);
- добавляє 5 нових вистав які вставляються в даний список зберігаючи відсортований список.
- Видаляє всі вистави, списку які заплановані у вівторок;
- З новоствореного списку записати у двійковий файл NEW.sol всі вистави, які відбудуться з 3-му місяці.

3. Написати програму яка зчитує з двійкового файлу NEW.sol, дані про товари, і записує в файл VISTAVA.DAT всі вистави, ціна яких нижча за середню ціну всіх вистав.

Варіант 4.

1. Описати структуру з ім’ям WORKER, яка містить наступні поля:
- Name – Ім’я працівника;
- Surname – Прізвище працівника ;
- Date – дата народження (структура: day; month, year - день, місяць, рік);
- Pos - посада працівника.
- Zarplata – заробітна плата працівника

2. Написати програму, що використовує дану структуру і виконує наступні дії:
- вводить з текстового файлу OFFICE.DAT, список працівників і дані про них згідно заданої структури (дані впорядковані по прізвищу);
- Добавляє з клавіатури нового працівника зберігаючи впорядкованість;
- З новоствореного списку записати у двійковий файл OFFICE.BIN працівників зарплата яких не перевищує 1000 грн.

1. Написати програму яка зчитує з файлу OFFICE.BIN дані про працівників і виводить у файл OFFICE.sol працівників, які народилися в березні місяці.

Варіант 5.

1. Описати структуру з ім’ям ZARPLATA, яка містить наступні поля:
- Name – прізвище ім’я та по батькові працівника;
- Date – дата народження (структура: day; month, year - день, місяць, рік);
- Work_day – кількість відпрацьованих днів;
- Stavka – ставка з урахуванням на 24 робочих дні;
- Naraxovano – нараховано заробітної плати з урахуванням ставки і кількості відпрацьованих днів;
- Do_viplati – сума виплати заробітної плати працівнику з вирахуванням 20% податку.

2. Написати програму, що використовує дану структуру і виконує наступні дії:
- вводить з текстового файлу BUHGALTER.dat дані , згідно структури, і створює список;
- з даного списку видалити всіх працівників зарплата яких нараховано більше 500 грн.
- З новоствореного списку записати в двійковий файл BUHGALTER.new ставка яких менша 450 грн.

3. Написати програму яка зчитує з двійкового файлу BUHGALTER.new дані записує.у файл BUHGALTER.rez працівників, які відпрацювали менше 18 днів в даному місяці.

Варіант 6.

1. Описати структуру з ім’ям STUDENT, яка містить наступні поля:

- Name – Прізвище та ініціали;
- Year – рік народження:
- Bal – оцінки з 4 предметів (масив з 4 елементів)

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з текстового файлу Group.dat дані згідно структури і створює список;
- з даного списку видалити всіх студентів середній бал яких менший 3;
- з новоствореного списку записати в двійковий файл Group.new студентів 1985 року народження.

3. Написати програму яка зчитує з двійкового файлу Group.new дані записує.у файл Group.rez студентів які навчаються тільки на відмінно.

Варіант 7.

1. Описати структуру з ім’ям SKLAD, яка містить наступні поля:

- Name – Назва товару;
- Type – одиниця вимірювання:
- Quantity – кількість одиниць товару;
- Cost – ціна одиниці товару.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з текстового файлу Baza.DAT, список товарів і дані про них згідно заданої структури (дані впорядковані по ціні);
- Добавляє з клавіатури новий товар зберігаючи впорядкованість;
- З новоствореного списку записати у двійковий файл Baza.BIN товар кількість якого більша 100.

3. Написати програму яка зчитує з файлу Baza.BIN дані і виводить у файл Baza.sol товар ціна якого <1000 грн.

Варіант 8.

1. Описати структуру з ім’ям TRAIN, яка містить наступні поля:

- Nazv – Назва
- Numer – номер поїзда;
- Date – дата відправлення (структура: day; month, year - день, місяць, рік);
- Cina – ціна квитка.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з текстового файлу ZDPARK.dat дані згідно структури TRAIN, використовуючи список;
- Видаляє з списку всі поїзди які відправляються в травні;
- Виводить у двійковий файл ZDPARK.rez всі рейси, які час відправлення яких після 9.00 і до 19.00.

3. Написати програму яка зчитує з файлу Baza.BIN дані і виводить у файл Baza.sol поїзди які відправляються до Одеси.

Варіант 9.

1. Описати структуру з ім’ям ABONENT, яка містить наступні поля:
- Name – прізвище абонента;
- Init – ініціали абонента;
- Nomer – номер телефону;
- Adress –домашня адреса.

2. Написати програму, що використовує дану структуру і виконує наступні дії:
- вводить з текстового файлу Phone.dat дані згідно структури ABONENT, використовуючи список (дані у файлі записані у алфавітному порядку по прізвищу);
- Добавляє до списку нового абонента зберігаючи алфавітний порядок;
- Виводить у двійковий файл Phone.rez всі абоненти номери яких починаються числом 25.

3. Написати програму яка зчитує з файлу Phone.rez дані і виводить у файл Phone.sol абонентів з даним прізвищем (прізвище вводиться з клавіатури)

Варіант 10.

1. Описати структуру з ім’ям AEROFLOT, яка містить наступні поля:
- Nazv –назва пункту призначення;
- Numer – номер рейсу;
- Type –тип літака;
- Time –час відправлення.

2. Написати програму, що використовує дану структуру і виконує наступні дії:
- вводить з текстового файлу Reys.dat дані згідно структури AEROFLOT, використовуючи список (дані у файлі записані у порядку зростання по номеру рейса);
- Добавляє до списку новий рейс зберігаючи порядок;
- Виводить у двійковий файл Reys.bin всі рейси які відправляються до Парижу.

3. Написати програму яка зчитує з файлу Reys.bin дані і виводить у файл Reys.sol всі рейси, які відправляються після 15.00.

Варіант 11.

1. Описати структуру з ім’ям DETAL, яка містить наступні поля:
- Name – назва деталі;
- Sort – сорт виробу;
- Date –дата виготовлення (структура: day; month, year - день, місяць, рік);
- Quant – кількість;
- Cost - ціна деталі.

2. Написати програму, що використовує дану структуру і виконує наступні дії:

- вводить з текстового файлу Sklad.dat дані згідно структури DETAL, використовуючи список (дані у файлі записані у порядку зростання по назві деталі);
- Видаляє з даного списку всі деталі виготовлені в 2000 році.
- Виводить у двійковий файл Sklad.bin всі деталі ціна яких більша 50 грн.

3. Написати програму яка зчитує з файлу Sklad.bin дані і виводить у файл Sklad.sol всі деталі другого сорту.

