ЛАБОРАТОРНА РОБОТА №2

Тема: Розробка програм з використанням класів з конструктором і деструктором.

Мета: Набуття навиків в проектуванні найпростіших класів, розробка найпростіших програм їх використання.

Порядок виконання роботи

1. Ознайомитися з теоретичними основами розробки та правилами використання конструктора і деструктора.
2. Розробити клас для роботи з текстовими файлами з декількома конструкторами та деструктором, з функціями наповнення файлу, визначення його розміру, функцією виведення змісту файлу та функцією відповідно до завдання 1.
3. Розробити програму для використання класу з пункту 2, для роботи з двома об’єктами (різними текстовими файлами).
4. Розробити клас для роботи з бінарними файлами з декількома конструкторами, деструктором, функціями наповнення файлу, визначення його розміру, функцією виведення змісту файлу та функцією відповідно до завдання 2.
5. Розробити програму для використання класу з пункту 4 для роботи з трьома об’єктами (різними бінарними файлами).
6. Розробити 2-3 теста для перевірки правильності роботи розроблених програм з п. 3, 5.
7. Оформити звіт до лабораторної роботи.

Завдання 1

Варіант 1. Розробити метод-член класу для підрахування кількості рядків в текстовому файлі.

Варіант 2. Розробити метод-член класу для підрахування кількості слів в текстовому файлі.

Варіант 3. Розробити метод-член класу для порівняння двох текстових файлів та виведення номеру рядка та позиції символу, де вони відрізняються.

Варіант 4. Розробити метод-член класу для підрахування кількості разів появи заданого слова в текстовому файлі.

Варіант 5. Розробити метод-член класу для до запису рядка в початок текстового файлу.

Варіант 6. Розробити метод-член класу для до запису рядка в кінець текстового файлу.

Варіант 7. Розробити метод-член класу для до запису рядка в середину текстового файлу.

Варіант 8. Розробити метод-член класу для видалення рядка в середині текстового файлу (новий файл складається не менше ніж з трьох рядків).

Варіант 9. Розробити метод-член класу для видалення рядка з початку текстового файлу (новий файл складається не менше ніж з трьох рядків).

Варіант 10. Розробити метод-член класу для до запису слова в початковий рядок текстового файлу (новий файл складається не менше ніж з трьох рядків)

Завдання 2

Варіант 1. Розробити метод-член класу для до запису заданої кількості даних у початок бінарного файлу (файл повинен складатися не менше ніж з п’яти компонентів).

Варіант 2. Розробити метод-член класу для до запису заданої кількості даних у середину бінарного файлу (файл повинен складатися не менше ніж з п’яти компонентів).

Варіант 3. Розробити метод-член класу для до запису заданої кількості даних у кінець бінарного файлу (файл повинен складатися не менше ніж з п’яти компонентів).

Варіант 4. Розробити метод-член класу для видалення заданої кількості даних з початку бінарного файлу.

Варіант 5. Розробити метод-член класу для видалення заданої кількості даних з середини бінарного файлу з заданої позиції.

Варіант 6. Розробити метод-член класу для видалення заданої кількості даних з кінця бінарного файлу.

Варіант 7. Розробити метод-член класу для копіювання заданої кількості даних з заданої позиції бінарного файлу.

Варіант 8. Розробити метод-член класу для видалення заданої кількості даних з заданої позиції бінарного файлу і вставки їх з заданої позиції у новий файл.

Варіант 9. Розробити метод-член класу для перезапису змісту бінарного файлу з кінця в початок в оберненому порядку.

Варіант 10. Розробити метод-член класу для створення нового файлу, який містить цілі дані з заданого файлу, кратні одинадцяти.

