НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

ПРОГРАМУВАННЯ ТА АЛГОРИТМІЧНІ МОВИ. ПРОГРАМУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ

до комп'ютерного практикуму

для студентів напрямів підготовки
124 «Системний аналіз»,
122 «Комп'ютерні науки та інформаційні технології»

Затверджено Методичною комісією «ІПСА» НТУУ «КПІ»
(Протокол № 7 від 17.05.2017)

Назарчук І.В.

Київ
НТУУ "КПІ"
2017

Програмування та алгоритмічні мови. Програмування: Методичні вказівки до комп'ютерного практикуму для студентів напрямів підготовки 124 «Системний аналіз», 122 «Комп'ютерні науки та інформаційні технології» / Уклад.: І.В. Назарчук. – К. НТУУ "КПІ", 2017. – 95 с.

Затверджено Методичною комісією «ІПСА» НТУУ «КПІ»

(Протокол № 7 від 17.05.2017)

ПРОГРАМУВАННЯ ТА АЛГОРИТМІЧНІ МОВИ. ПРОГРАМУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ

до комп'ютерного практикуму

для студентів напрямів підготовки
122 «Системний аналіз»,
124 «Комп'ютерні науки та інформаційні технології»

Укладач: Назарчук Ірина Василівна

Відповідальний редактор: О.Л. Тимощук, к.т.н, доцент

Рецензент: А.А. Гагарін, к.т.н.

Зміст

ВСТУП

Роботи комп’ютерних практикумів до кредитного модуля «Програмування» дисципліни «Програмування та алгоритмічні мови» виконуються студентами 1 курсу спеціальності 124 «Системний аналіз» галузі знань 12 «Інформаційні технології» денної форми навчання.

Основна мета виконання робіт комп’ютерних практикумів – це закріплення знань, які отримані на лекційних заняттях та в результаті самостійної роботи студента. Комп’ютерні практикуми призначені для отримання студентом навику з програмування на мові С++. В даних методичних вказівках зібрано завдання для закріплення декількох тем, що стосуються основ структурного та об’єктно-орієнтованого програмування мовою С++.

1. МЕТА ТА ОСНОВНІ ЗАВДАННЯ РОБІТ КОМП'ЮТЕРНИХ ПРАКТИКУМІВ

Метою проведення робіт є закріплення отриманих теоретичних відомостей, формування у студентів здатностей застосування основних можливостей роботи з динамічними структурами, файлової системи та об'єктно-орієнтованого підходу для розробки працюючих додатків з використанням мови програмування С++.

Згідно з вимогами робочої програми кредитного модуля студенти після виконання данного циклу робіт мають продемонструвати такі результати навчання.

Знання особливостей оголошення та використання динамічних структур даних; різновидів зовнішніх файлів та принципи роботи з ними; основні можливості об'єктно-орієнтованої мови та засоби їх використання; будову й загальні підходи до використання засобів стандартних бібліотек.

Уміння використати динамічну пам'ять для розміщення даних; створити зовнішній файл потрібної структури та працювати з ним; описати засобами С++, життєздатну абстракцію предметної області з використанням принципу інкапсуляції даних; побудувати ієрархію успадкування, використовувати поліморфізм; використовувати можливості узагальненого програмування для опису класів, функцій, при роботі зі стандартною бібліотекою, коректно інтерпретувати одержані результати; правильно оформити супровідну документацію відповідно до вимог чинного законодавства України зі стандартизації у вигляді звіту.

Досвід роботи з сучасними засобами автоматизації процесу об'єктно–орієнтованого програмування.

2. ПОРЯДОК ВИКОНАННЯ РОБІТ

2.1 Загальні вимоги

При виконанні робіт комп'ютерних практикумів (РКП) необхідно дотримуватись наступних вимог:

1. При роботі в лабораторії чи комп'ютерному класі слід дотримуватись правил техніки безпеки.
2. Ознайомитись з основами роботи у середовищі Dev C++, освоїти використання системи довідки.

2.2 Вимоги до етапів виконання кожного практикуму

Етапи виконання кожної РКП можна звести до наступного алгоритму:

1. проаналізувати умову задачі згідно з номером варіанту;
2. описати потрібні абстракції відповідно загального порядку виконання даної роботи, скорегувати відповідно варіанту;
3. для доведення працездатності кожної функції програми продумати тестові приклади, врахувати необхідність зробити потрібні для даної роботи висновки;
4. для зручності використання програми користувачем забезпечити дружній інтерфейс;
5. результати роботи оформити у вигляді звіту.

2.3 Загальні зауваження до оформлення програм

Щоб програму було легко читати та тестувати, потрібно:

1. широко використовувати коментарі;
2. обирати ідентифікатори так, щоб вони відображували зміст констант, змінних, функцій, ідентифікатори типів починати з великої букви;
3. реалізації функцій бажано надавати за межами опису класу;
4. кожному вводу даних повинно передувати повідомлення, які данні треба ввести і у якому вигляді.

3. ОФОРМЛЕННЯ ЗВІТУ ТА ПОРЯДОК ЙОГО ПОДАННЯ

Для кожної робіт комп'ютерного практикуму має бути підготовлений звіт. Без звіту студент не допускається до здачі РКП.

Звіт до РКП має містити:

1) формулювання теми роботи;
2) постановку завдання, яке необхідно виконати;
3) текст програми;
4) приклади тестів для перевірки працездатності програми;
5) висновки щодо роботи програми.

Шаблон титульної сторінки до звіту наведено у додатку 1.

Текст звіту виконується у відповідності до вимог стандартів машинним (на пристроях ЕОМ) способом. Формат аркуша - **А4**. Параметри сторінки (поля): верхнє – **25** мм, нижнє – **15** мм, ліве – **25** мм, праве – **10** мм. Основний текст набирається шрифтом **TimesNew Roman** розмір **12**, міжрядковий інтервал - **1,0**.

Вставлення результатів роботи програми забезпечити вставкою із сеансу **DOS**.

У висновках до комп'ютерного практикуму необхідно зазначити, чи вирішує програма поставлене завдання, вказати труднощі, з якими студент стикнувся під час виконання завдання.

4. ЗАХОДИ БЕЗПЕКИ ПІД ЧАС ВИКОНАННЯ РОБІТ КОМП'ЮТЕРНОГО ПРАКТИКУМУ В КОМП'ЮТЕРНОМУ КЛАСІ

До виконання лабораторних робіт допускаються студенти, які пройшли інструктаж з техніки безпеки з наступним записом про це в спеціальному журналі.

Загальні положення:

1. До виконання лабораторних робіт допускаються студенти, які пройшли інструктаж з техніки безпеки і правил поведінки в комп'ютерному класі.
2. Робота студентів в комп'ютерному класі дозволяється лише у присутності викладача.

Перед початком роботи необхідно:

1. Переконатися у відсутності видимих пошкоджень на робочому місці;
2. Розмістити на столі зошити, навчальні посібники так, що б вони не заважали роботі на комп'ютері;

При роботі в комп'ютерному класі категорично забороняється:

1. Знаходитися в класі у верхньому одязі;
2. Класти одяг , сумки та інші речі на клавіатуру, монітор і системний блок;
3. Знаходитися в класі з напоями та їжею;
4. Розташовуватися збоку або ззаду від включеного монітора;
5. Приєднувати або від'єднувати кабелі, торкатися роз'ємів, оголених дротів і розеток;
6. Пересувати комп'ютери і монітори;
7. Відкривати системний блок;
8. Включати і вимикати комп'ютери самостійно;
9. Намагатися самостійно усувати несправності в роботі обладнання;
10. Перекривати вентиляційні отвори на системному блоці та моніторі;

11. Видаляти і переміщати чужі файли;
12. Приносити і запускати комп'ютерні ігри.

Знаходячись в комп'ютерному класі, учні зобов’язані:

1. Дотримуватись тиші і порядку;
2. Виконувати вимоги викладача;
3. Знаходячись в мережі працювати лише під своїм ім'ям і паролем;
4. Дотримуватись режиму роботи (згідно Санітарних правил і норм);
5. При появі різі в очах, різкому погіршенні видимості, неможливості сфокусувати погляд або навести його на різкість, появи болю в пальцях і кистях рук, посилення серцебиття негайно покинути робоче місце, повідомити про це, викладача і звернутися до лікаря;
6. Після закінчення роботи завершити всі активні програми і коректно вимкнути комп'ютер;
7. Залишити робоче місце чистим.

Вимоги безпеки в аварійних ситуаціях:

1. При появі програмних помилок або збоях обладнання студент повинен негайно звернутися до викладача.
2. При появі запаху гару, незвичайного звуку негайно припинити роботу, і повідомити викладача.

5. РОБОТИ КОМП'ЮТЕРНОГО ПРАКТИКУМУ

5.1. Компютерний практикум № 1.

5.1.1. Тема

Динамічні структури даних та робота з файлами.

5.1.2. Мета роботи.

Навчитись працювати з динамічними структурами даних засобами С та С++ на прикладі динамічних масивів, масивів динамічних об'єктів і однозв'язних списків.

5.1.3. Основні теоретичні відомості

Структура. Це складений об'єкт, сформований з об'єктів, що належать до одного або різних типів даних. Елементи – складові можуть бути довільних типів, за винятком функцій.

Таким чином, структура – це новий тип даних, який забезпечує зручний спосіб організації пов'язаних за змістом змінних, оскільки дозволяють групу пов'язаних між собою змінних трактувати не як безліч окремих елементів, а як єдине ціле.

Допустимо окремо визначити тип і об'єкт (змінну) цього типу, допустимо це зробити одночасно. Синтаксис оголошення

struct <ідентифікатор типу> {<список компонентів>}[<список ідентифікаторів змінних>];
<ідентифікатор типу><список ідентифікаторів змінних>;

```
struct Person{
      char name[30];
      int day, month, year;
 } st{"Ann",2,3,1999}, arr[15], *p;
Person b, mas[2][2];                //в стилі С++
struct  Person c,*ptr=&st;          // в стилі С
```

Для доступу до поля структурного об'єкта використовують оператор **крапка - ".":**

<ідентікатор змінної>.<ідентифікатор поля>;

Якщо доступ потрібно організувати через вказівник на такий об’єкт, то потрібно використовувати оператор **стрілочка - “->”:**

<вказівник на структуру>-><ідентифікатор поля>;

Наприклад:

```
b.day=19; ptr->day=19; (*ptr). day=19;
```

Усі наведені оператори коректні і виконають однакову дію.

Виділення пам’яті в С. Для виділення пам’яті в С використовують такі три основні функції із **stdlib.h**

```
void *malloc(size_t size)
void *calloc(size_t num, size_t size)
void *realloc(void *ptr, size_t newsize)
```

Оскільки вони повертають **void *,** то для використання виділеної пам'яті, необхідно виконати перетворення типу як у розглянутому нижче прикладі.

Приклад коректного виділення пам'яті:

```
#include <stdlib.h>
struct addr {
   char street[40];
   char city[40];
   int bld, ap;
};
...
struct addr *get_struct(void)
{
   struct addr *p;
   if (!(p=(struct addr *)malloc(sizeof(addr)))) {
         printf("Allocation error.");
         exit(0);
   }
   return p;
}
    int main(){
    struct addr *ppp=get_struct();
    scanf(“%s”, ppp->city); …
```

Функція **free()**

void free(void *ptr)

повертає пам'ять, на яку вказує параметр **ptr**, до списку вільних областей.

Виділення пам’яті в С++. Для виділення пам'яті в С++ визначений оператор **new**. Синтаксис:

<тип>* <ідентифікатор>=new<тип>;

На відміну від **malloc** оператор **new** повертає значення потрібного типу, а не **void***. Замість функції **free** визначений оператор **delete**. Синтаксис:

delete <ідентифікатор>;

Функція із попереднього прикладу виглядатиме так:

```
addr *get_struct()
{
   addr *p;
   if (!(p= new addr)) {
         printf("Allocation error.");
         exit(0);
   }
   return p;
}
```

Для масиву структур аналогічний приклад виглядатиме так:

```
addr *get_struct(int n)
{
   addr *p;
   if (!(p= new addr[n])) {
         printf("Allocation error.");
         exit(0);
   }
   scanf("%s", p[1].city);
   return p;
}
int _tmain(int argc, _TCHAR* argv[]){
addr *ppp=get_struct_(5);
printf("%s", ppp[1].city); . . .
```

У даному випадку пам'ять було виділено під весь масив структур. Інколи потрібно працювати з масивом вказівників на структури, тоді пам'ять необхідно виділяти окремо під кожний елемент масиву. Тобто, наприклад так:

```
addr *p[10];
for(int i=0,n=5; i<n; i++)
p[i]= get_struct();
```

Відповідно і звільнення пам'яті виглядатиме по-різному: **delete** – для першого прикладу, **delete[]** – для другого, у циклі по **i** з використанням **delete p[i]** – для третього.

5.1.4. Приклад програми, що реалізує варіант0.

```
#include<stdio.h>
#define N 10
struct Settlement{
  char name[20];
  double x,y;
  };
```

```
void create(Settlement* a[], int n){
   for(int i=0; i<n; i++){
        a[i]=new Settlement;
   }
};
void inp(Settlement *a[], int n){
   for(int i=0; i<n; i++){
        printf("enter next settlement as name x y\n");
        scanf("%s%lf%lf",a[i]->name,&(a[i]->x), &(a[i]->y));
   }
};
void outp(Settlement *a[], int n){
for(int i=0; i<n; i++){
   printf("settlement %s , x=%f, y=%f\n", a[i]->name,a[i]->x,
a[i]->y);
   }
};
void outpf(Settlement *a[], int n){
   FILE *fp; //створення нового потоку
   char *fname="D://temp//test.txt";//ім'я файлу краще вводити)
//відкриття текстового файлу для запису
if((fp=fopen(fname, "w"))==NULL) {
printf ("Cannot open file %s.\n", fname);
exit(1);    }
//запис до файлу форматованої інформації
for(int i=0; i<n; i++){
   fprintf(fp,"settlement %s, x=%f, y=%f\n",
                  a[i]->name,a[i]->x,a[i]-> y);
   }
fclose(fp); // закриття файлу
};
double distance(double x1, double y1, double x2, double y2){
   double z=(x1-x2)*(x1-x2)+(y1-y2)*(y1-y2);
   return z;
   };

int count(Settlement *a[], int n, double coordx, double coordy,
double r){
   int k=0;
   for(int i=0; i<n; i++)
   if(distance(a[i]->x,a[i]->y,coordx,coordy)<r*r)k++;
   return k;
};
void del(Settlement *a[], int n){
   for(int i=0; i<n; i++) delete a[i];
 };
int main()
{
   int n;
   double xx, yy, r;
   Settlement *settlements[N];
   printf("How many settlements?\n");
   scanf("%d",&n);
   create(settlements,n);//створення масиву у динамічній пам.
   inp(settlements,n); //заповнення масиву
```

```
    outp(settlements,n); //виведення вмісту
   printf("enter the coordinates and radius you want\n");
   scanf("%lf%lf%lf", &xx, &yy, &r);
   printf("this is %d points\n", count(settlements,n,xx,yy,r));
   outpf(settlements,n); //виведення текстової інформації у файл
   del(settlements,n); //звільнення памяті
    return 0;
}
```

Відповідно до умови у програмі визначений користувацький тип – структура **Settlement**. Оголошення зроблено на зовнішньому рівні і цим типом можна користуватися у всіх функціях програми. Кожна функція виконує окрему дію, визначену пунктами умови задачі і прокоментована в прикладі. Зверніть увагу, що в С++ програмах використання ключового слова **struct** перед ідентифікатором структури обов'язкове лише для її оголошення, в подальшому для визначення змінних такого типу достатньо скористатись самим ідентифікатором. У даній програмі ідентифікатор користувацького типу заданий з великої букви, а змінні такого типу – з маленької (тут – тип **Settlement** і масив вказівників **settlements**). Це рекомендована практика в С++. Сам масив оголошений на внутрішньому рівні в **main**.

Друга особливість С++, використана в даній програмі, полягає в оголошенні і ініціалізації змінних всередині блоків, наприклад: **for(int i=0; i<n; i++)**. Про використання операторів **new** і **delete** сказано перед прикладом.

Функція **outpf**, призначена для запису інформації у текстовий файл і включає усі дії, потрібні для роботи з файлом: створення потоку для запису нового текстового файлу, зв'язування його через файлову змінну з файлом **D:/temp/test.txt**, запис туди **n** певним чином зформованих рядків, закриття файлу.