Лабораторна робота №4. Розробка програм з використанням класів

Мета: Набути уміння та навички розробки та описання програм з класами. Написання конструкторів, деструкторів, методів(функцій-членів)

Програмне забезпечення: Dev C++, Visual C++

Контрольні питання.

1. Який синтаксис опису класу?
2. Що таке поля, методи класу?
3. Які є специфікатори доступу класу, їхнє призначення?
4. Як можна забезпечити доступ до елементів класу?
5. Що таке конструктор? Які правила створення та роботи конструктора ?
6. Що таке конструктор позамовчунні, конструктор копіювання?
7. Що таке деструктор? Які правила створення та роботи деструктора ?
8. Які існують ініціалізації елементів у конструкторах?
9. Який порядок виклику конструкторів та деструкторів?

Зміст завдання

Варіант 1.

1. Створити клас STUDENT, яка містить наступні поля:

- Name – Прізвище та ініціали;
- Year – рік народження:
- Bal – оцінки з 4 предметів (масив з 4 елементів)

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних Group, що складається з N змінних типу STUDENT;
- Виводить на екран прізвища і рік народження студентів середній бал яких > 4.0;

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 2.

1. Створити клас SKLAD, яка містить наступні поля:

- Name – Назва товару;
- Type – одиниця вимірювання:
- Quantity – кількість одиниць товару;
- Cost – ціна одиниці товару.

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних SHOP, що складається з N змінних типу SKLAD;
- Виводить на екран ціну та кількість товару, назва якого вводиться з клавіатури або виводить повідомлення про його відсутність.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp.

Варіант 3.

1. Створити клас TRAIN, яка містить наступні поля:

- Nazv – Назва
- Numer – номер поїзда;
- Date – дата відправлення
- Time – час відправлення поїзда.

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних AVTOPARK що складається з N змінних типу TRAIN;
- Виводить на екран всі рейси, які час відправлення яких після 15.00 по введеній даті.

3. 3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 4.

1. створити клас ABONENT, яка містить наступні поля:

- Name – прізвище абонента;
- Init – ініціали абонента;
- Nomer – номер телефону;
- Adress –домашня адреса.

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних TELEFON, що складається з N змінних типу ABONENT;
- Виводить на екран прізвище, ініціали та домашню адресу за введеним номером телефону, або виводить повідомлення про його відсутність.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 5.

1. Написати клас AEROFLOT, яка містить наступні поля:

- Nazv –назва пункту призначення;

- Numer – номер рейсу;
- Type –тип літака;
- Time –час відправлення.

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних ROZKLAD, що складається з N змінних типу AEROFLOT;
- Виводить на екран всі номери рейсів, та час відправлення літаків які відправляються в введений з клавіатури пункт призначення або повідомляє про відсутність таких рейсів.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 6.

1. Написати клас DETAL, яка містить наступні поля:

- Name – назва деталі;
- Sort – сорт виробу;
- Date –дата виготовлення
- Quant – кількість;
- Cost - ціна деталі.

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних ZAKAZ, що складається з N змінних типу DETAL;
- Виводить на екран всі деталі I сорту які виготовлені пізніше заданої дати, яка введена з клавіатури.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 7.

1. Написати клас BOOK, яка містить наступні поля:

- Name – назва книги;
- Avtor – автори книги;
- Data –дата друку;
- Cost - ціна книги.

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних SHOP, що складається з N змінних типу BOOK;
- Виводить на екран всі книги які були надруковані в заданому році.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 8.

1. Написати клас TOVAR, яка містить наступні поля:

- Name – назва товару;
- Cost_Z – ціна закупки товару;
- Cost_P - ціна продажу товари.
- Quantity –кількість одиниць товару;
- Pributok – прибуток.

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних SHOP, що складається з N змінних типу TOVAR і обчислює прибуток по кожному товару;
- Виводить на екран всі товари, які мають найбільший прибуток.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 9.

1. Написати клас VISTAVA, яка містить наступні поля:

- Nazva – назва вистави;
- Date – дата вистави;
- Cost - ціна квитка.
- Day_week – день неділі;

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних THEATRE , що складається з N змінних типу VISTAVA;
- Виводить на екран всі вистави і дати їх проходження, які відбудуться в заданий день тижня.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 10.

1. Написати клас WORKER, яка містить наступні поля:

- Name – Ім’я працівника;
- Surname – Прізвище працівника ;
- Date – дата народження;
- Pos - посада працівника.
- Zarplata – заробітна плата працівника

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних OFFICE, що складається з N змінних типу WORKER;

- Виводить на екран всіх працівників старших 30 років, які мають заробітну плату меншу введеної з клавіатури.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 11.

1. Написати клас ZARPLATA, який містить наступні поля:

- Name – прізвище ім’я та по батькові працівника;
- Date – дата народження;
- Work_day – кількість відпрацьованих днів;
- Stavka – ставка з урахуванням на 24 робочих дні;
- Naraxovano – нараховано заробітної плати з урахуванням ставки і кількості відпрацьованих днів;
- Do_viplati – сума виплати заробітної плати працівнику з вирахуванням 20% податку.

2. Написати програму, що використовує даний клас і виконує наступні дії::

- вводить з клавіатури масив даних BUHGALTER, що складається з N змінних типу ZARPLATA і підраховує поля нараховано та до виплати ;
- Виводить на екран всіх працівників з їхньою заробітною платою, які відпрацювали менше 15 робочих днів.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 12.

1. Написати клас COIN, яка містить наступні поля:

- Country – держава виготовлення;
- Name – назва монети;
- Year – рік виготовлення;
- Nominal – номінал;
- Price – ціна;

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних COIN_COLECTION, що складається з N змінних типу COIN;
- Виводить на екран список монет 1961 року.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 13.

1. Написати клас CD, який містить наступні поля:

- Name – назва диска;
- Date – дата запису диска(структура: month, year - місяць, рік);

- Theme – тема;
- SIZE – розмір;

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних CD_COLECTION, що складається з N змінних типу CD;
- Виводить на екран список дисків які були записані у 2005 році.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Варіант 14.

1. Написати клас VIDEO, який містить наступні поля:

- Name – назва відео фільму;
- Year – рік зйомки фільму;
- Genre – жанр фільму;
- Rezhiser – режисер;

2. Написати програму, що використовує даний клас і виконує наступні дії:

- вводить з клавіатури масив даних VIDEO_COLECTION, що складається з N змінних типу VIDEO;
- Виводить на екран список фільмів, по введеному з клавіатури режисеру.

3. Програму створити в трьох файлах: заготовочний файл *.h з описом класу, файл з функціями класу *.cpp, та головна функція main.cpp. Доступ до полів класу виконати через методи класу.