Теоретичні відомості

Конструктори і деструктори

Зазвичай у найпростіших програмах існує два способи визначення цілочисельної змінної:

1) можна визначити змінну, а потім присвоїти її деяке значення, наприклад:

```
int weight;
weight = 5;
```

2) можна визначити змінну і негайно її ініціалізувати, наприклад:

```
int   w = 7;
```

при цьому операція ініціалізації сполучить у собі визначення змінної з присвоєнням початкового значення.

Як же ініціалізувати змінні-члени класу ?

Для цього в ООП використовуються дві спеціальні функції:

1) Конструктор - це метод класу, ім'я якого збігається з ім'ям класу, призначений для створення й ініціалізації - класу.

2) Деструктор - це метод класу, ім'я якого складається з (~) - тільди й імені класу, призначений для видалення з пам'яті об'єкти класу, що відробили, і звільнення виділеної для них пам'яті.

Наприклад, для класу `CTime` ім'я деструктора `~CTime ( );`

Правила роботи з конструкторами і деструкторами

1. У класах може бути оголошено декілька конструкторів для автоматичної ініціалізації об'єкта класу при його створенні.

2. У класах може бути оголошений тільки один деструктор, який викликається для очищення пам'яті при виході об'єкта класу з області видимості.

3. Конструктори і деструктори схожі на звичайні функції-члени, але вони рідше викликаються безпосередньо в операторах програми.

4. C++ автоматично викликає конструктори і деструктори для ініціалізації й очищення об'єктів класу.

5. Конструктори з'являються як функції члени, що не повертають ніяких значень, з довільним числом параметрів будь-якого типу.

6. Деструктори не приймають ніяких аргументів і не повертають ніяких значень.

7. Конструктори, як правило, з'являються у відкритій секції.

8. Якщо конструктор оголошений без параметрів, він викликається за замовчуванням.

9. Деструктор викликається автоматично, коли об'єкт класу виходить з області видимості.

10. Конструктор викликається автоматично в момент створення об'єкта чи класу за бажанням програміста.

Приклад використання конструктора і деструктора

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

class  CTime
{


   private:
```

Схована область

```
long dt; // Дата і час в секундах від 1 січня 1970 року.
char * dts; // Представлення (містить адреси) дати
                у вигляді рядка
void Delete Dts ( void );
```

// видаляє показник dts; *закрита функція*

```
   public:
    CTime ();  // Конструктор 1 – викликається за замовченням
    CTime (int m,int d=-1, int y=-1, int h2=-1, int
                                        min=-1);
```

// Конструктор 2 - перегружений

```
        ~ CTime ();  // Деструктор
 void Display(void) {cout<<ctime(&dt;}
 void  GetTime(int  &m,int  &d,int  &y,  int  &h2,  int
&min);
    void Set Time(int m=-1,int d=-1,int y=-1,int h2=-
1,int min=-1);
   const char*Get Stime (void);
   void Change Time (long n minuts)
         {dt+=(n minutes×60);Delete Dts();}
};
```

Обов'язки типового конструктора обмежені присвоюванням початкових значень даним-членам класу, виділенням пам'яті, використаної об'єктом класу і т. д. C++ автоматично викликає конструктор для ініціалізації об'єкта класу `CTime`, отже, усі такі об'єкти гарантовано мають ініціалізований показник `dts`.

Мета конструктору - забезпечити ясно визначену ініціалізацію об'єктів класу.

Приклад використання конструкторів

```
main ( )
{
CTime  t1;
CTime  t2 (8);
CTime  t3 (8, 1);
CTime  t4 (8, 1, 1999);
CTime  t5 (8, 1, 1999, 8);
CTime  t6 (8, 1, 1999, 8, 30);
 t1. Display ();
 t2. Display ();
 ……
 t6. Display ();
 return ();}
```

Приклад використання деструктора

```
int main ( )
 {
      func ();
      return ();
 }
      void func (void)
  {
// Конструктор викликається автоматично за замовчуванням щораз  при виклику функції
      CTime  today;
      Const char * sp;
 //Результат функції зберігається в локальному символьному показнику
                sp
```