Можливий результат роботи програми на екрані:

```
How many settlements?
4
enter next settlement as name x y
Ilfracombe 2.5 4
enter next settlement as name x y
Woolacombe 1.4 2.7
enter next settlement as name x y
Croyde 1.2 1.3
enter next settlement as name x y
```

```
Brauton 2 0.9
settlement Ilfracombe , x=2.500000, y=4.000000
settlement Woolacombe , x=1.400000, y=2.700000
settlement Croyde , x=1.200000, y=1.300000
settlement Brauton , x=2.000000, y=0.900000
enter the coordinates and radius you want
2 2 1
this is 1 points
Press any key to continue . . .
```

Вміст файлу **D:/temp/test.txt**

```
settlement Ilfracombe , x=2.500000, y=4.000000
settlement Woolacombe , x=1.400000, y=2.700000
settlement Croyde , x=1.200000, y=1.300000
settlement Brauton , x=2.000000, y=0.900000
```

5.1.5. Порядок виконання роботи

Виконання роботи включає наступні етапи:

1) оголосити структурний тип відповідно варіанту;
2) виконати завдання зі створення деякого набору даних з виділенням динамічної пам’яті відповідно варіанту;
3) процес створення потрібного динамічного набору даних і усі дії з ним оформити у вигляді окремих функцій;
4) розмір набору визначити в **main()** і передавати до функцій як параметр;
5) сформований результат зберегти у файлі відповідно варіанту.

5.1.6. Варіанти завдань

Варіант 0

Оголосити структуру «Населений пункт» з такими полями: назва населеного пункту у вигляді символьного рядку; координати на мапі – дійсні числа.

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний його елемент окремо;
2. заповненням його вмісту з клавіатури;
3. виведення вмісту створеного масиву на екран;

4. підрахунок кількості населених пунктів, що знаходяться не далі деякої відстані r від точки з координатами coordx, coordy (передати як параметри);
5. Створити текстовий файл, кожний рядок якого міститиме повну інформацію про кожний з населених пунктів.

Варіант 1

Оголосити структуру«Адреса» з такими полями: назва населеного пункту та назва вулиці, реалізовані у вигляді символьних рядків; номер будинку та номер квартири – цілі числа.

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний його елемент окремо і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що визначають кількість мешканців з одного населеного пункту (назву передати як параметр);
4. додавання нового елемента після **k**-го (**k** передати як параметр)
5. поміняти місцями назви населених пунктів для **k**-го і **j**-го елементів (**k** і **j** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме таку інформацію про мешканців заданого населеного пункту: прізвище, вулиця.

Варіант 2

Оголосити структуру «Шахова фігура» з полями: назва як символьний рядок і позиція на шахівниці як ціле і символ.

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний елемент масиву і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, назва яких передана як параметр;
4. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)

5. видалення **k**-го елемента (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме інформацію про одну фігуру на шахівниці у вигляді **nnnn: x, y,** а записи будуть розташовані у порядку їх розташування в масиві.

Варіант 3

Оголосити структуру «Позиція на шахівниці» поля з цілими координатами, значення яких від одиниці до восьми і третє поле з назвою кольору фігури.

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний елемент масиву і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що розташовані на одній половині поля;
4. додавання нового елемента після **k**-го (**k** передати як параметр)
5. поставити **k**-й елемент перед **j**-м (**k** і **j** передати як параметр)

Створити текстовий файл, який міститиме єдиний рядок з інформацією про координати білих фігур так, що координати однієї фігури будуть знаходитись у дужках і через двокрапку. А самі дужки будуть розділені комами.

Варіант 4

Оголосити структуру «Дата».

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний елемент масиву і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. впорядкування елементів за значенням поля «рік»;
4. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)
5. видалення усіх елементів, що визначають дату після заданої (це значення передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме інформацію про зимові дати, розташовані у тому же порядку, що і у масиві, у вигляді **dd – mm – yy.**

Варіант 5

Оголосити структуру «Нота»: назва ноти та назва октави у вигляді символьних рядків довільної довжини та числове поле із значенням тривалості звучання.

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний елемент масиву і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що відповідають нотам першої октави;
4. додавання нового елемента після **k**-го (**k** передати як параметр)
5. видалення елемента перед **k**-тим (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме інформацію з назвою однієї ноти, причому у зворотній відносно їх розташування у масиві послідовності.

Варіант 6

Оголосити структуру «Виробник» з реалізованими у вигляді символьних рядків фіксованої довжини полів з назвою фірми, країною реєстрації та контактним телефоном.

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний елемент масиву і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що відповідають виробникам із однієї країни, назву передати у функцію як параметр;
4. впорядкування елементів за назвою фірми;
5. додавання нового елемента після **k**-го (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме інформацію про одного виробника, а усі рядки відповідатимуть упорядкованому масиву.

Варіант 7

Оголосити структуру «Час».

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний елемент масиву і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що відповідають періоду від 0-00 до 6-00
4. впорядкування елементів за значенням поля «година»;
5. додавання нового елемента перед **k**-тим (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме інформацію про кількість секунд після опівночі для відповідного значення часу із масиву.

Варіант 8

Оголосити структуру «Постачальник», поля з назвою фірми, країною реєстрації та контактним телефоном у вигляді символьних рядків фіксованої довжини.

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний елемент масиву і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. видалення усіх елементів, що відповідають постачальникам із заданої країни, назву країни передавати як параметр;
4. додавання нового елемента перед **k**-тим (**k** передати як параметр)
5. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)
6. видалення елемента після **k**-го (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме інформацію про постачальників із заданої країни як (назва фірми, телефон).

Варіант 9

Оголосити структуру «Адреса»: назва населеного пункту та назва вулиці – символьні рядки; номер будинку – ціле число.

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний елемент масиву і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що відповідають заданій вулиці;
4. додавання нового елемента після **k**-го (**k** передати як параметр);
5. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр).

Створити текстовий файл, кожний рядок якого міститиме інформацію про одну адресу, а всього до файлу записати **n** адрес, **n** ввести з клавіатури.

Варіант 10

Оголосити структуру «Кухар» з з ім’ям та прізвищем, реалізовані у вигляді символьних рядків.

Оголосити масив вказівників на таку структуру.

Передбачити такі функції:

1. створення масиву з виділенням пам'яті під кожний елемент масиву і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. впорядкування елементів за прізвищем;
4. додавання нового елемента після **k**-го (**k** передати як параметр)
5. видалення елемента після **k**-го (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме інформацію про одного кухаря із впорядкованого масиву.

Варіант 11

Оголосити структуру «Людина» (прізвище, телефон – символьні рядки).

Оголосити вказівник на динамічний масив змінних такого структурного типу.

Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості людей, код міста у яких відповідає заданому (передати як параметр);
4. впорядкування елементів за телефонним номером;
5. додавання нового елемента перед **k**-тим (**k** передати як параметр)
6. видалення елемента після **k**-го (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме повну інформацію про людину із впорядкованого масиву.

Варіант 12

Оголосити структуру «Планета» з полями назва (рядок) і відстань від Землі (ціле).

Оголосити вказівник на динамічний масив змінних такого структурного типу.

Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що знаходяться від Землі не далі заданої відстані (передати як параметр);
4. впорядкування елементів за зменшенням відстані від Землі;
5. додавання нового елемента після **k**-го (**k** передати як параметр)
6. поміняти місцями назви **k**-ої і **j**-ої планети (**k** і **j** передати як параметр)
7. видалення елемента після **k**-го (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме повну інформацію про відповідну планету із невпорядкованого масиву.

Варіант 13

Оголосити структуру «Станція»: назва (рядок), розташування (координати на карті).

Оголосити вказівник на динамічний масив змінних такого структурного типу.

Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що розташовані не далі заданої відстані від початку відліку;
4. додавання нового елемента після **k**-го (**k** передати як параметр)
5. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)
6. видалення елемента перед **k**-тим (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме назву відповідної станції у порядку розташування у масиві.

Варіант 14

Оголосити структуру «Координати точки у просторі».

Оголосити вказівник на динамічний масив змінних такого структурного типу.

Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що віддалені більше, ніж на задану параметром відстань від початку координат;
4. впорядкування елементів за відстанню від початку координат;
5. додавання нового елемента після **k**-го (**k** передати як параметр)
6. видалення елемента після **k**-го (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме порядковий номер точки і її координати через кому, інформація впорядкована.

Варіант 15

Оголосити структуру «Розташування», поля – координати на площині і номер об’єкту.

Оголосити вказівник на динамічний масив змінних такого структурного типу.

Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;

2. виведення вмісту створеного масиву на екран;
3. впорядкування елементів за номером;
4. додавання нового елемента після елемента з заданими координатами (**x, y** передати як параметри)
5. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент масиву. Передбачити функцію для заповнення цього файлу із масиву.

Варіант 16

Оголосити структуру «Виконавець», поля – і'мя та прізвище, символьні рядки.
Оголосити вказівник на динамічний масив змінних такого структурного типу.
Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. впорядкування елементів за прізвищем;
4. додавання нового елемента після **k**-го (**k** передати як параметр);
5. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр);
6. видалення елементів із заданим прізвищем (передати як параметр).

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент масиву. Передбачити функцію для заповнення цього файлу із масиву.

Варіант 17

Оголосити структуру «Товар» (назва, кількість, ціна).
Оголосити вказівник на динамічний масив змінних такого структурного типу.
Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, дорожчих за задану ціну (передати як параметр);
4. впорядкування елементів за значенням поля ціна;
5. додавання нового елемента після **k**-го (**k** передати як параметр)

6. видалення елемента після **k**-го (**k** передати як параметр)

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент масиву. Передбачити функцію для заповнення цього файлу із масиву.

Варіант 18

Оголосити структуру «Алфавіт», «букви» – як слово, складене з усіх букв даного алфавіту у лексикографічному порядку, аналогічно побудоване поле «знаки» – із усіх припустимих знаків, і два числових поля з інформацією про кількість букв і спеціальних символів даного алфавіту.

Оголосити вказівник на динамічний масив змінних такого структурного типу.

Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, алфавіт яких складений із заданої кількості символів (передати як параметр);
4. впорядкування елементів за кількістю букв;
5. додавання нового елемента після **k**-го (**k** передати як параметр)
6. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент масиву. Передбачити функцію для заповнення цього файлу із масиву.

Варіант 19

Оголосити структуру «Система числення», поля – «цифри» – як речення, складене з усіх цифр, припустимих у даній системі числення через пропуски та числове поле із значенням основи даної системи числення

Оголосити вказівник на динамічний масив змінних такого структурного типу.

Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. впорядкування елементів за значенням основи;
4. додавання нового елемента після **k**-го (**k** передати як параметр)

5. видалення елемента після **k**-го (**k** передати як параметр)

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент масиву. Передбачити функцію для заповнення цього файлу із масиву.

Варіант 20

Оголосити структуру «Коло» - координати центру, радіус, ознака візуалізації.

Оголосити вказівник на динамічний масив змінних такого структурного типу.

Передбачити такі функції:

1. створення масиву потрібного розміру з виділенням пам'яті і заповненням його вмісту з клавіатури;
2. виведення вмісту створеного масиву на екран;
3. підрахунок кількості елементів, що мають даний радіус і є видимими, задати як параметри;
4. впорядкування елементів за значенням поля радіус;
5. додавання нового елемента після **k**-го (**k** передати як параметр)
6. видалення елемента після **k**-го (**k** передати як параметр)

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент масиву. Передбачити функцію для заповнення цього файлу із масиву.

Варіант 21

Оголосити структурний тип «Автор» - прізвище і гонорар.

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу до початку списку із заповненням вмісту цього елементу з клавіатури;
2. створення списку додаванням елементу в початок;
3. редагування **k**-го елемента списку (**k** передати як параметр)
4. виведення вмісту створеного списку на екран;
5. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)
6. видалення елемента перед **k**-м (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме повну інформацію про чергового автора із списку.

Варіант 22

Оголосити структурний тип «Обладнання», поля – назва, дата поставки – як символьні рядки, вартість – дійсне.

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу до початку списку із заповненням вмісту цього елементу з клавіатури;
2. створення списку додаванням елементу в початок;
3. виведення вмісту створеного списку на екран;
4. підрахунок кількості елементів з однією датою поставки;
5. видалення елементів з вартістю більше заданої;
6. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме повну інформацію про одиницю обладнання із даного списку.

Варіант 23

Оголосити структурний тип «Товар» – назва, ціна, фірма – постачальник.

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу у кінець списку із заповненням вмісту цього елементу з клавіатури;
2. створення списку додаванням елементу в кінець;
3. редагування **k**-го елемента списку (**k** передати як параметр)
4. виведення вмісту створеного списку на екран;
5. підрахунок кількості товарів заданого постачальника (передати як параметр);
6. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме повну інформацію про кожний із **n** перших товарів (**n** ввести з клавіатури).

Варіант 24

Оголосити структурний тип «Банк», назва (символьний рядок), МФО (число).

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу до початку списку і заповненням вмісту цього елементу з клавіатури;
2. редагування **k**-го елемента списку (**k** передати як параметр)
3. виведення вмісту створеного списку на екран;
4. впорядкування елементів за значенням поля МФО;
5. додавання нового елемента перед **k**-тим (**k** передати як параметр)
6. видалення елемента перед **k**-м (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме повну інформацію про відповідний елемент із списку. Файл містить інформацію про усі елементи списку.

Варіант 25

Оголосити структурний тип «Підключення», поля – назва мережі, наявність пільги.

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу у кінець списку із заповненням вмісту цього елементу з клавіатури;
2. виведення вмісту створеного списку на екран;
3. підрахунок кількості елементів заданої мережі (передати як параметр);
4. додавання нового елемента після **k**-го (**k** передати як параметр)
5. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)
6. видалення елемента перед **k**-м (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме повну інформацію про відповідний елемент із списку.

Варіант 26

Оголосити структурний тип «Персона», поля – прізвище, ім'я, телефон – усі рядки.

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу до початку списку і заповненням вмісту цього елементу з клавіатури;
2. додавання чергового елементу у кінець списку із заповненням вмісту цього елементу з клавіатури;
3. виведення вмісту створеного списку на екран;
4. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)
5. видалення елемента перед **k**-м (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме повну інформацію про відповідний елемент списку.

Варіант 27

Оголосити структурний тип «Власник», поля – прізвище, телефон, рік народження.

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу до початку списку і заповненням вмісту цього елементу з клавіатури;
2. виведення вмісту створеного списку на екран;
3. впорядкування елементів за зростанням віку;
4. додавання нового елемента перед **k**-тим (**k** передати як параметр)
5. видалення елемента перед **k**-м (**k** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме повну інформацію про черговий елемент списку.

Варіант 28

Оголосити структурний тип «Опис», поля – назва країни, назва кліматичного пояса, категорія (номер).

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу у кінець списку із заповненням вмісту цього елементу з клавіатури;
2. виведення вмісту створеного списку на екран;

3. підрахунок кількості елементів, що відповідають заданій категорії та поясу (передати як параметри);
4. додавання нового елемента після **k**-го (**k** передати як параметр);
5. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)

Створити текстовий файл, кожний рядок якого міститиме міститиме повну інформацію про черговий елемент списку.

Варіант 29

Оголосити структурний тип «Організатор», поля – назва фірми, телефон, наявність пільги.

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу до початку списку і заповненням вмісту цього елементу з клавіатури;
2. виведення вмісту створеного списку на екран;
3. підрахунок елементів із заданою назвою фірми (передати як параметр);
4. видалити усі елементи за назвою фірми (передати як параметр);
5. додавання нового елемента після **k**-го (**k** передати як параметр)

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент списку. Передбачити функцію для заповнення цього файлу із списку.

Варіант 30

Оголосити структурний тип «Видавець», назва фірми, адреса, контактний телефон.

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу до початку списку і заповненням вмісту цього елементу з клавіатури;
2. редагування **k**-го елемента списку (**k** передати як параметр)
3. виведення вмісту створеного списку на екран;
4. додавання нового елемента після **k**-го (**k** передати як параметр)
5. поміняти місцями **k**-й і **j**-й елементи (**k** і **j** передати як параметр)

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент списку. Передбачити функцію для заповнення цього файлу із списку.

Варіант 31

Оголосити структурний тип «Майстерня», поля – назва, години роботи (два цілих числа – початок і кінець).