Лабораторна робота №5. Перевантаження операцій

Мета: Набути уміння та навички розробки та описання програм з використанням перевантаженням операцій.

Програмне забезпечення: Dev C++, Visual C++

Теоретичні питання.

7. Що таке перевантаження операцій?
8. Який синтаксис перевантаження?
9. Які операції неможна перевантажувати?
10. За допомогою якої операції виконується перевантаження операцій?
11. Коли необхідно перевантажувати операцію присвоєння?
12. Як передаються параметри під час перевантаження операції?

Зміст завдання

Описати клас, що реалізовує вказаний нижче тип даних. Клас повинен містити конструктор та подані нижче операції над об’єктами (плюс обов’язково операції порівняння та присвоєння) з використання перевантаженням операцій.

Написати програму, яка демонструє роботу з об’єктами цього класу. Програма повинна містити меню для перевірки усіх методів цього класу і операцій.

Варіант	Тип даних	Операція
1	Комплексні числа	Сума, добуток, різниця, частка
2	Вектор у просторі	Додавання векторів, векторний добуток двох векторів
3	Множина цифр	Вилучення елемента, об’єднання множин, перетин множин
4	Вектор у площині	Віднімання та складання векторів
5	Рядок	Додавання рядків, копіювання рядків
6	Дроби(ціла частина, чисельник, знаменник)	Додавання дробів, ділення дробів
7	Вектор у просторі	Множення вектора на число, віднімання векторів
8	Дата(число, місяць,рік)	Додавання , віднімання.
9	Кути(градуси, мінути, секунди)	Додавання, віднімання

10	Довгі числа	Віднімання множення
11	Час(години, хвилини, секунди)	Додавання годин, віднімання годин
12	Матриця	Віднімання, множення матриць
13	Довгі числа	Додвання, множення.
14	Дроби(ціла частина, чисельник, знаменник)	Віднімання дробів, множення дробів.
15	Комплексне число	Додавання, віднімання, множення ділення комплексних чисел.

Лабораторна робота №6. Множинне успадкування мовою С++.

Мета: отримання практичних навиків при використання множинного успадкування мовою С++

Програмне забезпечення: Dev C++, Visual C++

Теоретичні питання.

1. Що таке успадкування класів?
2. Який синтаксис успадкування класів?
3. Принцип керування доступом елементів класу при успадкуванні?
4. Як успадковуються конструктори, деструктори.
5. Як викликаються конструктори, деструктори успадкованих класів?

Зміст завдання

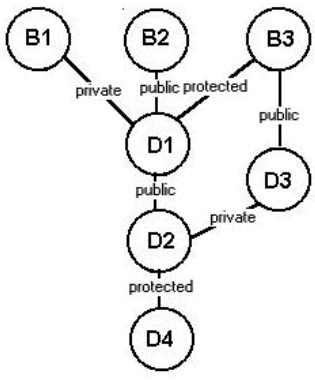
Необхідно побудувати ієрархію класів відповідно схемі успадкування, наведеній у варіанті завдання.

Кожен клас повинен містити конструктор ініціалізації, і функцію ***show*** для виведення значень.

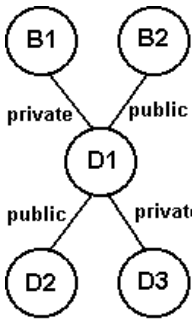
Функція ***main*** повинна ілюструвати ієрархію успадкування..

Варіанти індивідуальних завдань.