```
    sp = today. Get STime ();
    Cout <<  " First time: " <<  sp;
    sp = today. Get STime  ();
    Cout <<  " Second time: " <<  sp;
}
```

Об'єкт класу `Ctime today` оголошений в середині класу, тобто він локальний в області видимості функції `func.` При описанні функцій-членів класу деструктор може мати вигляд:

`CTime:: ~ CTime ( );` - Деструктор

```
{
    delete  dts;
  }
Const char * CTime :: Get STime (void)
 {
```

*//*перевірка чи не містить *dts* шлях *return dts;* інакше викликаються функції *Strdup ( ) і CTime ( )* для перетворення поточної дати і часу в рядок зі збереженням адреси рядка в символьному покажчику *dts* - закритому члені класу.

```
if (dts)
    dts =  strdup (CTime (&dt));
    return dts;
 }
```

Тоді наступні виклики `GetSTime`() будуть використовувати раніше створений рядок. Це скорочує кількість виділень простору в області, що динамічно розподіляється, і швидкодія програми. Клас `CTime` володіє показником *dts*, що посилається на виділений блок пам'яті, та керує розподілом цієї пам'яті. При виході об'єкта класу з області видимості, область пам'яті звільняється деструктором автоматично.

Приклад виконання завдання 1

Варіант 10. Розробити описовий алгоритм, схему алгоритму і написати метод для підрахування кількості слів в текстовому файлі.

Дана програма містить один клас `CSlov`, що працює з текстовим файлом`*f`. Цей клас містить конструктор `CSlov(),` за допомогою якого файл відкривається автоматично після ініціалізації даних в головній

програмі, деструктор `~CSlov()`, що автоматично викликається на заключному етапі виконання програми і виконує закриття файлу. Також клас містить функцію `Procesing()`, що призначена для обробки (для підрахування кількості слів) текстового файлу.

Структура класу має такий вигляд:

```
class CSlov{
 FILE *f; // Вказівник на файл
 int k;
 char c,c1;
public:
 CSlov(); // Конструктор
~CSlov(); // Деструктор
void Procesing();
};
```

Лістинг програми:

```
#include <stdio.h>
#include<stdlib.h>
#include <conio.h>
#define F "C:/TC/lab/lab4_1.txt"
 class CSlov{
  FILE *f;
     int k;
     char c,c1;
    public:
     CSlov();
     ~CSlov();
     void Procesing();
    };
    CSlov::CSlov()
     {
      clrscr();
      f=fopen(F, "r");
      if(f == NULL)
       {
        printf("Can't open file %s for read.", F);
        getch();
        exit(0);
       }
     }
```

```
CSlov::~CSlov()
 {
  fclose(f);
 }
void CSlov::Procesing()
 {
  k=0;c1='z';
  while(!feof(f))
  {
   c=fgetc(f);
   if(c==' ')
    {
     if(c==c1)
      {
       printf("%c",c);
       continue;
      }
     c1=c;
     k++;
    }
   c1=c;
   printf("%c",c);
  }
  printf("\n\nkolichestvo slov=%d",k+1);
  getch();
 }
void main()
{
 CSlov obj;
 obj.Procesing();
}
```

Тестування

Для перевірки правильності роботи програми введемо до текстового файлу деяке число слів, кількість яких заздалегідь порахуємо. Наприклад створимо файл такого виду:

```
Ckotland  Goverla gora    Nevada reka  voda trava figura   helo      klaviatura
```

Наочно видно, що в цьому файлі знаходиться 10 слів.

Перевіримо, який результат виводить програма:

```
    Ckotland  Goverla gora    Nevada reka  voda trava
figura   helo      klaviatura
    kolichestvo slov=10
```

Приклад виконання завдання 2

Варіант 2. Розробити метод (функцію) для до запису даних у середину існуючого двійкового файлу з заданої позиції.

Дана програма містить один клас `CVstavka`, що працює з бінарним файлом `*f`. Цей клас містить конструктор `CVstavka()`, за допомогою якого файл відкривається автоматично після ініціалізації даних в головній програмі, деструктор `~CVstavka()`, що автоматично викликається на заключному етапі виконання програми і виконує закриття файлу. Також клас містить функцію `Procesing()`, що призначена для обробки (для до запису даних у середину існуючого двійкового файлу з заданої позиції) бінарного файлу.

Структура класу має такий вигляд:

```
class CVstavka{
 FILE *f;   // Вказівник на файл
char s[100],s1[100],s2[200];
 int l,l1,l2,k;
public:
 CVstavka();   // Конструктор
 ~CVstavka(); // Деструктор
 void Procesing();};
```

Лістинг програми:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#define F "C:/TC/lab/lab4_2.dat"
class CVstavka{
 FILE *f;
 char s[100],s1[100],s2[200];
 int l,l1,l2,k;
```

```
  public:
   CVstavka();
   ~CVstavka();
   void Procesing();
  };
  CVstavka::CVstavka()
   {
    clrscr();
    f=fopen(F, "r+b");
    if(f == NULL)
     {
      printf("Can't open file %s for read.", F);
      getch();
      exit(0);
     } }
  CVstavka::~CVstavka()
   {
    fclose(f);
   }
  void CVstavka::Procesing()
   {
    fseek(f,0,0);
    fread(s1,sizeof(char),50,f);
    puts(s1);
    printf("\n\nvvedite frazy dla sohronenia v
file\n");
    gets(s);
    l=strlen(s);
    printf("vvedite pozitsiu sohranenia stroki\n");
    scanf("%d",&k);
    strncpy(s2,s1,k);
    l2=strlen(s2);
    fseek(f,k,0);
    fread(s1,sizeof(char),50-k,f);
    l1=strlen(s1);
    fseek(f,0,0);
    fwrite(s2,sizeof(char),l2,f);
    fwrite(s,sizeof(char),l,f);
    fwrite(s1,sizeof(char),l1-k,f);
    fseek(f,0,0);
    fread(s1,sizeof(char),50+l,f);
    printf("\n\n");
    puts(s1);
    getch();
```

```
 }
void main()
{
 CVstavka obj;
 obj.Procesing();
}
```

Тестування

Для тестування програми спочатку створимо бінарний файл за допомогою програми. Нехай цей файл буде такого виду:

```
norvegia doroga solntse robot poroh fontan
```

Тепер в даний файл вставимо слово, наприклад robota, на позицію починаючи з 13 байта. Шляхом ручної вставки отримаємо такий перетворений файл:

```
norvegia dororobotaga solntse robot poroh fontan
```

Введемо ці ж дані у програму і перевіримо результат. Результати тестуванні програми представлені ніжче.

```
norvegia doroga solntse robot poroh fontan
vvedite frazy dla sohronenia v file
robota
vvedite pozitsiu sohranenia stroki
13
norvegia dororobotaga solntse robot poroh fontan
```

Контрольні запитання

1. Специфікатори доступу.
2. Як описуються функції-члени класу?
3. Що таке екземпляр класу?
4. Що відносять до даних-членів класу?
5. Як здійснюється ініціалізація даних-членів класу?
6. Що таке конструктор? Наведіть приклад.
7. Що таке деструктор? Наведіть приклад.
8. Правила роботи з конструктором. Наведіть приклади.

9. Правила роботи з деструктором. Наведіть приклади.

10. Правила програмної організації конструктора в С++.

11. Правила програмної організації деструктора в С++.

12. Призначення конструктора.

13. Призначення деструктора.

14. Структура конструктора.

15. Структура деструктора.

16. Вимоги до вхідних даних конструктора та деструктора.

17. Вимоги до вихідних даних конструктора та деструктора.

18. Навести приклади використання декількох конструкторів в одному класі.