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу до початку списку і заповненням вмісту цього елементу з клавіатури;
2. додавання чергового елементу у кінець списку із заповненням вмісту цього елементу з клавіатури;
3. виведення вмісту створеного списку на екран;
4. додавання нового елемента перед **k**-тим (**k** передати як параметр);
5. видалення елемента перед **k**-м (**k** передати як параметр);

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент списку. Передбачити функцію для заповнення цього файлу із списку.

Варіант 32

Оголосити структурний тип «Експонат», поля – назва, три габаритні розміри.

Створити однозв'язний список із змінних такого структурного типу.

Передбачити такі функції:

1. додавання чергового елементу до початку списку і заповненням вмісту цього елементу з клавіатури;
2. редагування **k**-го елемента списку (**k** передати як параметр)
3. виведення вмісту створеного списку на екран;
4. впорядкування елементів за максимальним габаритним розміром;
5. видалення усіх елементів після **k**-го (**k** передати як параметр)

Створити двійковий файл, **k**-й запис якого міститиме інформацію про **k**-й елемент списку. Передбачити функцію для заповнення цього файлу із списку.

5.1.7. Запитання до самоконтролю

1. Що таке структура? Які діють обмеження на типи полів структур? Чи можуть вони бути однотипними?
2. Особливості використання структурного типу в С і С++.
3. Як звернутися до поля структури? Які дії можна виконувати над структурними змінними?
4. Що таке динамічна пам'ять? Які переваги використання динамічної пам'яті?
5. Особливості виділення динамічної пам'яті в С і в С++. Чим відрізняється оголошення і використання динамічного масиву і масиву вказівників на динамічні об’єкти?
6. Які складові елементу однозв’язного списку (вузла)? Що таке голова списку? Як визначити хвіст списку?
7. Як додати вузол однозв’язного списку у початок списку? У хвіст списку? Перед заданим елементом списку?

5.2. Комп’ютерний практикум № 2.

5.2.1 Тема

Основні конструкції мови програмування С. Реалізація розгалужень

5.2.2 Мета роботи.

Навчитися правильно описувати абстракції предметної області засобами С++ з використанням принципу інкапсуляції.

5.2.3 Основні теоретичні відомості

Об’єктна декомпозиція, абстракція, опис абстракції. В основі об’єктно-орієнтованого підходу до програмування лежить об’єктна декомпозиція: система розділена на елементи, кожний з яких належить до певної абстракції предметної області і характеризується своєю власною поведінкою, а поведінка системи забезпечується взаємодією цих автономних елементів.

У мовах програмування конкретним виявом певної абстракції є тип. В об’єктно-орієнтованих мовах існують механізми, що дозволяють створювати нові типи – типи користувача. В С++, наприклад, це класи.

Синтаксис опису класу. Конструкція **`class x{ ... };`** утворює новий тип і називається визначенням класу.

Повний опис класу складається з трьох частин: оголошень полів даних, прототипів функцій-членів, визначень функцій-членів. Наприклад:

```
class Point                    // опис класу
{
 int x,y;                      //данні стану
public:
void          SetX(int);      //прототип функції-члена
//прототип і визначення       функції-члена всередині классу
int GetX(){return x;};
};
    //визначення функції-     члена за межами класу
void Point:: SetX(int a){     x = a;  };
    void main()
    {
    Point a, b. . .;  }          //оголошення об’єктів
```

Поля даних – екземпляри змінних стану, які характеризують об'єкт класу. Оголошуються всередині опису класу і усі екземпляри об'єктів даного класу будуть мати однакову множину змінних стану.

Функції-члени – це методи, які обробляють повідомлення, що передаються об'єктам класів (визначають відповідальність абстракції). Прототипи (оголошення) цих функцій повинні знаходитись всередині опису класу, визначення (опис) їх може бути і всередині і за межами класу. Якщо опис функцій дається за межами класу, необхідно вказувати ім'я класу разом з оператором дозволу області видимості **::,** оскільки різні класи можуть мати функції-члени з однаковими назвами.

В оголошенні класу можуть використовуватись помітки **public, private** та **protected**, які визначають права доступу до відповідних розділів тіла класу (відкритий, закритий, захищений). Порядок розташування та кількість цих поміток в описі класу – довільна. За умовчанням тип доступу **private**.

Відкритий **(public)** розділ опису класу утворює відкритий інтерфейс для його об'єктів: значення закритих даних класу можна змінити тільки скориставшись відкритими функціями-членами цього класу (змінити свій стан об'єкт може тільки опрацювавши допустиме для нього повідомлення – тим методом, який оголошений у відкритій частині класу).

Оскільки клас **(class)** уводить новий тип даних (у наведеному прикладі **Point**), оголошення об'єкту цього класу **(a,b)-** це оголошення змінних такого типу **(Point),** аналогічне декларуванню змінних стандартних типів перед їх використанням.

Класифікація функцій – членів. Кожний клас має свій набір функцій-членів (методів), які характеризують поведінку об'єктів цього класу. Усі ці функції можна умовно віднести до таких категорій:

- Конструктори – відповідають за створення та ініціалізацію об'єкта;
- деструктор – руйнує об'єкт;
- конструктори копіювання – для копіювання об'єктів;

- селектори – для доступу до закритих (або захищених) частин об’єкта без зміни їх значень;
- модифікатори – для зміни стану об’єкта.

За умови відсутності функцій-членів для нового класу, компілятор надає йому чотири методи, які називаються канонічними методами. Це конструктор умовчання, конструктор копій, деструктор, оператор надання (присвоювання).

Назва конструктора збігається з назвою класу, вказувати тип значення, що повертається не потрібно. В класі може бути декілька конструкторів, вони повинні розрізнятися тільки кількістю та типами параметрів. Наприклад, у розглянутому раніше класі **Point** можуть бути такі конструктори:

```
Point();
Point(int);
Point(int, int);
Point(char*);
```

Якщо об’єкт потрібно ініціалізувати як копію існуючого, використовують конструктор копіювання. Для класу **Point** конструктор копіювання матиме наступний вигляд:

```
Point  :: Point ( const Point  & )
```

Конструктори ініціалізують об’єкт, інколи з захопленням певних ресурсів (пам’ять, файл), які потрібно звільнити після їх використання. Функція, яка гарантує правильне знищення об’єкту називається деструктором.

Назва деструктора збігається з іменем класу, перед яким ставлять «тильду», наприклад:

```
~Point();
```

Функції-селектори призначені для перевірки стану об’єкта, вони повертають значення полів, розташованих в захищених або закритих частинах класу. Наприклад,

```
int GetX(){return x;};
```

Функції-модифікатори призначені для зміни стану об’єкта, вони можуть не повертати значення (`void`) або повертати посилання на модифікований об’єкт, наприклад:

```
Point& SetX(int a) { x=a;   return *this;};
```

Запис ***this** означає об’єкт, для якого викликана функція-член.

Статичні данні-члени – глобальні змінні, які діють в межах одного класу. Існує лише одна копія такої змінної, яка належить класу (кожний об'єкт має власну копію нестатичного поля). Приклад оголошення:

```
class Point
{ . . . static int count;. . .};. . .
//обов'язкова  ініціалізація статичної змінної
int Point:: count=0;
```

Статичні функції-члени працюють тільки з членами класу, а не конкретного об'єкта класу.

Доступ до статичних членів класу відбувається так само, як і до звичайних нестатичних. Крім того, замість ідентифікатора об'єкта можна використати ім'я самого класу, наприклад: **`Point:: set_count(5);`**

`Inline`–функцій – це маленькі функції, звернення до яких компілятор перетворює на їх код, що значно підвищує ефективність роботи програми. Такі функції можна повністю описати в середині опису класу, наприклад

```
int GetX(){return x;};
```

або визначити за межами опису класу з ключовим словом `inline`, наприклад

```
inline int Date::get_year(){ return y;}
```

5.2.4 Приклад програми

Реалізувати задачу із варіанту 0. Зверніть увагу, що даний варіант передбачає реалізацію і тестування одного класу, у той час як у звичайних варіантах потрібно реалізувати стільки класів, скільки визначено у конкретному завданні. У більшості варіантів у другому класі присутнє поле, що має тип, визначений першим класом.

```
#include<string.h>
#include<stdio.h>
#include<math.h>
using namespace std;
class Settlement{
char *name;
double x,y;
public:
     Settlement();
     Settlement(char*, double, double);
     Settlement(Settlement&);
```

```
     ~Settlement();
     Settlement& setname(char*);
     Settlement& setx(double);
     Settlement& sety(double);
     double getx();
     double gety();
     char* getname();
     void print();
     double mydist(double, double);
     };
Settlement::Settlement(){
          name=new char[8];
          strcpy(name, "noname");
          x=y=0;
     }
Settlement::Settlement(char*title, double xx, double yy){
          name=new char[strlen(title)+1];
          strcpy(name, title);
          x=xx;y=yy;
     }
Settlement::Settlement(Settlement&a){
     x=a.x; y=a.y;
     name=new char[strlen(a.name)+1];
     strcpy(name, a.name);
};
Settlement::~Settlement(){if(name)delete[] name; };;
Settlement& Settlement::setname(char*title){
          delete[] name;
          name=new char[strlen(title)+1];
          strcpy(name, title);
          return *this;
     };
Settlement& Settlement::setx(double xx){x=xx;return *this;};
Settlement& Settlement::sety(double yy){y=yy;return *this;};
double Settlement::getx(){return x;};
double Settlement::gety(){return y;};
char*Settlement::getname(){return name;};
void Settlement::print(){
printf("settlement %s , x=%f, y=%f\n", name, x, y);
};
double Settlement:: mydist(double x0, double y0){
     return sqrt((x-x0)*(x-x0)+(y-y0)*(y-y0));
};

int _tmain(int argc, _TCHAR* argv[])
{
     double coordx, coordy, xx, yy, r;
     char title[20];
     Settlement obj1;
     printf("enter settlement as name x y\n");
     scanf("%s%lf%lf",title,&coordx, &coordy);
     Settlement obj2(title, coordx, coordy);
     Settlement obj3=obj2;
     printf("This is obj1, obj2, obj3 before change\n");
     obj1.print();
```

```
     obj2.print();
     obj3.print();
     printf("Enter new name and coordinates for obj3\n");
     scanf("%s%lf%lf",title,&coordx, &coordy);
     obj3.setname(title).setx(coordx).sety(coordy);
     printf("This is obj3 after change\n");
     obj3.print();
     printf("Enter the coordinates to search for a distance from
obj3\n");
     scanf("%lf%lf", &xx, &yy);
     printf("this is %f \n", obj3.mydist( xx, yy)) ;
          return 0;
}
```

Можливий результат роботи виглядатиме так:

```
enter settlement as name x y
first 5.5 6.6
This is obj1, obj2, obj3 before change
settlement noname , x=0.000000, y=0.000000
settlement first , x=5.500000, y=6.600000
settlement first , x=5.500000, y=6.600000
Enter new name and coordinates for obj3
second 7.7 4.4
This is obj3 after change
settlement second , x=7.700000, y=4.400000
Enter the coordinates to search for a distance from obj3
7.7 7.7
this is 3.300000
Press any key to continue . . .
```

5.2.5 Порядок виконання роботи

Спільні вимоги до усіх варіантів.

Побудувати конкретний клас відповідно варіанту з врахуванням необхідності приховання даних. Якщо не оговорено окремо, у кожному класі, що розробляється, передбачити:

- конструктори умовчання, з параметрами та копіювання;
- за наявності динамічних полів, у тому числі символьних рядків довільної довжини, у конструкторах передбачити виділення пам’яті;
- деструктор;
- функції-селектори для доступу окремо до кожного поля;
- дві функції для виведення на екран інформації про поточний стан об’єкту; одна – у вигляді рядка зі значеннями кожного з полів через пропуск, друга – у вигляді назви класу і значення одного з полів, яке найкраще ідентифікуватиме об’єкт; якщо клас містить масив, інформація

про кожний елемент масиву подається окремим рядком у першій з функцій;

- функції-модифікатори окремо для кожного з полів з повертанням посилання на модифікований об’єкт.

У тестовому прикладі створити три об’єкти заданого відповідно варіанту типу (з використанням кожного з визначених конструкторів). Для створення об’єкту з використанням конструктора з параметрами, данні для нього уводити з клавіатури.

5.2.6 Варіанти завдань

Варіант 0.

Тип «Населений пункт» визначити як клас, що містить:

- закриті поля з назвою населеного пункту у вигляді символьного рядка довільної довжини; та його координатами на мапі, реалізовані як дійсні числа;
- реалізувати потрібні відповідно загальним вимогам функції і додати функцію визначення відстані від точки з заданими координатими.

У тестовому прикладі створити об’єкти типу «Населений пункт» відповідно загальним вимогам, вивести інформацію про них, перевірити роботу кожної з функцій.

Варіант1.

Тип «Адреса» визначити як клас, що містить:

- закриті поля з назвою населеного пункту та назвою вулиці, реалізовані у вигляді символьних рядків довільної довжини; номер будинку та номер квартири – цілі числа;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Людина» визначити як клас, що містить:

- закриті поля з ім’ям та прізвищем, реалізовані у вигляді символьних рядків довільної довжини; телефон, що складатиметься з двох числових

полів: код міста та внутрішнього номера; адреса як об’єкт відповідного типу;

- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Людина» та «Адреса» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант2.

Тип «Позиція на шахівниці» визначити як клас, що містить:

- закриті поля з координатами точки, значення яких відповідатимуть нумерації шахового поля;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Фігура» визначити як клас, що містить:

- закриті поля з позицією на шахівниці відповідного типу та реалізоване у вигляді символьного рядка довільної довжини поле з назвою кольору;
- передбачити функцію «хід» на одну позицію в одному з чотирьох напрямків;
- решту функцій реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Фігура» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант3.

Тип «Позиція на шахівниці» визначити як клас, що містить:

закриті поля з цілими координатами, значення яких від одиниці до восьми;

- усі функції реалізувати відповідно загальним вимогам.

Тип «Фігура» визначити як клас, що містить:

- закриті поля з позицією на шахівниці як для гри у шашки (як об’єкт відповідного типу) та реалізоване у вигляді символьного рядка довільної довжини поле з назвою кольору;
- передбачити функцію «хід» на одну позицію в одному з двох напрямків;
- решту функцій реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Фігура» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант4.

Тип «Дата» визначити як клас, що містить закриті числові поля з днем, місяцем, роком та реалізовані відповідно загальним вимогам функції.

Тип «Точка на площині» визначити як клас, що містить:

- закриті числові поля з координатами та порядковим номером;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Точка на площині» та «Дата» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант5.

Тип «Нота» визначити як клас, що містить:

- реалізовані у вигляді символьного рядка довільної довжини закриті полі з назвою ноти та назвою октави та закрите числові поля із значенням тривалості звучання;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Нота» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант6.

Тип «Виробник» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закриті поля з назвою фірми, країною реєстрації та контактним телефоном;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Іграшка» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою та закриті числові поля із ціною та кількістю на складі та поле з інформацією про виробника як об’єкт відповідного типу;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Іграшка» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант7.

Тип «Час» визначити як клас, що містить:

- закриті числові поля з годинами, хвилинами та секундами;
- передбачити функцію «плюс секунда» як збільшення поточного часу на одну секунду;
- решту функцій реалізувати відповідно загальним вимогам.

Тип «Годинник» визначити як клас, що містить:

- реалізовані у вигляді символьного рядка довільної довжини закрите поле з назвою фірми виробника та закриті поля для визначення стилю відображення (військовий формат – 24 години чи стандартний – 12), поле для визначення часу як об’єкт відповідного типу;
- функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Час» та «Годинник» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант8.

Тип «Постачальник» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закриті поля з назвою фірми, країною реєстрації та контактним телефоном;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Товар» визначити як клас, що містить:

- реалізовані у вигляді символьного рядка довільної довжини закрите поле з назвою товару; закриті числові поля із ціною та кількістю у даній партії; постачальника визначити як об’єкт відповідного типу;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Товар» та «Постачальник» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант9.

Тип «Адреса» визначити як клас, що містить:

- закриті поля з назвою населеного пункту та назвою вулиці, реалізовані у вигляді символьних рядків довільної довжини; номер будинку – ціле число;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Експонат» визначити як клас, що містить:

- закрите поле з назвою, реалізоване як символьний рядок та числові поля з габаритними розмірами, а також реалізоване як об’єкт відповідного типу поле з адресою;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Адреса» та «Експонат» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант10.