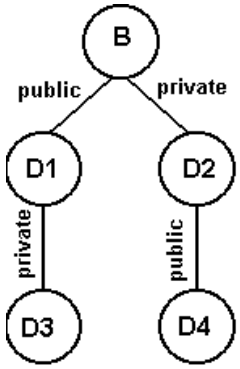
1



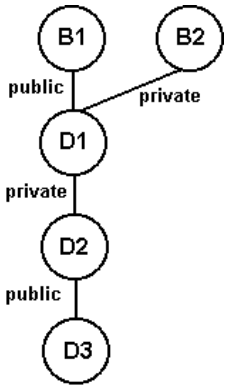
2



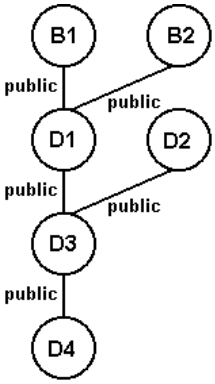
3



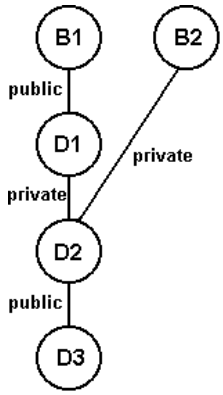
4



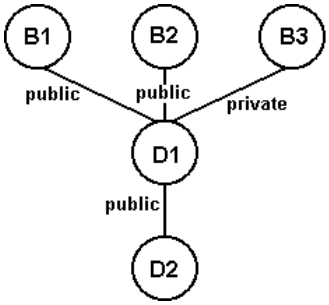
5



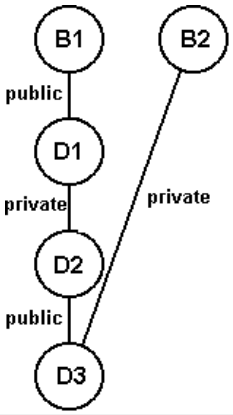
6



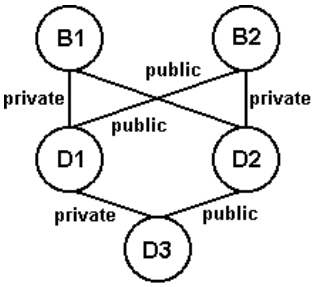
7



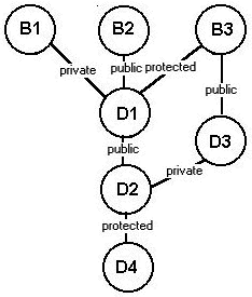
8



9

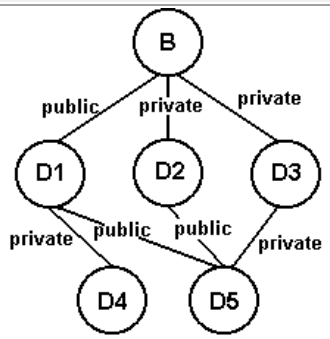


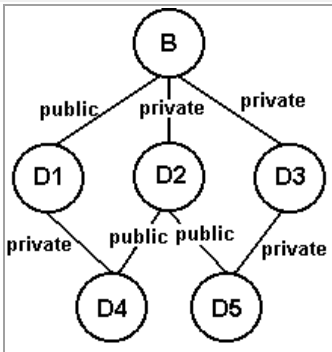
10



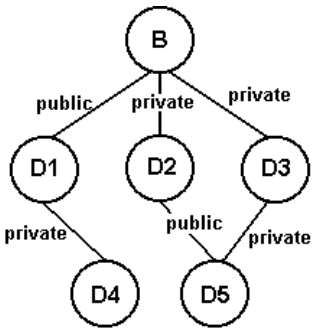
11

12

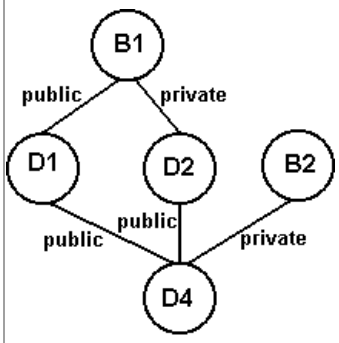




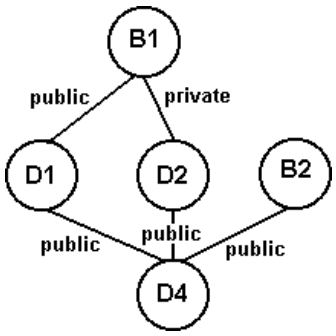
13



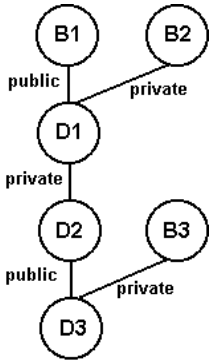
14



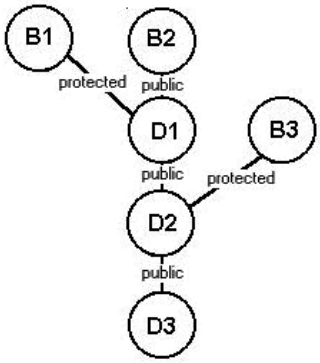
15



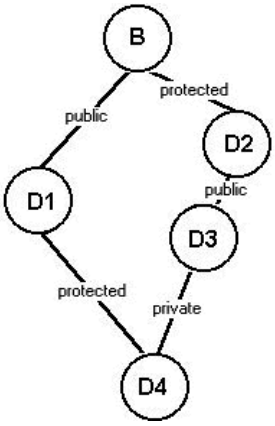
16



17

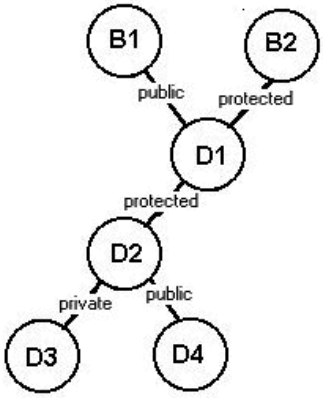


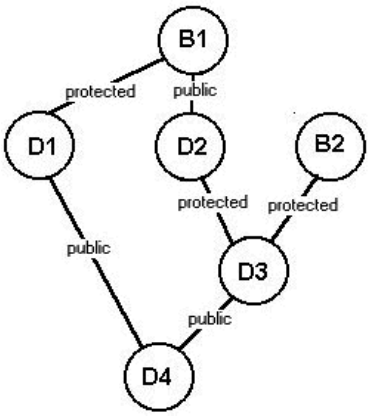
18



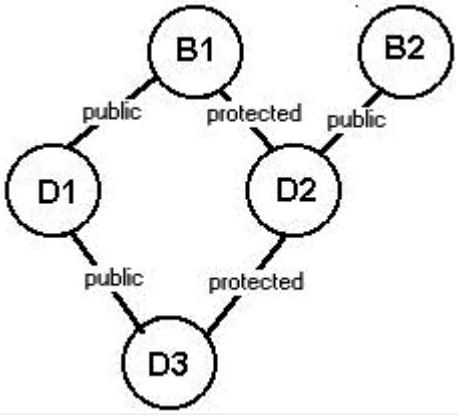
19

20

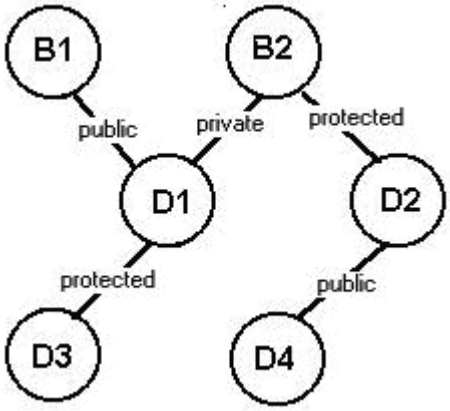




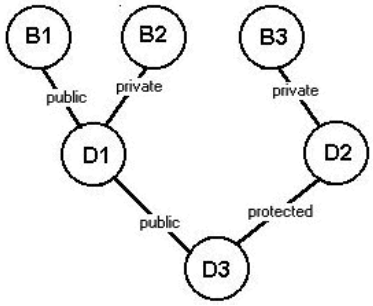
21



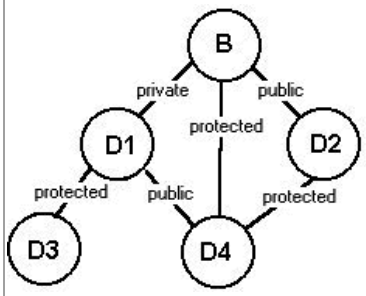
22



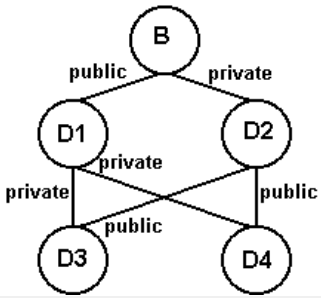
23



24



25



Приклад виконання (варіант 25)

Спочатку створимо необхідну ієрархію класів:

```
class B{
 int a;
public:
};

class D1: public B {
 int b;
public:
};

class D2: private B{
 int c;
public:
};

class D3: private D1, public D2 {
 int d;
public:
};

class D4: public D2, private D1 {
 int e;
public:
};
```

Створимо у всіх класах конструктори, які змогли б по ланцюгу успадкування ініціалізувати свої змінні і передати решту значень далі до конструктора базового класу.

```
 B(int x) { a=x; }
 D1(int x, int y) : B(y) { b=x;};
 D2(int x, int y) : B(y) { c=x;};
 D3(int x, int y, int z, int i, int j) : D1(y,z), D2(i,j) { d=x;}
 D4(int x, int y, int z, int i, int j) : D1(y,z), D2(i,j) { e=x;}
```

Тепер добавимо в кожен клас функцію ***show***, яка б виводила на екран змінну із розділу ***private*** класу, якому належить сама і викликала функцію ***show*** тих класів, які стоять вище по ієрархії успадкування.

```
 void show_B() { cout <<"B=   "<< a << "\n"; }
 void show_D1() { cout << "D1=  "<< b << "\n"; show_B();}
 void show_D2() { cout <<"D2=  "<< c << "\n"; show_B();}
 void show_D3() { cout << "D3=  "<<d << "\n"; show_D1(); show_D2();}
 void show_D4() { cout <<"D4=  "<< e << "\n"; show_D1(); show_D2();}
```

Повний текст програми

```
#include <iostream.h>
#include <stdlib.h>

class B{
 int a;
public:
 B() { };
 B(int x) { a=x; }
 void show_B() { cout <<"B=   "<< a << "\n"; }
};

class D1: public B {
 int b;
public:
 D1(int x, int y) : B(y) { b=x;};
 void show_D1() { cout <<"D1=  "<< b << "\n"; show_B();}
};

class D2: private B{
 int c;
public:
 D2(int x, int y) : B(y) { c=x;};
 void show_D2() { cout <<"D2=  "<< c << "\n"; show_B();}
};

class D3: private D1, public D2 {
 int d;
public:
 D3(int x, int y, int z, int i, int j) : D1(y,z), D2(i,j) { d=x;}
 void show_D3() { cout << "D3=  "<<d << "\n"; show_D1(); show_D2();}
};

class D4: public D2, private D1 {
 int e;
public:
 D4(int x, int y, int z, int i, int j) : D1(y,z), D2(i,j) { e=x;}
 void show_D4() { cout <<"D4=  "<< e << "\n"; show_D1(); show_D2();}
};

main() {

D3 temp(100,200,300,400,500);
D4 temp1(1,2,3,4,5);
```

```
cout << "D3 temp(100,200,300,400,500);\n";
cout << "D4 temp1(1,2,3,4,5);\n";
cout<< "\n Відповідно до ієрархії класу D3: \n";
temp.show_D3();
cout<< "\n Відповідно до ієрархії класу D4\n";
temp1.show_D4();
system(“pause”);
return 0;
}
```

Лабораторна робота №7. Виключні ситуації

Мета: Набути уміння та навички розробки та описання програм з використанням обробки виняткових(виключних) ситуацій

Програмне забезпечення: Dev C++, Visual C++

Теоретичні питання.

1. Що таке виключні ситуації.
2. Синтаксис виключних ситуацій.
3. Як контролюються виключні ситуації
4. Як відбувається перехват виключних ситуацій

Зміст завдання

Варіант1

Скласти програму знаходження площі трикутника за відомими сторонами з використанням обробки виключних ситуацій для випадків:

- одна з сторін трикутника=0
- одна з сторін трикутника від’ємна
- з даних сторін неможливо створити трикутник

Варіант2

Скласти програму значення функції на проміжку з використанням обробки виключних ситуацій для випадків для значень, які не задовільняють область визначення. $y=\left|\frac{|x-2|+1}{x-2}\right|$

Варіант3

Скласти програму знаходження найбільшого спільного дільника для N чисел з використанням обробки виключних ситуацій для випадків:

- числа від’ємні
- числа = нулю

Варіант4

Скласти програму знаходження середнього арифметичного додатних чисел з використанням обробки виключних ситуацій для випадків коли додатні числа відсутні

Варіант5

Скласти програму знаходження суму квадратних коренів чисел масиву використанням обробки виключних ситуацій для від’ємних чисел.

Варіант6
Скласти програму знаходження коренів біквадратного рівняння з використанням обробки виключних ситуацій для випадків коли рівняння немає розв’язків та виключення сторонніх коренів

Варіант7
Скласти програму знаходження n! з використанням обробки виключних ситуацій для від’ємних та дійсних чисел.

Варіант8
Скласти програму знаходження n того числа Фібоначі з використанням обробки виключних ситуацій для від’ємних та дійсних чисел та нуля.

Варіант9
Скласти програму знаходження об’єму зрізаного конуса з використанням обробки виключних ситуацій для від’ємних та нульових даних.

Варіант10
Скласти програму знаходження об’єму прямої трикутної призми за сторонами основи та висотою з використанням обробки виключних ситуацій від’ємних, нульових параметрів та випадку коли неможливо створити таку фігуру.

Варіант11
Скласти програму знаходження сили всесвітнього тяжіння з використанням обробки виключних ситуацій для від’ємних та нульових даних.

Варіант12
Скласти програму знаходження площі трапеції за основами та висотою з використанням обробки виключних ситуацій від’ємних, нульових, від’ємних параметрів.

Приклад виконання лабораторноъ роботи.
Виключні ситуації

```
#include<iostream>
#include<iostream.h>

int NSD(int a,int b)
{
        try
        { if (a==0 || b==0) throw "\n Dilenny na null";
          if (a<0)      throw "\n Nigative parametr 1";
```

```
          if (b<0)  throw "\n Nigative parametr 2";
            while(a!=b)
                 {if (a!=b) a-=b; else b-=a;}
         return a;
         }
catch(const char *report)
 {
   cerr<<report<<" a= "<<a<<"  b="<<b;
   return 0;
 }
}

int main()
{
cout<<"\nNSD(64,44)="<<NSD(64,44);
cout<<"\n\nNSD(0,44)="<<NSD(0,44);
cout<<"\n\nNSD(64,-44)="<<NSD(64,-44);
return 0;
}
```

Варsант 2

```
#include<iostream.h>

int NSD(int a,int b)
{
          if (a==0 || b==0) throw "\n Dilenny na null";
          if (a<0)      throw "\n Nigative parametr 1";
          if (b<0)  throw "\n Nigative parametr 2";
            while(a!=b)
                 {if (a!=b) a-=b; else b-=a;}
         return a;
}
int main()
{        try{
    cout<<"\nNSD(64,44)="<<NSD(64,44);
    cout<<"\n\nNSD(0,44)="<<NSD(0,44);
    cout<<"\n\nNSD(64,-44)="<<NSD(64,-44);
         }
catch(const char *report)
{   cerr<<report;  }
return 0;
}
```

Варіант 3

```
#include<iostream>
#include<iostream.h>
struct DATA
{
  int m,n;//значення
  char *s; //константа виключення
  DATA (int a,int b,char *c)
  {n=a;m=b;s=c;}
};
int NSD(int a,int b)
{
         if (a==0 || b==0) throw DATA(a,b,"\n Dilenny na null");
          if (a<0)      throw DATA(a,b,"\n Nigative parametr 1");
          if (b<0)  throw DATA(a,b,"\n Nigative parametr 2");
            while(a!=b)
                {       if (a!=b) a-=b; else b-=a;          }
        return a;
}

int main()
{
        try{
   cout<<"\nNSD(64,44)="<<NSD(64,44);
   cout<<"\n\nNSD(0,44)="<<NSD(0,44);
   cout<<"\n\nNSD(64,-44)="<<NSD(64,-44);
        }
catch(DATA d)
{
  cerr<<d.s<<" a="<<d.m<<" b="<< d.n;
 }
return 0;
}
```

Лабораторна робота №8. Клас String

Мета: Набути уміння та навички розробки та описання програм з використанням класу String..