Тип «Кухар» визначити як клас, що містить:

- закриті поля з ім’ям та прізвищем, реалізовані у вигляді символьних рядків довільної довжини;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Страва» визначити як клас, що містить:

- закриті поля типу символьного рядка з назвою, ціле з вартістю, та типу «Кухар» з інформацією про кухаря;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Кухар» та «Страва» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант11.

Тип «Людина» визначити як клас, що містить:

- закриті поля з ім’ям та прізвищем, реалізовані у вигляді символьних рядків довільної довжини; телефон, що складатиметься з двох числових полів: код міста та внутрішнього номера;

- усі функції реалізувати відповідно загальним вимогам.

Тип «Учасник змагань» визначити як клас, що містить:

- реалізовані у вигляді символьних рядків довільної довжини закриті поля з назвами країни і виду спорту; поле типу «Людина» з інформацією про тренера, цілий номер учасника.
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Людина» та «Учасник змагань» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант12.

Тип «Дата» визначити як клас з трьома цілими полями і з реалізованими відповідно загальним вимога функціями.

Тип «Адреса» визначити як клас, що містить:

- закриті поля з назвою населеного пункту та назвою вулиці, реалізовані у вигляді символьних рядків довільної довжини; номер будинку та квартири – цілі числа;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Поштове відправлення» визначити як клас, що містить:

- поля з адресою та датою отримання – відповідних типів, дійсне поле з вартістю доставки;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти усіх трьох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант13.

Тип «Розташування» визначити як клас, що містить:

- закриті числові поля з двома координатами та порядковим номером;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Станція» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле із назвою та закрите поле з інформацією про розташування як об’єкт відповідного типу;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Розташування» та «Станція» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді

Варіант14.

Тип «Координати» визначити як клас, що містить закриті числові поля з двома координатами, усі функції реалізувати відповідно загальним вимогам.

Тип «Істота» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле із назвою, закриті числове поле з віком та поле з координатами як об’єкт відповідного типу;
- передбачити функцію «рух» як збільшення поточних значень координат на одиницю;
- решту функцій реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Істота» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант15.

Тип «Розташування» визначити як клас, що містить:

- закриті числові поля з двома координатами та порядковим номером;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Кліматичні умови» визначити як клас, що містить:

- закриті числові поля із значеннями температури, освітленості, вологості та кислотності грунту;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Кліматичні умови» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант16.

Тип «Виконавець» визначити як клас, що містить:

- закриті поля з ім'ям та прізвищем, реалізовані у вигляді символьних рядків довільної довжини;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Твір» визначити як клас, що містить:

- закрите поле з назвою, реалізоване у вигляді символьного рядка довільної довжини; два закритих числових поля – розмір і гонорар;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об'єкти типу «Твір» та «Виконавець» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант17.

Тип «Розташування» визначити як клас, що містить:

- закриті числові поля з двома координатами та категорією приміщення;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Товар на складі» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою товару, закриті поля з інформацією про розташування як об'єкт відповідного типу, ціну та кількість одиниць у даній партії;
- визначити функцію підрахунку загальної вартості партії товару;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об'єкти типу «Розташування» та «Товар» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант18.

Тип «Алфавіт» визначити як клас, що містить:

- приховані поля: «букви» – як слово, складене з усіх букв даного алфавіту у лексикографічному порядку, аналогічно побудоване поле «знаки» – із усіх припустимих знаків, і два числових поля з інформацією про кількість букв і спеціальних символів даного алфавіту;

- усі функції реалізувати відповідно загальним вимогам.

Тип «Фраза» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини приховане поле із вмістом фрази та приховане поле типу «Алфавіт»;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Алфавіт» та «Фраза» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант19.

Тип «Система числення» визначити як клас, що містить:

- приховане поле «цифри» – як речення, складене з усіх цифр, припустимих у даній системі числення через пропуски та приховане числове поле із значенням основи даної системи числення;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Число» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини приховане поле із значенням числа та приховане поле з інформацією про застосовану систему числення як об’єкт відповідного типу;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типів «Система числення» та «Число» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант20.

Тип «Координати» визначити як клас, що містить закриті числові поля з двома координатами, усі функції реалізувати відповідно загальним вимогам.

Тип «Коло» визначити як клас, що містить:

- закрите поле з координатами як об’єкт відповідного типу, закриті числові поля з радіусом та ознакою візуалізації;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Координати» та «Коло» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант21.

Тип «Автор» визначити як клас, що містить:

- закриті поля з ім’ям та прізвищем, реалізовані у вигляді символьних рядків довільної довжини;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Книга» визначити як клас, що містить:

- реалізовані у вигляді символьних рядків довільної довжини закриті поля із назвою і кодом УДК, поле з інформацією про автора як об’єкт відповідного типу, числові поля з роком видання та кількістю сторінок;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Книга» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант22.

Тип «Постачальник» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закриті поля з назвою фірми, країною реєстрації та контактним телефоном;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Обладнання» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою, закриті числові поля з вартістю, датою виготовлення та номером групи визначення терміну амортизації (5, 15, 25), поле з інформацією про постачальника як об’єкт відповідного типу;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Модернізація» визначити як клас, що містить:

- закриті числові поля з вартістю та датою ремонту;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти усіх типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант23.

Тип «Виробник» визначити як клас, що містить:

- реалізовані у вигляді символьних рядків довільної довжини закрите поле з назвою фірми, адресою та контактним телефоном, числове поле з періодом гарантійного обслуговування;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Товар» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою, числове поле з вартістю та поле з інформацією про виробника як об’єкт відповідного типу;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти обох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант24.

Тип «Банк» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою банку та числове поле з його МФО;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Рахунок» визначити як клас, що містить:

- закрите поле з інформацією про банк як об’єкт відповідного типу, закриті числові поля з номером та вмістом рахунку;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти обох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант25.

Тип «Дата» визначити як клас, що містить закриті числові поля з днем, місяцем, роком та реалізовані відповідно загальним вимогам функції.

Тип «Підключення» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою мережі та числове поле, що визначатиме наявність пільги;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти обох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант26.

Тип «Дата» визначити як клас, що містить закриті числові поля з днем, місяцем, роком та реалізовані відповідно загальним вимогам функції.

Тип «Персона» визначити як клас, що містить:

- реалізовані у вигляді символьних рядків довільної довжини закриті поля з прізвищем, ім’ям та телефоном людини, датою народження, реалізованою як об’єкт відповідного типу;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти обох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант27.

Тип «Власник» визначити як клас, що містить

- реалізовані у вигляді символьних рядків довільної довжини закриті поля з прізвищем, ім’ям та телефоном;
- реалізовані відповідно загальним вимогам функції.

Тип «Земельна ділянка» визначити як клас, що містить:

- закрите поле з інформацією про власника, реалізоване як об’єкт відповідного типу; реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою населеного пункту; числові поля, що визначають розташування ділянки;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти обох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант28.

Тип «Опис» визначити як клас, що містить

- реалізовані у вигляді символьних рядків довільної довжини закриті поля з назвою країни походження, кліматичного пояса та біологічною характеристикою (вид, рід, родина);
- усі функції реалізувати відповідно загальним вимогам.

Тип «Тварина» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини приховане поле з назвою, приховане поле опису як об'єкт відповідного типу та числове поле, що визначатиме вік;
- передбачити функцію, що додає рік до віку тварини;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об'єкти обох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант29.

Тип «Організатор» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою фірми, контактним телефоном та адресою та числове поле, що визначатиме наявність пільги;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Екскурсія» визначити як клас, що містить:

- закриті поля з назвою та місцем проведення екскурсії, реалізовані у вигляді символьного рядка;
- закриті поля типу «Організатор» та цілі – вартість та наявність пільг;
- функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об'єкти обох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант30.

Тип «Видавець» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою фірми, контактним телефоном та адресою;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Видання» визначити як клас, що містить:

- приховане поле з назвою, реалізоване у вигляді символьного рядка;
- приховане поле з інформацією про видавця як об’єкт відповідного типу;
- приховані числові поля з вартістю, кількістю сторінок та кодом використаного паперу;
- функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти обох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант31.

Тип «Адреса» визначити як клас, що містить:

- закриті поля з назвою населеного пункту та назвою вулиці, реалізовані у вигляді символьних рядків довільної довжини; номер будинку як ціле число;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Майстерня» визначити як клас, що містить:

- реалізоване у вигляді символьного рядка довільної довжини закрите поле з назвою фірми, контактним телефоном та адресою як об’єкт відповідного типу;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти обох типів відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

Варіант32.

Тип «Виставковий центр» визначити як клас, що містить:

- закриті поля з власною назвою центру та назвою вулиці, де він розташований, реалізовані у вигляді символьних рядків довільної довжини; номер будинку – ціле число;
- усі функції реалізувати відповідно загальним вимогам.

Тип «Експонат» визначити як клас, що містить:

- закрите поле з назвою, реалізоване як символьний рядок та числове поле з ідентифікаційним номером у каталозі, а також реалізоване як об’єкт відповідного типу поле типу «Виставковий центр»;
- усі функції реалізувати відповідно загальним вимогам.

У тестовому прикладі створити об’єкти типу «Виставковий центр» та «Експонат» відповідно загальним вимогам і вивести інформацію про них у повному та скороченому вигляді.

5.2.7 Запитання до самоконтролю

1. Що таке об’єктна декомпозиція? Дати визначення поняттям «повідомлення», «метод», «поведінка об’єкту». Навести приклад опису класу, визначення об’єкта і передачі повідомлення об’єкту в C++.
2. Які механізми забезпечують захист об’єкта?
3. Які методи компілятор може надати для нового класу? В яких випадках вони не надаються?
4. Чи обов’язково в описі класу повинні міститись конструктор і деструктор? Скільки їх може бути? Що таке конструктор умовчання? Для чого потрібен список ініціалізації?
5. У яких випадках відбувається виклик конструктора копій? Чим відрізняється поверхневе і глибоке копіювання об’єктів? Як працює конструктор копії за умовчанням?
6. Що таке функції-селектори та функції-модифікатори? Що таке «посилання на себе»?
7. Що таке inline-функції і для чого вони потрібні?
8. Пояснити призначення і використання статичних членів класу.

5.3. Компютерний практикум № 3.

5.3.1. Тема

Реалізація успадкування в C++.

5.3.2. Мета роботи.

Навчитися правильно будувати ієрархії об’єктів з використанням успадкування та агрегації засобами C++.

5.3.3. Основні теоретичні відомості

Ієрархічність та її різновиди. Для подолання складності системи будують як ієрархічні. Ієрархія – це впорядкування абстракцій, розташування їх по рівнях. Розрізняють два різновиди ієрархій: ієрархія класів або спадковість і ієрархія об’єктів або агрегація.

Спадковість – це механізм отримання нового класу із існуючих шляхом запозичення структурної або функціональної частини одного або декількох інших. Відповідно розрізняють одиночне та множинне успадкування. Спадковість створює таку ієрархію абстракцій, у якій підкласи (похідні класи, класи-нащадки) успадковують будову і поведінку одного або декількох суперкласів (базових класів, батьківських класів). Кажуть, що спадковість описується відношенням «`is-a`» і породжує ієрархію «узагальнення-спеціалізація».

Агрегація – це такий різновид ієрархії, який передбачає використання об’єктів одного класу в оголошенні іншого класу. Агрегацію (або включення) можна описати відношенням «`part of`» або «бути частиною». Клас, що містить поля – об’єкти інших класів, називається агрегатом або контейнером.

Одиночна спадковість. Реалізація в C++ . Ідея спадковості одна з основних в об’єктно-орієнтованому програмуванні. З погляду фізичної реалізації побудова нового класу на базі старого дає можливість повторного використання старого коду і додавання нових властивостей, зміни деяких аспектів поведінки за рахунок заміни коду деяких методів (функцій) і прав доступу. Приклад оголошення:

```
class Student
{char* name, *group;..}
class Starosta: public Student{int level;   . . . }
```

Клас `Starosta` – похідний від класу `Student`, а `Student` – базовий для `Starosta`.

Клас `Starosta` крім своїх власних членів (`level`) містить і члени класу `Student`.

Виведення похідного класу із базового робить його підтипом базового. Тому вказівник на базовий клас можна використовувати як вказівник на похідний (але не навпаки). Наприклад:

```
void f(Student * p, Starosta *q)
{Student *KA01 [10];
KA01[0] = p;
KA02[1] = q;…}
```

Щоб клас можна було використовувати як базовий, його потрібно повністю визначити до визначення похідного, недостатньо тільки оголосити.

Для членів похідного класу доступні усі відкриті (**`public`**) та захищені (**`protected`**) члени базового класу напряму, як власні. Закриті поля (**`private`**) базового класу входять до складу похідного, але недоступні напряму – тільки через відкриті методи базового класу.

В C++ існує три різновиди успадкування: відкрите (**`public`**), захищене (**`protected`**) та за умовчанням закрите (**`private`**)

За допомогою цього механізму похідний клас може змінити рівень доступу до успадкованої від базового класу частини своїх об’єктів.

Загальне правило: за допомогою похідного класу елементи базового класу не можна зробити більш відкритими (тільки більш закритими).

Конструктори похідних класів. В оголошенні любого конструктора похідного класу потрібно враховувати наступне.

- Конструювання об’єкту похідного класу обов’язково відбувається з викликом конструктора базового класу.

- Якщо явний виклик конструктора базового класу не передбачено у конструкторі похідного, за умовчання буде викликано конструктор без параметрів базового класу.
- Для організації виклику потрібної версії конструктора базового класу, можна скористатись списком ініціалізації, наприклад:

```
Student::Student (char*a, char* b):name(a),group(b){…}
Starosta::Starosta(char*a, char* b, int c):Student (a,b),
level (c) {…}
```

Об'єкти створюються згори вниз: спочатку базовий клас, тоді члени похідного класу, потім сам похідний клас. Знищуються – в протилежному порядку. Про це особливо важливо пам'ятати. коли в деструкторі необхідно явно звільняти ресурси.

Похідний клас може бути одночасно базовим для іншого класу. Такий набір зв'язаних класів називається ієрархією класів.

Первизначення функцій в похідних класах. Якщо в похідному класі створена функція з таким самим ім'ям, що і в базовому класі, то має місце заміщення методу. Кажуть, що в цьому випадку метод похідного класу приховує метод або методи (якщо їх декілька з однаковими іменами і різними сигнатурами) базового класу.

Множинне успадкування. Допустиме у C++ множинне успадкування (**multiple inheritance**) дає можливість отримати похідний клас від декількох базових. Для опису ієрархії множинного успадкування можна використати орієнтований ациклічний граф.

Синтаксис заголовка класу розширюється, щоб можна було використати список базових класів з атрибутами. Наприклад:

```
class A:  public B, public C{…}
```

Створення об'єкта похідного класу відбувається з викликом конструкторів базових класів у тій послідовності, в якій визначена спадковість, навіть якщо у списку ініціалізації вони описані у зворотному порядку. Для розглянутого приклада: **B, C, A.**

5.3.4. Приклад програми.

Базовий клас – «Населений пункт», похідний – «Адміністративна одиниця», додаткове поле – кількість мешканців.