Програмне забезпечення: Dev C++, Visual C++

Теоретичні питання.

1. Що таке клас string?
2. Які конструктори існують в класі string?
3. Які операції допустимі для об’єктів классу string?
4. Як реалізовуються методи обробки рядка, об’єкту string?
5. Які методи обробки рядка існують?
6. Які методи переводять з типу string в тип char?

Зміст завдання

Варіант 1

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість слів, які мають непарну довжину; виводить на екран частоту кожної літери.

2. Вводиться з клавіатури час у форматі хв:сек наприкдад 12: 11. Записати даний час текстом: Дванадцять хвилин одинадцять секунд. Перед виведенням результат помістити в об’єкт типу string.

Варіант 2

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка перевіряє, чи співпадає кількість відкритих і закритих дужок у введеному рядку (перевірити для круглих та квадратних дужок); виводить на екран найдовше слово.

2. Вводиться з клавіатури час у форматі гг:хв наприкдад 13:12. Записати даний час текстом: Тринадцять годин дванадцять хвилин. Перед виведенням результат помістити в об’єкт типу string.

Варіант 3

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість різних слів, що входять до заданого тексту; виводить на екран кількість використаних символів.

2. Вводиться з клавіатури дата у форматі місяць:рік наприкдад 06:2011. Записати дату текстом: червень дві тисячі дванадцятого року. Перед виведенням результат помістити в об’єкт типу string.

Варіант 4

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість слів у тексті; слово, що містить найбільшу кількість голосних літер.

2. Вводиться з клавіатури дата у форматі дата:місяць наприкдад 13:12. Записати дату текстом: Тринадцяте грудня. Перед виведенням результат помістити в об’єкт типу string.

Варіант 5

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість розділових знаків у тексті; виводить всі слова, що мають парну кількість літер.

2. Вводиться з клавіатури кут у форматі градусах,мінутах гг:хв наприкдад 13,12. Записати кут текстом: Тринадцять градусів вісімнадцять мінут. Перед виведенням результат помістити в об’єкт типу string.

Варіант 6

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість великих літер у тексті; виводить на екран слова, що мають найменшу кількість літер.

2. Вводиться з клавіатури кут у радіанах наприкдад 52 рад.. Записати даний кут текстом П’ятдесят два радіани. Перед виведенням результат помістити в об’єкт типу string.

Варіант 7

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість чисел у тексті (не цифр, а саме чисел); виводить на екран всі слова, що складаються тільки з латинських літер

2. Вводиться з клавіатури вартість товару у грн.,коп. наприкдад 52,12.. Записати дану суму текстом П’ятдесят дві гривні дванадцять копійок. Перед виведенням результат помістити в об’єкт типу string.

Варіант 8

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість цифр у тексті; виводить на екран слова, що починаються з приголосних літер.

2. Вводиться з клавіатури вартість товару у доларах/центах наприкдад 52,12.. Записати дану суму текстом П’ятдесят два долари дванадцять центів. Перед виведенням результат помістити в об’єкт типу string.

Варіант 9

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість слів у тексті які закінчуються на голосну літеру; виводить на екран всі слова довжина яких менша п’яти символів;.

2. Вводиться з клавіатури вартість товару у рублях/копійках наприкдад 52,12.. Записати дану суму текстом П’ятдесят два рублі дванадцять копійок . Перед виведенням результат помістити в об’єкт типу string.

Варіант 10

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість слів у тексті, які починаються з голосної літери; виводить на екран всі слова, які містять непарну кількість приголосних літер.

2. Вводиться з клавіатури розмір файлу байтах наприкдад 52128.. Записати даний розмір текстом в кілобайтах/байтах. П’ятдесят кілобайт дев’ятсот двадцять вісім байт. Перед виведенням результат помістити в об’єкт типу string.

Варіант 11

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка змінює всі великі літери, що входять до тексту на відповідні малі; виводить на екран найдовше слово.

2. Вводиться з клавіатури відстань в кілометрах/метрах 52,12.. Записати данк відстань текстом П’ятдесят два кілометри дванадцять метрів. Перед виведенням результат помістити в об’єкт типу string.

Варіант 12

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість слів, які мають однакову кількість приголосних і голосних літер; виводить на екран найдовше слово.

2. Вводиться з клавіатури маса в кілограмах/грамах наприкдад 52,12.. Записати дану масу текстом П’ятдесят два кілограми дванадцять грам. Перед виведенням результат помістити в об’єкт типу string.

Варіант 13

1. З клавіатури вводиться текстовий рядок. Скласти програму, яка виводить на екран всі символи, які розташовані після першого символу «:»; підраховує кількість речень, що має непарну кількість слів.;.

2. Вводиться з клавіатури довжина відрізка в сантиметрах/метрах наприкдад 52,12. Записати дану відстань текстом П’ятдесят два сантиметри дванадцять міліметрів. Перед виведенням результат помістити в об’єкт типу string.

Варіант 14

З клавіатури вводиться текстовий рядок. Скласти програму, яка підраховує кількість слів у кожному речені; виводить на екран кожне речення;.

2. Вводиться з клавіатури маса в тонах/кілограмах наприклад 52,012. Записати дану масу текстом П’ятдесят дві тони дванадцять кілограм. Перед виведенням результат помістити в об’єкт типу string.

Варіант 15

З клавіатури вводиться текстовий рядок. Скласти програму, яка інвертує рядок, подаючи його у зворотному вигляді; підраховує кількість чисел у тексті.

2. Вводиться з клавіатури об’єм в літрах/мілілітріах наприкдад 52,012. Записати даний об’м текстом П’ятдесят два літри дванадцять мілілітрів. Перед виведенням результат помістити в об’єкт типу string.

Лабораторна робота №9. Контейнери

Мета: Набути уміння та навички розробки та описання програм з використанням контейнерів та бібліотеки <algorithm>.

Програмне забезпечення: Dev C++, Visual C++

Теоретичні питання.