У тестовому прикладі створити три об’єкти похідного типу, вивести повну інформацію про кожний з них, змінити поля одного з них і визначити відстань від заданої точки.

```
#include<stdio.h>
#include<math.h>
#include<string.h>

class Settlement{
char *name;
double x,y;
public:
     Settlement();
     Settlement(char*, double, double);
     Settlement(Settlement&);
     ~Settlement();
     Settlement& setname(char*);
     Settlement& setx(double);
     Settlement& sety(double);
     double getx();
     double gety();
     char* getname();
     void print();
     double mydist(double, double);
     };
class AdminUnit: public Settlement{
     long inhabitants;
     static int count;//оголошення статичної змінной в класі
public:
     AdminUnit();
     AdminUnit(char*, double, double, long);
     AdminUnit(AdminUnit&);
     ~AdminUnit();
     AdminUnit& setihabit(long);
     long& getihabit();
     void print();
};
int AdminUnit::count=0;//визнач.статичн.змінной за межами класу
Settlement::Settlement(){
          name=new char[8];
          strcpy(name, "noname");
          x=y=0;
     printf("default constructor Settlement\n");
     };
Settlement::Settlement(char*title, double xx, double yy){
          name=new char[strlen(title)+1];
          strcpy(name, title);
```

```
          x=xx;y=yy;
          printf("constructor Settlement with parameters\n");
     };
Settlement::Settlement(Settlement&a){
     x=a.x; y=a.y;
     name=new char[strlen(a.name)+1];
     strcpy(name, a.name);
     printf("copy constructor Settlement\n");
};
Settlement::~Settlement(){
     if(name) delete[] name;
     printf("destructor Settlement\n");
};
Settlement& Settlement::setname(char*title){
          delete[] name;
          name=new char[strlen(title)+1];
          strcpy(name, title);
          return *this;
     };
Settlement& Settlement::setx(double xx){x=xx;return *this;};
Settlement& Settlement::sety(double yy){y=yy;return *this;};
double Settlement::getx(){return x;};
double Settlement::gety(){return y;};
char*Settlement::getname(){return name;};
void Settlement::print(){
printf("settlement %s , x=%f, y=%f\n", name, x, y);
};
double Settlement:: mydist(double x0, double y0){
     return sqrt((x-x0)*(x-x0)+(y-y0)*(y-y0));
};
AdminUnit::AdminUnit():inhabitants(1000){   count++;
     printf("default constructor AdminUnit\n");
};
AdminUnit::AdminUnit(char*n, double a, double b, long
c):Settlement(n, a, b),inhabitants(c){count++;
     printf("constructor AdminUnit with parameters\n");
};
AdminUnit::AdminUnit(AdminUnit&a):Settlement(a.getname(),a.getx(),
a.gety()), inhabitants(a.getihabit()){count++;
     printf("copy constructor AdminUnit\n");
};
AdminUnit::~AdminUnit(){count--;
     printf("destructor AdminUnit\n");
};
AdminUnit& AdminUnit::setihabit(long a){inhabitants=a;return
*this;};
long& AdminUnit::getihabit(){return inhabitants;};
void AdminUnit::print(){
     Settlement::print();//виклик функціїї базового в похідн.класі
     printf("number of inhabitants is %ld, count=%d\n",
inhabitants, count);};

int _tmain(int argc, _TCHAR* argv[])
{
     double coordx, coordy, xx, yy;
```

```
     char title[20];
     long numb;
     AdminUnit obj1; obj1.print();
printf("\n enter Administrative Unit as name x y
number_of_inhabitants\n");
     scanf("%s%lf%lf%ld",title,&coordx, &coordy, &numb);
     AdminUnit obj2(title, coordx, coordy, numb);
     obj2.print();
     AdminUnit obj3=obj2;
     printf("\n This is obj3 as copy obj2 \n");
     obj3.print();
     printf("\n Enter new name and coordinates for obj3\n");
     scanf("%s%lf%lf",title,&coordx, &coordy);
     obj3.setname(title).setx(coordx).sety(coordy);
     printf("\n This is obj3 after change\n");
     obj3.print();
printf("Enter the coordinates to search for a distance from
obj3\n");
     scanf("%lf%lf", &xx, &yy);
     printf("this is %f \n", obj3.mydist( xx, yy)) ;
          return 0;
}
```

Можливий сеанс роботи:

```
default constructor Settlement
default constructor AdminUnit
settlement noname , x=0.000000, y=0.000000
number of inhabitants is 1000, count=1

 enter Administrative Unit as name x y number_of_inhabitants
first 1.1 2.2 12000
constructor Settlement with parameters
constructor AdminUnit with parameters
settlement first , x=1.100000, y=2.200000
number of inhabitants is 12000, count=2
constructor Settlement with parameters
copy constructor AdminUnit

 This is obj3 as copy obj2
settlement first , x=1.100000, y=2.200000
number of inhabitants is 12000, count=3

 Enter new name and coordinates for obj3
second 4.5 6.7

 This is obj3 after change
settlement second , x=4.500000, y=6.700000
number of inhabitants is 12000, count=3
Enter the coordinates to search for a distance from obj3
3.5 5.6
this is 1.486607
destructor AdminUnit
destructor Settlement
destructor AdminUnit
destructor Settlement
```

```
destructor AdminUnit
destructor Settlement
Press any key to continue . . .
```

5.3.5. Порядок виконання роботи

Третя робота є розвитком другої роботи і код другої роботи повністю входитиме до коду третьої. Один або два класи, описані у другій роботі, мають стати базовими класами у третій відповідно варіанту. Необхідно передбачити:

- Описати нові класи як відкриті похідні від указаного базового;
- Виконати оголошення й реалізацію конструкторів за умовчанням, з параметрами та копій для похідних класів, у конструкторі з параметрами обов'язково передбачити використання конструктора з параметрами базового класу;
- у реалізації функції виведення на екран інформації в похідних класах використати прототип відповідної функції базового класу;
- **до конструкторів та деструкторів базового, похідного та одного з агрегованих класів долучити виведення відповідної контрольної інформації для можливості відстеження порядку створення та знищення об'єктів;**
- продумати, які з полів варто переробити на статичні;
- у класах, розроблених у попередній роботі, може виникнути необхідність внести деякі зміни: додати додаткове поле, переробити статус поля, тощо.

У тестовому прикладі створити по одному об'єкту кожного з похідних класів для контролю етапів утворення відповідних об'єктів й викликати функції виведення інформації. Пояснити отримані результати.

5.3.6. Варіанти завдань

Варіант1.

Базовий клас – «Людина», похідні:

- «Учасник черги на отримання житла»; додаткові поля – дата постановлення на облік, наявність пільг, порядковий номер у черзі;

- «Учасник виїзної конференції»; додаткові поля – чи потребує поселення, тривалість доповіді, час початку роботи конференції, час початку виступу даного доповідача.

  У тестовому прикладі створити об’єкти похідних типів.

  Варіант2.

  Базовий клас – «Фігура», похідні:

- «кінь», «пішак»; додаткові поля – порядковий номер, ознака чи своя половина поля;
- «ферзь» зі своїми методами «хід» і «удар».

  У тестовому прикладі створити об’єкти похідних типів.

  Варіант3.

  Базовий клас – «Фігура», похідні:

- «шашка»; додаткове поле – порядковий номер;
- «дамка», методи – «хід» і «удар».

  У тестовому прикладі створити об’єкти похідних типів.

  Варіант4.

  Базовий клас – «Точка на площині», похідні:

- «Ромб»; додаткові поля – довжини діагоналей;
- «Коло»; додаткове поле – радіус;
- похідний від класу «Коло» клас «Вписаний правильний трикутник» – додаткове поле – вершина типу «Точка на площині» і метод виведення координат усіх вершин.

  У тестовому прикладі створити об’єкти похідних типів.

  Варіант5.

  Базовий клас – «Нота», похідні:

- «звук»; додаткове поле – частота;
- «зображення»; додаткові поля – координати на екрані лівого верхнього кута фрагмента нотного стану;
- у обох класах додаткове поле – порядковий номер ноти.

  У тестовому прикладі створити об’єкти похідних типів.

Варіант6.

Базовий клас – «Іграшка», похідні:

- «машина»; додаткові поля – наявність дистанційного керування, порядковий номер;
- «м'яка іграшка»; додаткові поля – матеріал, звук.

У тестовому прикладі створити об'єкти похідних типів.

Варіант7.

Базовий клас – «Годинник», похідні:

- «з прямокутним табло» ; додаткові поля – координати двох протилежних кутів;
- «з круглим циферблатом» ; додаткові поля – координати центру, радіус.

У тестовому прикладі створити об'єкти похідних типів.

Варіант8.

Базовий клас – «Товар», похідні:

- «Промисловий товар» ; додаткові поля – умови транспортування, місце знаходження: на складі, в торговому залі, на вітрині;
- «Харчовий продукт»; додаткові поля – дата виготовлення, термін зберігання).
- Для кожного класу визначити функцію `alarm()` – для промислового товару з повідомленням із умов транспортування («не кантувати», «обережно!», ...), або «товар непридатний для споживання» – для харчового, для базового – просто назва товару.

У тестовому прикладі створити об'єкти похідних типів.

Варіант9.

Базовий клас – «Експонат», похідні:

- «Витвір мистецтва» ; додаткові поля – рік створення, ціна, автор;
- «Промисловий товар»; додаткові числові поля – категорія і ціна, назва фірми-виробника.

У тестовому прикладі створити об'єкти похідних типів.

Варіант10.

Базовий клас – «Страва», похідні:

- «Готова страва» ; додаткові поля – час приготування, наявність в меню (0 або 1);
- «Напівфабрикат»; додаткові числові поля – умови і термін зберігання, вага.

У тестовому прикладі створити об’єкти похідних типів.

Варіант11.

Базовий клас – «Учасник змагань», похідні:

- «футбольна команда», додатково – кількість забитих голів, результат, порядковий номер;
- «легкоатлет», додатково – час, час лідера, відставання від лідера, місце у фінальній таблиці;
- для усіх класів визначити функцію `print()` – друк тільки назви учасника або і назви і кількості голів для футбольної команди або часу для легкоатлета.

У тестовому прикладі створити об’єкти похідних типів.

Варіант12.

Базовий клас – «Поштове відправлення», похідні:

- «Рекомендований лист», додатково – дата доставки, прізвище листоноші;
- «Бандероль», додатково – числові поля вага і габарити;

У тестовому прикладі створити об’єкти похідних типів.

Варіант13.

Базовий клас – «Станція», похідні:

- «радіостанція»; додаткові поля – досяжність, вартість ефірного часу, діапазон частот, порядковий номер;
- «залізнична станція»; додаткові поля – кількість запасних шляхів, тривалість зупинки швидкісних потягів, категорія: ціле без знака 1..5, порядковий номер.

У тестовому прикладі створити об’єкти похідних типів.

Варіант14.

Тип «Форма існування» описати як базовий клас з єдиним методом `move()`, який виводить поточне положення.

Похідні від нього: «істота» (додаткове поле – «ознака життя»:0 або 1), «рослина» (координати), «нерухомий об’єкт» (координати, назва), похідні від істоти: «хижак» (максимальний вік істот цього класу) і «здобич» (максимальний вік істот цього класу).

Визначити функцію `move()`, яка для істот виконує наступне: збільшує на одиницю координати та збільшує на одиницю вік, поки він не стане більший за максимальний, тоді ознака життя стає нулем.

У тестовому прикладі створити об’єкти усіх похідних типів.

Варіант15.

Базовий клас – «Кліматичні умови», похідні:

- «кліматичні умови в теплиці» додаткові поля – оптимальні кліматичні умови, допуски;
- «кліматичні умови на городі» додаткові поля – критичний рівень вологості, критичні рівні кислотності, розташування як об’єкт відповідного типу (див. КП1);
- визначити функцію `show()`: для базового – поточний стан, для похідних – виводити тільки ті значення, які перевищують критичні, і розмір цього перевищення.

У тестовому прикладі створити об’єкти похідних типів.

Варіант16.

Базовий клас – «Твір», похідні:

- «Музичний твір», додатково – з полями типу «Виконавець» і числовим полем з тривалістю звучання;
- «Друковане видання», додатково – з числовим полем, що визначає кількість друкованих сторінок та полем типу «Виконавець», що визначатиме видавця.

У тестовому прикладі створити об’єкти похідних типів.

Варіант17.

Базовий клас – «Товар на складі», похідні:

- «продукт з малим терміном зберігання» додаткові поля – оптимальна температура, дата поставки, термін зберігання,

- «хімічний елемент» додаткові поля – оптимальна температура, оптимальна вологість, допуски по температурі і вологості,
- визначити функцію `attention()` – для кожного класу з різною реалізацією.

У тестовому прикладі створити об’єкти похідних типів.

Варіант18.

Базовий клас – «Фраза» похідні:

- «число», додаткові поля – система числення, довжина дробової частини, форма запису – з фіксованою або з плаваючою точкою,
- «речення», додаткові поля – кількість символів в алфавіті, чи ігнорувати регістр,
- визначити функцію `view()`: виведення самої фрази, або разом із значенням основи системи числення, або разом із кількістю символів в алфавіті.

У тестовому прикладі створити об’єкти похідних типів.

Варіант19.

Базовий клас – «Число» похідні:

- «ціле», додаткові поля – наявність знакового розряду,
- «дійсне», додаткові поля – наявність знакового розряду, довжина дробової частини, форма подання: з плаваючою або фіксованою точкою, зробити це поле `static`,
- визначити функцію `print()` для кожного класу з різною реалізацією: просто значення або з указуванням типу.

У тестовому прикладі створити об’єкти похідних типів.

Варіант20.

Базовий клас – «Коло», похідні:

- «вписаний многокутник», додаткові поля – кількість сторін,
- «описаний многокутник», додаткові поля – кількість сторін, колір,
- увести статичну змінну з кількістю фігур і поле з порядковим номером фігури до базового класу.

У тестовому прикладі створити об’єкти похідних типів.

Варіант21.

Базовий клас – «Книга», похідні:

- «книга в бібліотеці», додаткові поля – інвентарний номер, ознака наявності, кому видана, вартість,
- «книга в магазині», додаткові поля – відпускна ціна, кількість екземплярів,

У тестовому прикладі створити об’єкти похідних типів.

Варіант22.

Базові типи «Обладнання» та «Модернізація». Похідні:

- від «Обладнання» – «Працююче обладнання», додаткові поля – дата введення в експлуатацію, залишкова вартість,
- від «Працююче обладнання» та «Модернізація» – «Модернізоване обладнання», додаткове поле – додана вартість;
- визначити функцію `view()` для назви і вартості, для кожного класу з різною реалізацією.

У тестовому прикладі створити об’єкти похідних типів.

Варіант23.

Базовий клас – «Товар»,

- похідний від нього – «виготовлений товар», додаткові поля – дата виготовлення та кількість,
- похідний від «виготовленого» – «проданий товар», додаткові поля – шифр партії (складається з дати виготовлення і порядкового номера), дата продажу, кількість в партії,
- визначити функцію `print()` для кожного класу з різною реалізацією (назва і категорія).

У тестовому прикладі створити об’єкти похідних типів.

Варіант24.

Базовий клас – «Рахунок», похідні:

- «депозитний» додаткові поля – дата відкриття, період, ставка, сума,
- «розрахунковий» додаткові поля – дата останньої операції, ставка, залишок.

У тестовому прикладі створити об’єкти похідних типів.

Варіант25.

Базовий клас – «Підключення», похідні:

- «за контрактом» додаткові поля – номер рахунку, дата відкриття, кваліфікація рахунку, залишок,
- «за карткою» додаткові поля – дата закінчення, залишок.

У тестовому прикладі створити об’єкти похідних типів.

Варіант26.

Базовий клас – «Персона», похідні:

- «Студент» додаткові поля – назва групи, курс і рейтинг,
- «Співробітник» додаткові поля – посада, оклад, стаж роботи.

У тестовому прикладі створити об’єкти похідних типів.

Варіант27.

Базовий клас – «Земельна ділянка» похідні:

- «Сільськогосподарського призначення» додаткові поля – тип грунту, рік останньої рекультивації,
- «Під забудову» додаткові поля – доступність інженерних мереж, віддаленість від населеного пункту, вартість будівництва, .

У тестовому прикладі створити об’єкти похідних типів.

Варіант28.

Базовий клас – «Тварина», похідні:

- «Свійська» додаткові поля – назва, дата останнього щеплення, адреса хазяїна,
- «Дика» додаткові поля – адреса зоопарку, де вона утримується, вартість утримання, ідентифікаційний номер.

У тестовому прикладі створити об’єкти похідних типів.

Варіант29.

Базовий клас – «Екскурсія», похідний:

- «Пішохідна» додаткові поля – довжина маршруту, категорія, пункт закінчення,

- «Автобусна» додаткові поля – тип автобусу, розмір групи, номер автобусу.

  У тестовому прикладі створити об’єкти похідних типів.

  Варіант30.

  Базовий клас – «Видання», похідні:

- «Періодичне» додаткові поля – періодичність, категорія, наклад,
- «Книга» додаткові поля – тип палітурки, кількість сторінок, формат.

  У тестовому прикладі створити об’єкти похідних типів.

  Варіант31.

  Базовий клас – «Майстерня» , похідні:

- «Ательє індпошиву» додаткові поля – кількість видів робіт, можливість виїзду до замовника, середня вартість робіт,
- «Автомайстерня» додаткові поля – клас автомобілів, рейтинг, кількість персоналу.

  У тестовому прикладі створити об’єкти похідних типів.

  Варіант32.

  Базовий клас – «Експонат», похідні:

- «Харчовий товар»; додаткові поля – термін споживання, країна-виробник;
- «Промисловий товар»; додаткові числові поля – категорія і ціна, назва фірми-виробника.

  У тестовому прикладі створити об’єкти похідних типів.

5.3.7. Запитання до самоконтролю

1. Що таке успадкування? Описати структуру об’єкта похідного класу.
2. Описати правила доступу для різних типів успадкування.
3. Чим відрізняються два різновиди ієрархії? Навести приклади із свого варіанта.
4. Як відбувається конструювання об’єктів при множинному успадкуванні? При ромбовидному успадкуванні?

5.4. Компютерний практикум № 4.

5.4.1. Тема

Поліморфізм. Перевантаження функцій. Перевантаження операторних функцій.

5.4.2. Мета роботи.

Навчитися правильно користуватися можливостями перевантаження звичайних і операторних функцій.

5.4.3. Основні теоретичні відомості

Перевантаження функцій. Перевантаження функцій означає, що для передачі повідомлень об'єктам різних класів можна скористатись функціями з однаковими іменами і що кожний об'єкт буде реагувати відповідним чином.

Така техніка використовується для базових операцій C++. Наприклад, існує одна назва і одна позначка для додавання: "+", але його можна використовувати для додавання цілих, дійсних, для інкременту вказівників. Ця ідея поширюється і на функції користувача. Фактично це різні функції і перевантаження імен має значення тільки для зручності запису.

Під час виклику функції *f* процес пошуку потрібної із множини перевантажених відбувається як пошук найбільшої відповідності типів формальних і фактичних параметрів.

Перевантаження операторів. Перевантаження операторів – можливість зміни семантичного навантаження стандартних операторів відповідно потребам нового класу або нових потреб існуючого класу.

В C++ допустимі перевантаження таких операторів:

+, -, /,*, %, ^, &, !, ~, =, <, >, +=, - =, * =, / =, % =, ^ =, & =, / =, <<, >>, ==, ! =, < =, > =, &&, //, ++, --, -> *, [], (), *new*, *new*[], *delete*, *delete*[].

Не можуть бути перевизначені оператори:

::, ., .*, :, sizeof() та непередбачені в синтаксисі мови.

Назва операторної функції починається з ключового слова **operator**, за яким іде символ оператора, наприклад

```
operator *
```

Операторна функція оголошується і може бути викликана як звичайна функція. Використання її як оператора – скорочена форма явного виклику. Наприклад в описі класу «комплексне число» оголошено оператор += :

```
class Complex{
    double re, im;
public:
complex (double r, double i): re(r), im(i){};
complex& operator += (complex&);
};
```

Тоді операторна функція може бути використана двома способами:

```
Complex a(1,2), b(4,-5);
a. operator += (b);     //   явний виклик
a+=b;                   //   скорочена форма
```

У перевантажених операторах зберігається пріоритет і порядок виконання у виразах, закріплені за даними операторами.

Унарний оператор можна визначити або у вигляді функції-члена без аргументів, або у вигляді функції-не-члена з одним аргументом.

Оператор індексації використовується для класів-контейнерів і надає можливість доступу до елемента колекції, яку містить такий клас, напряму, за індексом. Такий оператор має повертати посилання на тип елемента колекції, якщо його бажано використовувати і зліва і справа від оператора присвоєння.

Бінарний оператор можна визначити або у вигляді функції-члена з одним аргументом, або у вигляді функції-не-члена з двома аргументами.

Для ілюстрації сказаного, можна розглянути наступний приклад. Нехай в деякому класі X існує ціле поле some. Операцію **“+=”** можна визначити як суму полів some двох об’єктів *X*:

```
class X{ . . .
 friend X& operator += (X&, const X&); . . .
};
X& operator += (X&a, const X&b) {
     a.some+=b.some; return a;
}
```

Або з використанням функції-члена:

```
class X{. . . X& operator +=(const X&);. . . };
X&  X :: operator += (const X&b){
     some += b.some; return* this;}
```

Оператор присвоєння відноситься до бінарних операторів і в С++ може бути перевизначений як звичайний бінарний оператор, але тільки як член класу.

Дружні класи і дружні функції. Оскільки деякі операторні функції можуть бути зовнішніми, виникає проблема доступу до прихованих частин класу. В цьому випадку використовують так звані дружні функції.

Дружні функції повинні бути оголошені в середині опису класу, з яким вони дружні, за допомогою префікса *friend*. Наприклад:

```
class A{int q; . . . friend void func (A, int);. . .};
void func (A a1, int i) {a1. q =i;};
```

Якщо усі функції-члени одного класу **X** є дружніми для другого класу **Y**, говорять, що клас **X** – дружній класу **Y.** Тоді цей факт оголошують так:

```
class Y{. . . friend class X;. . . };
```

Перевантаження операторів уведення – виведення. Оскільки існує можливість створення типів даних користувача, логічно мати можливість форматного уведення і виведення значень такого типу, скориставшись формою запису відповідних стандартних операторів. Це бінарні операції “<<” – «помістити в потік» і “>>” – «взяти із потоку». Вони можуть бути визначені тільки як зовнішні функції.

Операторну функцію “<<” потрібно перевизначити так, щоб вона мала два аргументи: типу `ostream &` і користувацького типу і повертала `ostream &` (посилання на потік використовується, щоб не копіювати об’єкт потоку).

Для розглянутого раніше типу **`complex`**:

```
class Complex {. . .
friend ostream & operator << (ostream & , Complex x);
. . .};
ostream & operator << (ostream & o, Complex x){
     return o <<x.re << ’+’ <<’ i’ <<x. im;    };
```

Аналогічно можна перевизначити оператор уведення з потоку.

У деяких випадках для перевантаження цих операторів зручно користуватись функціями-членами класів **`ostream`** та **`istream`**, наприклад **`put, get, getline, write, read`** та **`ignore`**.

5.4.4. Приклад програми

Задача: Описати клас Complex з перевантаженими арифметичними операторами і операторами уведення- виведення.

```
#include "stdafx.h"
#include<iostream>
using namespace std;
class Complex{
double re, im;
public:
     Complex();
     Complex(double, double);
     Complex(double);
     Complex(Complex&);
     ~Complex();
     Complex& operator=(Complex&);
     friend Complex operator+(Complex&, Complex&);
     Complex operator-(Complex&);
     friend Complex& operator+=(Complex&, Complex&);
     Complex& operator-=(Complex&);
     friend Complex& operator++(Complex&);
     Complex& operator--();
     friend Complex operator++(Complex&, int);
     Complex operator--(int);
     friend ostream& operator<<(ostream &, Complex&);
     friend istream& operator>>(istream &, Complex&);
};
Complex::Complex():re(0), im(0){};
Complex::Complex(double a, double b):re(a), im(b){};
Complex::Complex(double a):re(a), im(0){};
Complex::Complex(Complex&a):re(a.re), im(a.im){};
Complex::~Complex(){};
Complex& Complex::operator=(Complex&a){re=a.re; im=a.im; return
*this;};
Complex operator+(Complex& a, Complex& b){Complex temp;
temp.re=a.re+b.re; temp.im=a.im+b.im; return temp;};
Complex Complex::operator-(Complex&b){Complex temp; temp.re=re-
b.re;temp.im=im-b.im; return temp;};
Complex& operator+=(Complex&a, Complex&b){a.re+=b.re; a.im+=b.im;
return a;};
Complex& Complex::operator-=(Complex&b){re-=b.re; im-=b.im; return
*this;};
Complex& operator++(Complex&a){a.re+=1; a.im+=1; return a;};
Complex& Complex::operator--(){re--; im--; return *this;};
Complex operator++(Complex& a, int i){Complex temp(a);a.re+=1;
a.im+=1; return temp; };
Complex Complex::operator--(int i){Complex temp(*this); re-=1; im-
=1; return temp;};
```

```
     ostream& operator<<(ostream &out, Complex&c){return
out<<'('<<c.re<<','<<c.im<<')';};
     istream& operator>>(istream &inp, Complex&c){char c1, c2, c3;
return inp>>c1>>c.re>>c2>>c.im>>c3;};

int _tmain(int argc, _TCHAR* argv[]){
     Complex c1, c2(5.6), c3(1.1, 2.2), c4;
     cout<<"c1="<<c1<<", c2="<<c2<<", c3="<<c3<<endl;
     c4=c2+c3;
     cout<<"c2+c3="<<c4<<endl;
     c4=c3-c2;
     cout<<"c3-c2="<<c4<<endl;
     cin>>c4;
     cout<<endl<<c4++;
     cout<<c4;
     return 0;
}
```

Можливий результат роботи:

```
c1=(0,0), c2=(5.6,0), c3=(1.1,2.2)
c2+c3=(6.7,2.2)
c3-c2=(-4.5,2.2)
(4.4,7.7)

(4.4,7.7)(5.4,8.7)Press any key to continue . . .
```

5.4.5. Порядок виконання роботи

Четверта робота базована на коді третьої роботи, є її розвитком.

Відповідно варіанту до тексту третьої роботи потрібно внести наступні зміни:

- перевантажити указані арифметичні, логічні оператори та оператори форматного уведення-виведення для класів відповідно варіанту;
- у випадку, якщо доцільно перевантажити ці оператори для інших класів варіанту, це потрібно зробити;
- визначити оператор присвоювання для класів, для яких це доцільно;
- у конструкторах і деструкторах закоментувати функції виведення контрольної інформації, щоб скоротити текст результату.

Реалізувати тестовий приклад, у якому передбачити демонстрацію роботи кожного з перевантажених операторів.

5.4.6. Варіанти завдань

Варіант 1 Визначити:

- арифметичні оператори “+=” для класу «Учасник виїзної конференції» за значенням поля тривалість доповіді та “--” за порядковим номером для; «Учасника черги»;
- оператори порівняння “<” для «Учасник виїзної конференції» за тривалістю доповіді та “==” для перевірки збігу двох об’єктів класу «Людина» за усіма полями;
- оператори форматного уведення-виведення – для класів «Адреса», та «Людина».

Варіант 2 Визначити

- арифметичний оператор “++” для класів «кінь» та «пішак» з переходом до наступної позиції (логику визначити самостійно);
- оператори порівняння “==” визначити для усіх похідних класів за положенням на дошці;
- оператори форматного уведення-виведення – для класу «Позиція на шахівниці».

Варіант 3.Визначити:

- арифметичний оператор “-” для класу «Шашка», якщо одна фігура може побити іншу, тоді координати однієї з них стають 0,0 і оператор “++” для перехода у наступну позицію вперед (можна скористатись префіксною і постфіксною формою);
- оператори порівняння “<”та “>” для порівняння об’єктів класу “Шашка” за значеннями їх віддаленості від протилежного краю дошки;
- оператори форматного уведення-виведення – для позиції і фігури.

Варіант 4 Визначити:

- арифметичні оператори “+” та “++”для класу «Дата»;
- оператор порівняння “<” для кола за радіусом;
- оператори порівняння “==” за значеннями усіх полів для усіх похідних класів;
- оператори форматного уведення-виведення – для дати і точки.

Варіант 5 а.Визначити:

- арифметичні оператори “--” та “++”для класу «Нота» для зміни значення тривалості звучання на 0.125;
- оператор порівняння “<” для «Звук» за частотою;
- оператори порівняння “==” за значеннями усіх полів для обох похідних класів;
- оператори форматного уведення-виведення – для ноти і звука.

Варіант 6 Визначити:

- арифметичні оператори “+” та “++”для класу «Іграшка» за кількістю на складі;
- оператор порівняння “<” для іграшки за ціною;
- оператори порівняння “==” за значеннями усіх полів для виробника;
- оператори форматного уведення-виведення – для виробника і іграшки.

Варіант 7. Визначити:

- арифметичні оператори “+” та “++”для класу «Час»;
- оператор порівняння “<” для годинника «з круглим циферблатом» за радіусом;
- оператори порівняння “==” за значеннями усіх полів для класу «Годинник»;
- оператори форматного уведення-виведення – для класів «Час» і «Годинник».

Варіант 8. Визначити:

- арифметичні оператори “+” та “++”для класу «Товар» за кількістю;
- оператор порівняння “<” для класу «Товар» за ціною;
- оператори порівняння “==” за значеннями усіх полів для класу «Товар»;
- оператори форматного уведення-виведення – для класу «Постачальник».

Варіант 9. Визначити:

- арифметичні оператори “+” та “++”для класу «Промисловий товар» за ціною;
- оператор порівняння “<” для «Витвір мистецтва» за роком створення;

- оператори порівняння “==” за габарітними розмірами для класу «Експонат»;
- оператори форматного уведення-виведення – для класу «Адреса».

Варіант 10. Визначити:

- арифметичні оператори “+” та “++”для класу «Страва» за ціною;
- оператор порівняння “<” для «Готова страва» за часом приготування;
- оператори порівняння “==” за усіма полями для класу «Кухар»;
- оператори форматного уведення-виведення – для класів «Кухар» та «Страва».

Варіант 11. Визначити:

- арифметичні оператори “+=” та “++”для класу «Футбольна команда» за кількістю забитих голів;
- оператор порівняння “<” для «Футбольна команда» за кількістю забитих голів;
- оператори порівняння “==” за усіма полями для класу «Учасник змагань»;
- оператори форматного уведення-виведення – для класів «Людина» та «Учасник змагань».

Варіант 12. Визначити:

- арифметичні оператори “+” та “++”для класу «Дата»;
- оператор порівняння “<” для «Бандероль» за вагою;
- оператори порівняння “==” за усіма полями для класу «Поштове відправлення»;
- оператори форматного уведення-виведення – для класів «Дата» та «Адреса».

Варіант 13. Визначити:

- арифметичні оператори “++”для класу «Радіостанція» за досяжністю та “+=” – за діапазоном частот;
- оператор порівняння “<” для «Залізнична станція» за кількістю запасних шляхів;

- оператори порівняння “==” за усіма полями для класу «Радіостанція» за досяжністю;
- оператори форматного уведення-виведення – для класів «Розташування» та «Станція».

Варіант 14. Визначити:

- арифметичні оператори “–” для класів «Хижак» та «Здобич» незалежно від віку (між ними, тільки як дружню), “++” за віком для тих самих класів;
- оператори порівняння “==” за усіма полями для класу «Нерухомий об’єкт»;
- оператори форматного уведення-виведення – для двох похідних класу «Форма існування».

Варіант 15. Визначити:

- арифметичні оператори “—” та “++”для класу «Кліматичні умови» за рівнем вологості;
- оператор порівняння “<” для «Кліматичні умови» за вологістю;
- оператори порівняння “==” за усіма полями для класу «Кліматичні умови»;
- оператори форматного уведення-виведення – для класів «Кліматичні умови» та «Розташування».

Варіант 16. Визначити:

- арифметичні оператори “+” за розміром гонорару для класу «Твір» та “++”для класу «Друковане видання» за кількістю сторінок;
- оператор порівняння “<” для «Друковане видання» за кількістю сторінок;
- оператори порівняння “==” за усіма полями для класу «Виконавець»;
- оператори форматного уведення-виведення – для класів «Виконавець» та «Твір».

Варіант 17. Визначити:

- арифметичні оператори “++”для класу «Товар на складі» за ціною та “+=” – за сумарною вартістю;
- оператор порівняння “<” для «Товар на складі» за сумарною вартістю;

- оператори порівняння “==” за усіма числовими полями для класу «Хімічний елемент»;
- оператори форматного уведення-виведення – для класів «Товар на складі» та «Розташування».

Варіант 18. Визначити:

- арифметичний оператор “+” для класу «Число»;
- оператор порівняння “<” для «Число»;
- оператори порівняння “==” за вмістом для класу «Фраза»;
- оператори форматного уведення-виведення – для класів «Алфавіт» та «Фраза».

Варіант 19. Визначити:

- арифметичні оператори “+” і “++”для класу «Ціле»;
- оператор порівняння “<” для «Ціле»;
- оператори порівняння “==” за вмістом для класу «Дійсне»;
- оператори форматного уведення-виведення – для класів «Ціле» та «Дійсне».