1. Що таке контейнер.
2. Які є контейнери. Їх призначення.
3. Для чого призначений вектор? Які методи він підтримує?
4. Для чого призначений черга? Які методи він підтримує?
5. Для чого призначений список? Які методи він підтримує?
6. В якому з контейнерів можна добавити, видалити елементи в кінець.
7. В якому з контейнерів можна добавити, видалити на початок?
8. В якому з контейнерів можна добавити, вставити елементи всередину.

Зміст завдання

Варіант1
З текстового файлу зчитати дійсні числа у вектор. У інший вектор перенести всі від’ємні числа. Утворені вектори відсортувати і вивести на екран.

Варіант2
З текстового файлу зчитати цілі числа помістити в чергу. Перенести у вектор V1 першу половину елементів, у V2 половину. Перший вектор відсортувати по зростанню, другий по спаданню. Вектори вивести на екран.

Варіант3
З текстового файлу зчитати натуральні числа у список. Вивести на екран номери тих елементів, які не змінили місце після сортування.
Варіант4
З текстового файлу зчитати дійсні числа у список . Знайти середнє арифметичне 10 найбільших чисел.

Варіант5
З текстового файлу зчитати цілі числа у вектор. У чергу записати всі непарні числа у відсортованому порядку. Вивести на екран елементи черги.

Варіант6

З текстового файлу зчитати натуральні числа у список. У чергу занести всі парні числа відсортовані по спаданню. Видалити найбільший та найменший елемент. Вивести на екран елементи утвореної черги.

Варіант7
З текстового файлу зчитати дійсні числа у вектор. З вектора видалити всі додатні числа. Утворені елементи перенести у список у зворотному порядку. Вивести на екран утворений список.

Варіант8
З текстового файлу зчитати цілі числа у список. У чергу перенести у всі числа, які менші за середнє арифметичне списку, розмістити їх у зворотньому порядку. Вивести на екран утворений список.

Варіант9
З текстового файлу зчитати дійсні числа у чергу. У вектор записати всі числа, які більші по модулю за середнє арифметичне додатніх чисел черги у зворотньому порядку.

Варіант10
З текстового файлу зчитати натуральні числа у список. У чергу записати квадрати чисел, які менші за перше число. Вивести на екран утворену чергу.

Варіант11
З текстового файлу зчитати дійсні числа у вектор. З вектора видалити всі непарні додатні числа. Утворені елементи перенести у список у зворотному порядку. Вивести на екран утворений список.

Варіант12
З текстового файлу зчитати цілі числа у список. У чергу перенести у всі числа, які більші за середнє арифметичне списку, розмістити їх у зворотньому порядку. Вивести на екран утворений список.

Варіант13
З текстового файлу зчитати дійсні числа у чергу. У вектор записати всі числа, які менші по модулю за середнє арифметичне додатніх чисел черги у зворотньому порядку.

Варіант14
З текстового файлу зчитати натуральні числа у список. У чергу записати квадрати чисел, які більші за перше число. Вивести на екран утворену чергу.

ТЕСТОВІ ЗАВДАННЯ

1. Конструктор класу призначений для:
а) Створення та ініціалізації об’єкта.
б) Видалення об’єкта
в) Модифікації об’єкта
г) Опису об’єкта

2. Які операції неможна перевантажувати?
а) >
б) [ ]
в) ::
г) ++

3. Як оголосити файлову змінну використовуючи бібліотеку stdio.h?
а) file f;
б) FILE f;
в) TEXT f:
г) ifstream f;

4. Як правльно відкрити текстовий файл для читання, використовуючи можливості бібліотеки stdio.h?
а) f=fopen(“text.txt”,”rb”);
б) FILE f;
в) f=fopen(“text.dat”,”r”);
г) ifstream f (“text.txt”);

5. Для встановлення вказівника в кінець бінарного файлу використовується функція:
а) fseek(f, 0, SEEK_END);
б) ftell(f);
в) while(!feof(f){…..}
г) fseek(f, 0, SEEK_CUR);

6. Для визначення поточної позиції вказівника у файлі використовується функція:
а) fseek(f, 0, SEEK_END);
б) ftell(f);
в) while(!feof(f){…..}
г) fseek(f, 0, SEEK_CUR);

7. Для запису елементу структури в бінарний файл використовується функція
а) fwrite(&i,sizeof(int),1,stream);

б) ftell(f);
в) while(!feof(f){…..}
г) fseek(f, 0, SEEK_CUR);

8. Скільки байт займає структура:
```
struct  P
{
int x;
char b[20];
};
```

а) 4 байти
б) 20 байт
в) 2 байти
г) 24 байти

9. Скільки байт займає об’єднання:
```
union  P
{
int x;
int64 y;
char b[20];
};
```

а) 4 байти
б) 8 байт
в) 20 байти
г) 32 байти

10. Закриті елементи класу мають специфікатор доступу:
а) private
б) public
в) protected
г) virtual

11. Метод класу призначений для ініціалізації полів об’єкту, і визивається під час оголошення об’єкту класу називається:
а) деструктор
б) конструктор
в) поле

12. Для визначення розміру об’єкту класу використовується функція:
а) strlen()
б) size()
в) sizeof()

г) length()

13. Які операції неможна перевантажувати?
а) +
б) ::
в) new
г) =

14. Конструктором копіювання називається конструктор:
а) в якого відсутні параметри;
б) в якого лише один параметр;
в) в якого параметри – всі поля класу;
г) в якого один параметр – екземпляр цього ж класу

15. Що означає ключове слово try?
а) блок, що контролюється
б) блок обробки виключення
в) секція-ловушка
г) генератор виключення

16. Що означає ключове слово catch?
а) блок, що контролюється
б) блок обробки виключення
в) секція-ловушка
г) генератор виключення

17. Конструктор без параметрів називається:
а) конструктор копіювання;
б) деструктор;
в) метод класу;
г) конструктор по замовчуванню.

18. Вказівник this означає:
а) об’єкт, що викликає метод класу;
б) метод класу;
в) конструктор, без параметрів;
г) об’єкт, що передається параметром.

19. Деструктор призначений:
а) для ініціалізації об’єкту класу;
б) для знищення об’єкту класу;
в) для виклику методів класу;
г) для обнуління змінних класу.

20. Конструктором перетворення називається конструктор:
а) який має один параметр – об’єкт цього ж класу;
б) який має лише один параметр іншого типу;
в) який має більше ній один параметр;
г) який немає параметрів.

21. Вкажіть розмір даного класу:

```
struct  A{
       private:
            int x;
            char m[20];
       public:
          static int n;
          A() { x=0; }
          void out(){cout<<x;}
};
```

а) 20 байт;
б) 24 байта;
в) 28 байт
г) 36 байт.

22. Під час перевантаження у функцію-операцію передається два параметри:
а) коли функція знаходиться і класі;
б) коли виконується операція над двома значеннями;
в) коли перевантажується бінарна операція;
г)коли функція реалізована поза класом.