Варіант 20. Визначити:

- арифметичні оператори “+” та “++”для класу «Координати» за значеннями одразу двох полів;
- оператор порівняння “<” для «Описаний многокутник» за кількістю вершин;
- оператори порівняння “==” за усіма полями для класу «Вписаний многокутник»;
- оператори форматного уведення-виведення – для класів «Коло» та «Координати».

Варіант 21.Визначити:

- арифметичні оператори “+” та “++”для класу «Книга в магазині» за кількістю екземплярів;
- оператор порівняння “<” для «Книга в бібліотеці» за ціною;
- оператори порівняння “==” за усіма полями для класу «Книга»;

- оператори форматного уведення-виведення – для класів «Автор» та «Книга».

Варіант 22.Визначити:

- арифметичні оператори “+” та “++”для класу «Працююче обладнання» за вартістю;
- оператор порівняння “<” для «Працююче обладнання» за вартістю;
- оператори порівняння “==” за усіма полями для класу «Обладнання»;
- оператори форматного уведення-виведення – для класів «Обладнання» та «Працююче обладнання».

Варіант 23. Визначити:

- арифметичні оператори “+=” та “++”для класу «виготовлений товар» за кількістю;
- оператор порівняння “<” для «виготовлений товар» за кількістю;
- оператори порівняння “==” за усіма полями для класів «Виробник» та «Товар»;
- оператори форматного уведення-виведення – для класів «Виробник» та «Товар».

Варіант 24. Визначити:

- арифметичні оператори “+=” та “++”для класу «Розрахунковий» за залишком;
- оператор порівняння “<” для «Розрахунковий» за залишком;
- оператори порівняння “==” за усіма полями для класів «Банк» та «Рахунок»;
- оператори форматного уведення-виведення – для класів «Банк» та «Рахунок».

Варіант 25. Визначити:

- арифметичні оператори “+=” та “++”для класу «Дата»;
- оператор порівняння “<” для «За карткою» за залишком;
- оператори порівняння “==” за усіма полями для класів «Дата» та «Підключення»;

- оператори форматного уведення-виведення – для класів «Дата» та «Підключення».

Варіант 26. Визначити:

- арифметичні оператори “++”для класу «Студент» та“+=” для класу «Дата»;
- оператор порівняння “<” для «Студент» за рейтингом;
- оператори порівняння “==” за усіма полями для класів «Дата» та «Персона»;
- оператори форматного уведення-виведення – для класів «Дата» та «Персона».

Варіант 27. Визначити:

- арифметичні оператори “++” та“+=” для класу «Під забудову» за вартістю будівництва;
- оператор порівняння “<” для «Під забудову» за вартістю будівництва;
- оператори порівняння “==” за усіма полями для класу «Власник»;
- оператори форматного уведення-виведення – для класів «Власник» та «Земельна ділянка».

Варіант 28. Визначити:

- арифметичний оператор “++”для класу «Тварина»;
- оператор порівняння “<” для «Тварина» за віком;
- оператори порівняння “==” за усіма полями для класів «Опис» та «Тварина»;
- оператори форматного уведення-виведення – для класів «Опис» та «Тварина».

Варіант 29. Визначити:

- арифметичні оператори “++” та“+=” для класу «Екскурсія» за вартістю;
- оператор порівняння “<” для «Екскурсія» за вартістю;
- оператори порівняння “==” за усіма полями для класів «Організатор» та «Екскурсія»;
- оператори форматного уведення-виведення – для класів «Організатор» та «Екскурсія».

Варіант 30. Визначити:

- арифметичні оператори для класу «Видання» “++” за кількістю сторінок та для «Періодичне» “+=” за накладом;
- оператор порівняння “<” для «Періодичне» за накладом;
- оператори порівняння “==” за усіма полями для класів «Видавець» та «Видання»;
- оператори форматного уведення-виведення – для класів «Видавець» та «Видання».

Варіант 31. Визначити:

- арифметичні оператори “++”для класу «Ательє індпошиву» за кількістю видів робіт;
- оператор порівняння “<” для «Автомайстерня» за рейтингом;
- оператори порівняння “==” за усіма полями для класів «Адреса» та «Майстерня»;
- оператори форматного уведення-виведення – для класів «Адреса» та «Майстерня».

Варіант 32. Визначити:

- арифметичні оператори “++”для класу «Харчовий товар» за терміном споживання;
- оператор порівняння “<” для «Промисловий товар» за ціною;
- оператори порівняння “==” за усіма полями для класу «Виставковий центр»;
- оператори форматного уведення-виведення – для класів «Виставковий центр» та «Експонат».

5.4.7. Запитання до самоконтролю

1. Що таке перевантаження функцій?
2. Навести принцип визначення кількості параметрів операторної функції для перевантаження унарних та бінарних операторів залежно від порядку її оголошення як функції-члена або не-члена класу.
3. Що повертає операторна функція для бінарного логічного оператора?
4. Як повинен бути організований оператор індексації?

5.5. Компютерний практикум № 5.

5.5.1. Тема

Поліморфізм. Механізм віртуальних функцій

5.5.2. Мета роботи.

Навчитися правильно використовувати механізм віртуальних функцій в С++.

5.5.3. Основні теоретичні відомості

Перевантаження операторів. Перевантаження операторів – можливість зміни семантичного навантаження стандартних операторів відповідно потребам нового класу або нових потреб існуючого класу.

Назва операторної функції починається з ключового слова **`operator,`** за яким іде символ оператора, наприклад

```
operator *
```

У перевантажених операторах зберігається пріоритет і порядок виконання у виразах, закріплені за даними операторами.

Оператор індексації використовується для класів-контейнерів і надає можливість доступу до елемента колекції, яку містить такий клас, напряму, за індексом. Такий оператор має повертати посилання на тип елемента колекції, якщо його бажано використовувати і зліва і справа від оператора присвоєння.

Оператор присвоєння відноситься до бінарних операторів і в С++ може бути перевизначений як звичайний бінарний оператор, але тільки як член класу.

Перетворення типів. У арифметичних виразах можуть бути присутні об’єкти різних типів, тому виникає проблеми перетворення типів.

Для перетворення визначеного раніше типу до типу, який визначає користувач, можна скористатись конструктором з одним параметром (конструктор перетворення). Наприклад:

```
complex (double r) {re = r; im = 0;}
```

Для перетворення у зворотному порядку, використовують спеціальну нестатичну функцію-член класу

```
operator тип () {…};
```

Наприклад:

```
complex::operator double(){return (sqrt(re*re+im*im));};
```

Пізнє зв'язування. Якщо у класах, зв'язаних відносинами успадкування, існують функції з однаковими сигнатурами, це означає, що об'єктам цих класів можуть бути передані однакові повідомлення. Може виникнути ситуація, коли на етапі компіляції відомо, яке повідомлення потрібно передати, але тільки під час виконання стане відомо об'єкту якого класу. У цьому випадку використовують механізм пізнього зв'язування.

В С++ пізнє зв'язування реалізоване за допомогою функцій-членів, які називаються **віртуальними функціями**.

Віртуальну функцію оголошують у базовому класі за допомогою префікса **`virtual`**, а потім переозначають у похідних класах.

Віртуальні функції покладаються на додаткову структуру даних, яка підтримує зв'язок між різними версіями функцій. Це таблиця віртуальних функцій **(virtual method table-vtbl)** – таблиця вказівників на віртуальні функції, яка конструюється для кожного класу окремо. Всі екземпляри класу містять вказівник на цю таблицю.

Такий механізм дозволяє асоціювати і зв'язати повідомлення з методом під час виконання програми автоматично. Програміст тільки визначає дії, які повинен виконати об'єкт, отримавши повідомлення.

Щоб досягти пізнього зв'язування для об'єкта, необхідно скористатись вказівником або посиланням на нього. Для відкритих похідних класів вказівники і посилання на об'єкти цих класів сумісні з вказівниками і посиланнями на об'єкти базового класу. Обрана функція-член залежить від класу, на об'єкт якого вказана, а не від типу вказівника.

Наприклад [4]:

```
class Animal{
      int age;
public:  . . .  virtual void Ask();
};
class Dog: public Animal{
    char*  breed;
public: . . . void Ask();
};
```

```
class Cat: public Animal{
      char* color;
public: . . . void Ask();
};
void Animal:: Ask(){cout<<”I’m animal ! ”;};
void Dog:: Ask(){cout<<”I’m “<<breed <<’!’;};
void Cat:: Ask(){cout<<”I’m “<<color<<”  cat! ”;};
void main(){
     Animal *a[3];
     Animal a;
     Cat c(“white”);
     Dog d(“Doberman”);
 a[0] = &a;  a[1] = &c; a[2] = &d;
 for (int i;  i=0;  i<3) a[i]->Ask();   };
```

Результат:

```
I’m animal! I’m white cat! I’m Doberman!
```

Якщо в класі присутня хоча б одна віртуальна функція, його деструктор потрібно визначити як віртуальний.

5.5.4. Порядок виконання роботи

Відповідно варіанту до тексту програми четвертого комп’ютерного практикуму потрібно внести наступні зміни:

- перевантажити оператор індексації для новоствореного класу відповідно варіанту;
- перевірку коректності за бажанням можна реалізувати без генерування виключних ситуацій;
- визначити оператор перетворення типу для класу відповідно варіанту;
- функцію, що виводить деяку скорочену інформацію про об’єкт, визначити як віртуальну.

Тестовий приклад продумати так, щоб продемонструвати об’єкт нового класу та роботу усіх новостворених функцій.

Демонстрацію роботи кожної з віртуальних функцій потрібно організувати так, щоб очевидно був задіяний віртуальний механізм.

5.5.5. Варіанти завдань

Варіант 1.

Описати клас «Громадська організація», у якому передбачити поле з назвою цієї організації, кількістю її членів і їх переліком у вигляді динамічного масиву об'єктів типу «Людина».

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про людину. У класі «Учасник виїзної конференції» передбачити можливість перетворення типу до об'єкту типа «Людина» і навпаки.

Функцію `print()` з виведенням скороченої інформації в класах «Людина», «Учасник виїзної конференції» та «Учасник черги» переробити на віртуальну.

Варіант 2.

Описати клас «Партія», у якому передбачити поле з прізвищами гравців, поточною кількістю фігур і їх переліком у вигляді динамічного масиву об'єктів типу «Фігура».

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про фігуру. У класах «Кінь» та «Пішак» передбачити можливість перетворення типу до об'єкту типа «Фігура» і навпаки.

Функцію `print()` з виведенням скороченої інформації у класах «Фігура» та усіх похідних переробити на віртуальну.

Варіант 3.

Описати клас «Гра», у якому передбачити поле з прізвищами гравців, кількістю фігур на дошці і їх переліком у вигляді динамічного масиву об'єктів типу «Фігура».

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про фігуру. У класі «Шашка» передбачити можливість перетворення типу до об'єкту типа «Дамка» і навпаки.

Функцію `print()` з виведенням скороченої інформації у класах «Фігура» та усіх похідних переробити на віртуальну.

Варіант 4.

Описати клас «Зображення», у якому передбачити поле з назвою, кількістю фігур на зображенні і їх переліком у вигляді динамічного масиву об'єктів типу «Точка на площині».

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про точку. У класі «Коло» передбачити можливість перетворення типу до об'єкту типа «Точка на площині» і навпаки.

Функцію `print()` з виведенням скороченої інформації у класах «Точка на площині» та усіх похідних переробити на віртуальну.

Варіант 5.

Описати клас «Мелодія», у якому передбачити поле з назвою, кількістю нот і їх переліком у вигляді динамічного масиву об'єктів типу «Нота».

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про ноту. У класі «Звук» передбачити можливість перетворення типу до об'єкту типа «Нота» і навпаки.

Функцію `show()` з виведенням скороченої інформації у класах «Нота» та усіх похідних переробити на віртуальну.

Варіант 6.

Описати клас «Партія товару», у якому передбачити поле з назвою, динамічний масив об'єктів типу «Іграшка» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про іграшку. У класі «М'яка іграшка» передбачити можливість перетворення типу до об'єкту типа «Іграшка» і навпаки.

Функцію з виведенням скороченої інформації у класах «Іграшка» та усіх похідних переробити на віртуальну.

Варіант 7.

Описати клас «Майстерня», у якому передбачити поле з номером, динамічний масив об’єктів типу «Годинник» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про годинник. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію з виведенням скороченої інформації у класах «Годинник» та усіх похідних переробити на віртуальну.

Варіант 8.

Описати клас «Склад», у якому передбачити поле з номером, динамічний масив об’єктів типу «Товар» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про товар. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію з виведенням скороченої інформації у класах «Товар» та усіх похідних переробити на віртуальну.

Варіант 9.

Описати клас «Експозиція», у якому передбачити поле з назвою, динамічний масив об’єктів типу «Експонат» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про експонат. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію з виведенням скороченої інформації у класах «Експонат» та усіх похідних переробити на віртуальну.

Варіант 10.

Описати клас «Прайс», у якому передбачити поле з датою, динамічний масив об’єктів типу «Страва» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про страву. Для одного з похідних класів передбачити можливість перетворення типу до об'єкту базового і навпаки.

Функцію з виведенням скороченої інформації у класах «Страва» та усіх похідних переробити на віртуальну.

Варіант 11.

Описати клас «Турнір», у якому передбачити поле з роком провндення, динамічний масив об'єктів типу «Учасник змагань» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про учасника. Для одного з похідних класів передбачити можливість перетворення типу до об'єкту базового і навпаки.

Функцію `print()` у класах «Учасник змагань» та похідних переробити на віртуальну.

Варіант 12.

Описати клас «Пошта», у якому передбачити поле з датою, динамічний масив об'єктів типу «Поштове відправлення» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про поштове відправлення. Для одного з похідних класів передбачити можливість перетворення типу до об'єкту другого похідного і навпаки.

Функцію з виведенням скороченої інформації у класах «Поштове відправлення» та усіх похідних переробити на віртуальну.

Варіант 13.

Описати клас «Регіон», у якому передбачити поле з назвою, динамічний масив об'єктів типу «Станція» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації

про станцію. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію з виведенням скороченої інформації у класах «Станція» та усіх похідних переробити на віртуальну.

Варіант 14.

Описати клас «Поле», у якому передбачити поле з номером, динамічний масив об’єктів типу «Форма існування» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про форму існування. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту другого похідного і навпаки.

Функцію `move()` у класі «Форма існування» зробити суто віртуальною з переозначенням в усіх похідних.

Варіант 15.

Описати клас «Прогноз», у якому передбачити поле з номером, динамічний масив об’єктів типу «Кліматичні умови» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про кліматичні умови. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію з виведенням скороченої інформації у класах «Кліматичні умови» та усіх похідних переробити на віртуальну.

Варіант 16.

Описати клас «Концерт», у якому передбачити поле з виконавцем, динамічний масив об’єктів типу «Музичний твір» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про музичний твір. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію з виведенням скороченої інформації у класах «Твір» та усіх похідних переробити на віртуальну.

Варіант 17.

Описати клас «Склад», у якому передбачити поле з номером, динамічний масив об’єктів типу «Товар на складі» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про товар. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію `attention()`у класах «Товар на складі» та усіх похідних переробити на віртуальну.

Варіант 18.

Описати клас «Текст», у якому передбачити поле з заголовком, динамічний масив об’єктів типу «Фраза» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про фразу. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію `view()`у класах «Фраза» та усіх похідних переробити на віртуальну.

Варіант 19.

Описати клас «Масив», у якому передбачити поле з ідентифікатором, динамічний масив об’єктів типу «Число» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про число. Для одного з похідних класів передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію `print()` у класах «Число» та усіх похідних переробити на віртуальну.

Варіант 20.

Описати клас «Зображення», у якому передбачити поле з назвою, динамічний масив об'єктів типу «Коло» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про коло. Для одного з похідних класів передбачити можливість перетворення типу до об'єкту другого похідного і навпаки.

Функцію з виведенням скороченої інформації у класах «Коло» та усіх похідних переробити на віртуальну.

Варіант 21.

Описати клас «Бібліотека», у якому передбачити поле з номером, динамічний масив об'єктів типу «Книга в бібліотеці» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про книгу. Для одного з похідних класів передбачити можливість перетворення типу до об'єкту базового і навпаки.

Функцію з виведенням скороченої інформації у класах «Книга» та усіх похідних переробити на віртуальну.

Варіант 22.

Описати клас «Цех», у якому передбачити поле з датою, динамічний масив об'єктів типу «Працююче обладнання» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про обладнання.