23. Яка з наступних операцій не перевантажується:
а) +;
б) =;
в) new;
г) sizeof()

24. По замовчуванні елементи класу мають специфікатор доступу:
а) private;
б) public;
в) protected;
г) friend.

25. Специфікатор доступу protected:
а) відкриває доступ для елемнентів класу тільки для успадкованих класів;
б) відкриває доступ для елемнентів класу;
в) закриває доступ для елементів класу під час успадкування;

26. Генерація виключення відбувається за допомогою ключового слова:
а) try;
б) throw;
в) catch;
г) return;

27. Для копіювання одному рядку типу string фрагмент інфшого використовують метод:
а) assign();
б) append();
в) copy();
г) insert();

28. Для добавлення до одного рядка типу string фрагмент інфшого використовують метод:
а) assign();
б) append();
в) copy();
г) insert();

29. Для вставлення в один рядк типу string фрагмент інфшого використовують метод:
а) assign();
б) append();
в) copy();
г) insert();

30. Для очищення рядка типу string т використовують метод:
а) errase();
б) delete();
в) replace();
г) clear();

31. Для заміни одним рядком типу string фрагменту іншого використовують метод:
а) errase();
б) delete();
в) replace();
г) clear();

32. Для обміну двох рядків типу string використовують метод:
а) errase();
б) swap();
в) replace();
г) clear();

33. Для перетворення рядка типу string в массив типу char використовують метод:
а) c_str();
б) errase();
в) replace();
г) clear();

34. Для перетворення рядка типу string в массив типу char з добавленням в кінець рядка нуль-символ використовують метод:
а) errase();
б) c_str();
в) data()
г) replace();

35. Для визначення довжини рядка типу string використовують метод:
а) length();
б) dlina();
в) strlen();
г) empty();

36. Для визначення чи пустий рядоктипу string використовують метод:
а) errase();
б) c_str();
в) empty();
г) compare();

37. Для визначення розміру контейнера vector використовують метод:
а) sizeof();
б) size();
в) lehgth();
г) max_size();

38. Для добавлення елемента в кінець контейнера list використовують метод :
а) pop_end();
б) push_front();
в) push_back();
г) pop_back();

39. Для добавлення елемента на початок контейнера list використовують метод :
а) pop_end();
б) push_front();
в) push_back();
г) pop_back();

40. Для видалення елемента з контейнера list використовують метод :
а) erase();
б) delete();
в) del();
г) empty();

41. Ітератор – це:
а) номер елемента контейнера;
б) вказівник на елемент контейнера;
в) послідовний контейнер;
г) асоціативний контейнер;

42. Для заповнення масиву певним числами за вказаним правилом використовують функцію … з бібліотеки <algorithm>
а) generate();
б) random_shufle();
в) rand();
г) random();

43. Для виконання переміщення еліментів масиву в рівномірно випадковому порядку використовують функцію … з бібліотеки <algorithm>
а) generate();
б) random_shufle();
в) rand();
г) random();

44. Для переміщення елементів масиву в зворотному порядку використовують функцію … з бібліотеки <algorithm.h>
а) generate();
б) rotate();
в) rand();
г) reverse();

45. Для переміщення елементів масиву в циклічному порядку використовують функцію … з бібліотеки <algorithm.h>
а) generate();
б) rotate();
в) rand();
г) reverse();

46. Для створення наступної лексикографічної перестановки елементів масиву використовують функцію … з бібліотеки <algorithm.h>
а) lecsicografical_compare();
б) next_permutation();

в) prev_permutation();
г) rotate();

47. Для створення попередньої лексикографічної перестановки елементів масиву використовують функцію … з бібліотеки <algorithm.h>
а) lecsicografical_compare();
б) next_permutation();
в) prev_permutation();
г) rotate();

48. Для сортування елементів послідовності із збереженням порядку однакових елементів використовують функцію … з бібліотеки <algorithm.h>
а) stable_sort();
б) sort_heap();
в) sort();
г) sort_stable();

49. Для роботи з множинами призначень контейнерний клас:
а) map;
б) vector;
в) set;
г) lost;

50. Для видалення елемента з множини set використовують метод:
а) delete();
б) remove();
в) erase();
г) del();

51. Для визначення кількості елементів у множині set використовують метод:
а) size();
б) sizeof();
в) length();
г) end();

52. Для вставки в множину set елемент використовують метод:
а) ins();
б) save();
в) load();
г) insert();

53. Для копіювання частину рядка класу string використовують метод:
а) substr();

б) copy();
в) insert();
г) strstr();

54. Для визначення розміру контейнера типу vector використовують метод:
а) size();
б) sizeof();
в) length();
г) end();

55. Для видалення з рядка типу string фрагмент використовують метод:
а) errase();
б) delete();
в) replace();
г) clear();

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Керниган Б, Ридчи Д. Язык программирования Си. М. Финансы и статистика.1988.
2. Павловская Т.А. С/С++ Программирование на языке высокого уровня: учебное пособие для вузов СПб, Питер, 2002;
3. Павловская Т.А. С/С++ Структурное программирование. Практикум. учебное пособие для вузов СПб, Питер, 2005;
4. Подбельский В.В. Язык Си++. Финансы и статистика.2001.
5. Войтенко В.В. Морозов А.В. С/С++ теорія та практика ЖДТУ 2003
6. Глинський Я.М., Анохін В.Є., Ряжська В.А. С++ і С++ Builder. Навчальний посібник. – Львів; Деол, СПД Глинський, 2003. – 192 с.
7. Пол Лукас С++ под рукой. КиевНИПФ «Диа Софт» 1993
8. www.e-olimp.com – Інтернет-портал організаційно-методичного забезпечення дистанційних олімпіад з програмування.
9. Кнут Д. Искусство программирования для ЭВМ. -М.: Мир, 1976. -Т.2.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчально-методичне видання

ЖУКОВСЬКИЙ Сергій Станіславович,
ВАКАЛЮК Тетяна Анатоліївна

ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ МОВОЮ С++

*Навчально-методичний посібник для студентів напряму 6.040302 Інформатика**

Надруковано з оригінал-макета автора

Підписано до друку 03.11.16. Формат 60х90/16. Папір офсетний.
Гарнітура Times New Roman. Друк різографічний.
Ум. друк. арк. 5,8. Обл. вид. арк. 3,5. Наклад 300. Зам. 83.

Видавець і виготовлювач
Видавництво Житомирського державного університету імені Івана Франка
м. Житомир, вул. Велика Бердичівська, 40
Свідоцтво суб’єкта видавничої справи:
серія ЖТ №10 від 07.12.04 р.
електронна пошта (E-mail): zu@zu.edu.ua