Для класу «Працююче обладнання» передбачити можливість перетворення типу до об'єкту базового і навпаки.

Функцію з виведенням скороченої інформації у класах «Обладнання» та усіх похідних переробити на віртуальну.

Варіант 23.

Описати клас «Склад», у якому передбачити поле з номером, динамічний масив об'єктів типу «Товар» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про товар. Для класу «Виготовлений товар» передбачити можливість перетворення типу до об'єкту базового і навпаки.

Функцію `print()` у класах «Товар» та усіх похідних переробити на віртуальну.

Варіант 24.

Описати клас «Клієнт», у якому передбачити поле з прізвищем, динамічний масив об'єктів типу «Рахунок» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про рахунки. Для класу «Розрахунковий» передбачити можливість перетворення типу до об'єкту базового і навпаки.

Функцію `print()` у класах «Рахунок» та усіх похідних переробити на віртуальну.

Варіант 25.

Описати клас «Клієнти», у якому передбачити поле з назвою мережі, динамічний масив об'єктів типу «За контрактом» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про підключення за контрактом.

Для класу «Підключення за контрактом» передбачити можливість перетворення типу до об'єкту базового і навпаки.

Функцію `print()` у класах «Підключення» та усіх похідних переробити на віртуальну.

Варіант 26.

Описати клас «Кафедра», у якому передбачити поле з абревіатурою назви, динамічний масив об'єктів типу «Персона» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про персону. Для класу «Студент» передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію `print()` у класах «Персона» та усіх похідних переробити на віртуальну.

Варіант 27.

Описати клас «Населений пункт», у якому передбачити поле з назвою, динамічний масив об’єктів типу «Земельна ділянка» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про земельну ділянку. Для класу «Під забудову» передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію `print()` у класах «Земельна ділянка» та усіх похідних переробити на віртуальну.

Варіант 28.

Описати клас «Ветклініка», у якому передбачити поле з назвою, динамічний масив об’єктів типу «Тварина» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про тварину.

Для класу «Свійська» передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію `print()` у класах «Тварина» та усіх похідних переробити на віртуальну.

Варіант 29.

Описати клас «Тур», у якому передбачити поле з назвою, динамічний масив об’єктів типу «Екскурсія» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації

про екскурсію. Для класу «Пішохідна» передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію `print()` у класах «Екскурсія» та усіх похідних переробити на віртуальну.

Варіант 30.

Описати клас «Довідник», у якому передбачити поле з назвою, динамічний масив об’єктів типу «Видання» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про видання. Для класу «Періодичне» передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію `print()` у класах «Видання» та усіх похідних переробити на віртуальну.

Варіант 31.

Описати клас «Населений пункт», у якому передбачити поле з назвою, динамічний масив об’єктів типу «Майстерня» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про майстерню. Для класу «Автомайстерня» передбачити можливість перетворення типу до об’єкту базового і навпаки.

Функцію `print()` у класах «Майстерня» та усіх похідних переробити на віртуальну.

Варіант 32.

Описати клас «Виставка», у якому передбачити поле з назвою, динамічний масив об’єктів типу «Експонат» та розмірністю цього масиву.

Передбачити конструктори, деструктор і усі функції, які вважаєте за доцільне. Обовязково визначити оператор індексації для доступу до інформації про експонат. Для класу «Промисловий товар» передбачити можливість перетворення типу до об’єкту «Харчовий товар» і навпаки.

Функцію `print()` у класах «Експонат» та усіх похідних переробити на віртуальну.

5.5.6. Запитання до самоконтролю

1. Як компілятор визначає потрібну функцію із списку перевантажених?
2. Навести принцип визначення кількості параметрів операторної функції для перевантаження унарних та бінарних операторів.
3. Як повинен бути організований оператор індексації?
4. Як повинен бути організований оператор потокового уведення?
5. Як повинен бути організований оператор потокового виведення?
6. Що потрібно враховувати для реалізації перетворення типів?
7. Наведіть порівняльну характеристику віртуальних та звичайних функцій.
8. Поясніть призначення дружніх класів та дружніх функції.

5.6. Компютерний практикум № 6.

5.6.1. Тема

Використання шаблонних функцій.

5.6.2. Мета роботи.

Навчитися створювати власні шаблони функцій і користуватися стандартними, визначеними в бібліотеці **STL**.

5.6.3. Основні теоретичні відомості

Узагальнена (**шаблонна** функція, **шаблон** функції) визначає алгоритм, який можна застосувати до даних різних типів. Конкретний тип даних передається функції у вигляді параметра на етапі компіляції.

Синтаксис оголошення такої шаблонної функції наступний:

```
template <class T> тип ім'я_функції(список_параметрів)
{
// оператори тіла функції
}
```

У шаблоні функції може бути оголошено декілька формальних типів даних, а також використано параметри визначених раніше типів. Наприклад:

```
template <class T1, class T2>
void my_func(T1 a, T2 b, int c, double d, char s)
{
// оператори тіла функції
}
```

Конкретна версія узагальненої функції, створювана компілятором, називається *спеціалізацією* функції. Сам процес підстановки фактичного типу замість формального параметра-типа називається *конкретизацією* або *інстанціюванням*.

За наявності шаблонної функції компілятор може автоматично генерувати стільки її різних варіантів, скільки існує способів виклику цієї функції в програмі.

Під час виклику функції параметри шаблону можуть визначатися аргументами, що передаються в функцію (*виводяться* за типами фактичних аргументів). Цей процес здійснюється тільки за умови точної відповідності (з дозволом тривіального перетворення). Тобто автоматичне перетворення типів в

шаблонних функціях не дозволене, відповідність типів параметрів і аргументів повинна бути точною. Наприклад:

```
double b=34.0;
my_func(45.6, b, c, d, ':'); //підстановка double і double
```

Інколи буває необхідно явно вказати тип фактичного параметра-типа, тоді говорять про явне інстанціювання. Наприклад:

```
my_func <int, double>(a,  b, c, d, s);
```

Реалізацію будь-якої шаблонної функції варто починати з реалізації звичайної функції і перевірки коректності її роботи. Після цього конкретний тип в ній перевизначити як формальний і перевірити уже коректність роботи шаблонної функції для різних типів параметрів під час виклику.

Потрібно враховувати, що тип, використаний як фактичний параметр, має забезпечувати очікувану поведінку. Наприклад, здатність бути виведеним за допомогою оператора форматного виведення як це буде розглянуто нижче.

Шаблонні функції часто застосовують для роботи з так званими контейнерними класами. Клас, що містить набір елементів деякого типу, називається **класом-контейнером** або просто контейнером. Існує низка стандартних контейнерів, визначених в бібліотеці STL. Контейнером можна вважати також масиви або символьні рядки.

5.6.4. Приклад програми з використанням шаблонної функції

Програма призначена для виведення у стовпчик на екран n елементів масиву.

```
//користувацький клас для тесту в main()
class Point{
   int x,y;
public:
Point():x(0),y(0){};
Point(int a, int b):x(a),y(b){};
friend ostream& operator<<(ostream&,Point&);
};
//перевантажений оператор форматного виводу
ostream& operator<<(ostream & o,Point & p)
{
   return o<<'['<<p.x<<':'<<p.y<<']';
};
//сама шаблонна функція
```

```
template <class T>
void print (T a[], int n)
{
for (int i=0; i<n; i++) cout<<a[i]<<endl;
}
//тестовий приклад з викликами шаблонної функції
int _tmain(int argc, _TCHAR* argv[])
{
   int mas1[5]; double mas2[5]; Point mas3[3];
//заповнення масивів тестовими значеннями,
// Point - за умовчанням
for (int i=0; i<5; i++){    mas1[i]=i;    mas2[i]=i*10.1;  };
print (mas1, 3); print (mas2, 5); print( mas3,3);

   return 0;
}
```

5.6.5. Порядок виконання роботи

Побудувати шаблонну функцію відповідно варіанту. Зважити на те, що функція має коректно працювати незалежно від того, про який контейнер йдеться – масив, рядок або контейнер STL.

Передбачити тестовий приклад для ілюстрації коректності роботи, у якому реалізувати все необхідне для демонстрації двох або трьох викликів функції.

Перший виклик функції реалізувати для одного з вбудованих типів.

Другий – для одного з реалізованих у попередніх роботах класів вашого варіанту (якщо номер варіанту попередніх робіт не співпадає з номером варіанту шостої роботи).

Третій виклик – з використанням явного інстанціювання.

5.6.6. Варіанти завдань

Варіант 1.

Реалізувати шаблон функції для реверсування контейнера.

Варіант 2.

Реалізувати шаблон функції для видалення елемента контейнера.

Варіант 3.

Реалізувати шаблон функції для вставлення елемента контейнера.

Варіант 4.

Реалізувати шаблон функції для виведення на екран елементів контейнера.

Варіант 5.

Реалізувати шаблон функції для визначення кількості елементів контейнера із заданим значенням.

Варіант 6.

Реалізувати шаблон функції для пошуку першого входження до елементів контейнера елемента із заданим значенням.

Варіант 7.

Реалізувати шаблон функції для пошуку останнього входження до елементів контейнера елемента із заданим значенням.

Варіант 8.

Реалізувати шаблон функції для ініціалізації вмісту одного контейнера за допомогою іншого.

Варіант 9.

Реалізувати шаблон функції для ініціалізації вмісту контейнеру певним значенням з використанням

a) визначеного для типа параметра оператора присвоювання;

b) визначеної функції для генерації потрібного значення.

Варіант 10.

Реалізувати шаблон функції для реалізації циклічного зсуву значень елементів контейнера.

a) Справа наліво.

b) Напрямок зсуву задати як параметр.

Варіант 11.

Реалізувати шаблон функції для пошуку максимального елемента контейнера.

a) Скористатись визначеним для типа параметра оператором порівняння.

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 12.

Реалізувати шаблон функції для пошуку потрібного за значенням елемента контейнера.

a) Скористатись визначеним для типа параметра оператором порівняння «==».

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 13.

Реалізувати шаблон функції для пошуку елемента контейнера, що задовольняє потрібній умові.

a) Для порівняння передбачити можливість використання відповідної функції.

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 14.

Реалізувати шаблон функції для визначення кількості елементів контейнера, значення яких задовольняють певній умові.

a) Умову визначити фіксовану, наприклад >val.

b) Умову задати як шаблонну функцію, що передається як параметр.

Варіант 15.

Реалізувати шаблон функції для перевірки відсортованості елементів контейнера.

a) Скористатись визначеним для типа параметра оператором порівняння.

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 16.

Реалізувати шаблон функції для сортування контейнера з використанням алгоритму вибору.

a) Скористатись визначеним для типа параметра оператором порівняння «<».

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 17.

Реалізувати шаблон функції для сортування контейнера з використанням алгоритму перестановки.

a) Скористатись визначеним для типа параметра оператором порівняння «<».

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 18.

Реалізувати шаблон функції для визначення кількості елементів контейнера, що задовольняють певній умові. Умову визначити як шаблону функцію, що передається як параметр.

Варіант 19.

Реалізувати шаблон функції для пошуку першого входження до елементів контейнера елемента, значення якого визначено деякою умовою, заданою

a) фіксовано, у самій функції;

b) за допомогою деякої зовнішньої умовної функції.

Варіант 20.

Реалізувати шаблон функції для пошуку останнього входження до елементів контейнера елемента, значення якого визначено деякою умовою, заданою

a) фіксовано, у самій функції;

b) за допомогою деякої зовнішньої умовної функції.

Варіант 21.

Реалізувати шаблон функції для пошуку першого входження значень N перших елементів одного контейнера до складу другого контейнера. Елементи мають бути розташовані підряд.

Варіант 22.

Реалізувати шаблон функції для пошуку останнього входження значень N перших елементів одного контейнера до складу другого контейнера. Елементи мають бути розташовані підряд.

Варіант 23.

Реалізувати шаблон функції для визначення збіг значень перших N елементів двох контейнерів.

a) Скористатись визначеним для типа параметра оператором порівняння «==».

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 24.

Реалізувати шаблон функції для видалення поряд розташованих елементів з однаковими значеннями так, щоб залишився тільки один з них.

a) Скористатись визначеним для типа параметра оператором порівняння «==».

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 25.

Реалізувати шаблон функції для злиття елементів двох контейнерів шляхом збільшення розміру одного з них і видалення другого. Контейнер – список.

Варіант 26.

Задачу варіанту 25 реалізувати як включення частини одного з них перед N-ним елементом другого, зі зміною розмірів обох. Контейнер – список.

Варіант 27.

Реалізувати шаблон функції для злиття елементів двох відсортованих контейнерів шляхом збільшення розміру одного з них і видалення другого так, щоб не порушити відсортованості. Контейнер – список. Відсортованість початкових значень не перевіряти.

a) Скористатись визначеним для типа параметра оператором порівняння «<».

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 28.

Реалізувати шаблон функції для вставлення у відсортований контейнер деякого нового елемента так, щоб не порушити відсортованість. Початкову відсортованість контейнера не перевіряти.

a) Скористатись визначеним для типа параметра оператором порівняння «<».

b) Для порівняння передбачити можливість використання відповідної шаблонної функції.

Варіант 29.

Реалізувати шаблон функції для заміни усіх елементів контейнера зі значенням val1 на значення val2.

Варіант 30.

Реалізувати шаблон функції для видалення мінімального елемента контейнера.

Варіант 31.

Реалізувати шаблон функції для обміну мінімального та максимального елементів контейнера.

Варіант 32.

Реалізувати шаблон функції для видалення усіх елементів контейнера після N–го зі зміною розміру самого контейнера.

5.6.7. Запитання до самоконтролю

1. Як називається конкретна версія шаблонної функції, створювана компілятором?
2. Як називається процес генерації конкретної функції?
3. Скільки варіантів шаблонної функції генерує компілятор?
4. Перевантаження шаблонних функцій. Як компілятор визначає потрібну функцію із списку перевантажених?
5. Як можна використовувати аргументи шаблонів за умовчанням?
6. Які об’єкти можна використовувати в якості параметрів шаблонів?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

6.1. Базова

1. Страуструп Б. Язык программирования С++, 3-е изд. — М.: ”БИНОМ”, 1999. – 991 с.
2. Назарчук І.В. Об’єктно-орієнтоване програмування. Навчальний посібник – К.: «Політехніка». 2004. – 114с.
3. І.В.Назарчук, О.Л. Тимощук Програмування. Методичні вказівки до самостійної роботи студентів. – К.: НТУУ «КПІ» 2012 – 23с.[Електронне навчальне видання]
4. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення: ДСТУ 3008-95 (ГОСТ 7.32-91). – Чинний від 2006-01-01. К.: Держспоживстандарт України, 2006. – 213 с. – (Національний стандарт України).

6.2. Допоміжна

1. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. 2-е изд. — М.: «БИНОМ». 1999. — 560 с.
2. Либерти Дж. С++. Энциклопедия пользователя. Пер. с англ. – К.: «ДиаСофт». 2000. – 584с.
3. Вайнер Р., Пинсон Л. С++ изнутри. – Киев: «ДиаСофт». 1993. - 304с.
4. Пол А. Объектно-ориентированное программирование на С++. 2-е изд.. — М.: «БИНОМ». 1999. — 462 с.
5. Либерти Дж. Освой самостоятельно С++ за 21 день: Уч. пос. – М.: Издат. дом «Вильямс». 2001. - 816с.
6. Дейтел Х., Дейтел П. Как программировать на С++. – М.: «БИНОМ». 2001. – 1152с.

6.3. Інформаційні ресурси

1. Електронний кампус НТУУ «КПІ ім. І. Сікорського» [сайт] / Єдине інформаційне середовище НТУУ «КПІ ім. І. Сікорського», 2011-2017. – Режим доступу: http://campus.kpi.ua
2. Библиотека MSDN [Електронний ресурс]//Microsoft: [сайт]/ MSDN Library Numerical, 2012. – Режим доступу: http://msdn.microsoft.com/library/default.aspx
3. База знань Microsoft Developer Network (MSDN) [Електронний ресурс]. – Режим доступу : http://msdn.microsoft.com/ru-ru.

Додаток А. Титульна сторінка до звіту

НТУУ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА ММСА

Комп'ютерний практикум № N
Варіант № NN
з дисципліни:
Програмування та алгоритмічні мови

Виконав:
Студент І курсу групи КА-NN
Nnnnnnnnn Nnnnn
Перевірив: Nnnnnnnn N.N.