ЛАБОРАТОРНА РОБОТА №9

Тема : Технології КОНСТРУЮВАННЯ БАГАТОФАЙЛОВИХ ПРОГРАМ на C/C++

Мета роботи:

- Вивчення основних директив препроцесорної обробки програм.
- Отримання практичних навиків у роботі зі структурами, функціями і модульною структурою програм.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Директиви препроцесорної обробки

Директиви - інструкції препроцесора. Обробка програми препроцесором здійснюється перед її компіляцією. Призначення препроцесорної обробки:

- включення у файл, що компілюється, інших файлів,
- визначення символічних констант і макросів,
- встановлення режиму умовної компіляції програми і умовного виконання директив препроцесора.

Синтаксис:

- директиви повинні починатися з символу "#".
- після директив препроцесора не ставиться " ;".

Основні директиви:

#include - включає копії вказаного файла в те місце програми, де знахо-диться ця директива;

#define - здійснює **макропідстановку** - заміняє всі входження ідентифікатора у програмі на текст, що слідує в директиві за ідентифікатором;

#undef - анулює визначення символічних констант і макросів;

#if, #elif, #else, #endif - здійснюють умовну компіляцію;

#ifdef і ***#ifndef*** - використовуються замість #zf- *defined* (...);

#line - повідомляє компілятору про зміну імені програми і порядку нумерації рядків;

#error - вказує компілятору, що треба надрукувати повідомлення про помилку і перервати компіляцію (зазвичай використовують у середині умовних директив);

#pragma - дозволяє керувати можливостями компіляції. Існує і ряд інших директив.

Визначені макроси ANSI:

__DATE_ - рядок у формі mmm.dd.yyyy - дат створення даного файлу;

__TIME_ - час початку обробки поточного файлу у форматі hh:mm:ss ;

__FILE__ - ім'я поточного файлу;

__LINE_ - номер поточного рядка;

__STDC_ - визначений, якщо встановлено режим сумісності з ANSI.

Багатофайлові проекти

У мові C/C++ модулі на рівні синтаксису не виділяються. Реалізація модулів можлива з використанням роздільної компіляції та файлів заголовків. Програма на С може складатись з декількох файлів, кожний з яких вміщує підпрограми та описи. Головним файлом вважається той, що містить функцію *main()*.

Роздільна компіляція - це обробка компілятором кожного файлу програми окремо. Файли заголовків мають розширення ".h" та вміщують описи глобальних констант, типів, змінних, підпрограм. Описи констант, типів, змінних не відрізняються від розглянутих вище. Для підпрограм у файлах заголовків наводять тільки заголовок, після якого ставлять ";":

```
extern t f(t1 x1, ... ,  tn n);
```

де t, t_i - тип, x_i - змінні.

Ключове слово *extern* вказує на те, що підпрограма реалізована у зовнішньому файлі.

Файли заголовків підключають до інших файлів за допомогою директиви *#include*. Наприклад,

```
#include "myfile.h"
```

Ми вже знайомі з цією директивою та постійно користувались нею для підключення стандартних бібліотек. Різниця із підключенням власних файлів заголовків в тому, що для стандартних бібліотек вико ристову-
ють символи "< > " замість лапок: `#include <stdio.h>` При реалізації модулів для кожного модуля у С створюють два файли з однаковим іменем та з розширеннями *".h"* та *".cpp"*. Файл заголовку з розширенням *.h* грає роль інтерфейсної частини модуля, а файл з розширенням *".cpp"* - частини реалізації модуля. Для використання модуля треба підключити відповідний файл заголовка за допомогою *#include*.

Приклад розробки багатофайлового проекту

Задача. Необхідно забезпечити роботу з таблицею, яка містить інформацію про студентів.

Формалізація задачі

1. Програма повинна складатися, як мінімум, з двох файлів:
   а) у першому повинна знаходитися головна програма, функції якої - вибір у діалоговому режимі однієї з вищевказаних дій;
   б) у інших - функції, які безпосередньо реалізують ці дії.
2. Визначаємо змінні, загальні для усіх модулів програми.

По-перше, необхідно оголосити структуру, яка являє собою рядок таблиці:

```
struct stud
{     char name[25];   // Ф.І.Пб.
      chat gend;   // стать
      int year;    // рік народження
```

```
        float sq;    // вага }
```

Оскільки необхідно створити програмний засіб з декількох файлів, виділимо це оголошення в окремий файл-заголовок, який буде включатися (`#include`) у всі файли-програми. Це забезпечить однаковість цієї структури в усіх модулях програми. Для скорочення запису визначимо скорочені імена для структури і її розміру:

```
#define STUD struct stud
#define SSTUD sizeof(struct stud)
```

Дані таблиці будуть знаходитися в масиві `STUD st[N]`. Розмірність масиву визначимо через макроконстанту:

```
#define N 100
```

Окрім того, що цей масив повинен розміщуватись в оперативній пам'яті, він ще повинен бути доступним для декількох модулів програми. Тобто, в одному модулі він повинен бути оголошений поза блоками, а в інших (які до нього звертаються) - описаний як зовнішній.

```
extern STUD st[];
```

Те саме стосується і змінної, яка міститиме кількість заповнених елементів у масиві. Вона повинна бути визначена в одному модулі:

```
int n;
```

а в інших описана як зовнішня:

```
extern int n.
```

3. Модулі програми

Окремий модуль програми буде складати головна функція програми (це загальноприйнятий підхід). В ньому розміщається програма-монітор, яка керує порядком виконання інших функцій програмного засобу (він буде називатись `mainProg`).

Розробимо ще два модуля:

- один такий, що вимагає звернення до глобальних змінних програми;
- другий такий, що його функції не залежать від глобальних змінних.

Для кожного з модулів зробимо ще і заголовочний файл. Це модулі `Prog1` і `Prog2`.

Розробка модуля mainProg

Даний модуль міститиме головну функцію, яка керуватиме порядком виконання інших функцій програмного засобу. В цьому модулі необхідно зібрати глобальні змінні програми і функцію `main()`. Локальні змінні функції `main()`:

`op` - код тієї дії, яку повинна виконати програма;

`num` - номер запису для тих дій, які його потребують;

`eoj` - ознака кінця роботи.

3.2. Розробка модуля Prog1

У файл `Prog1.h` поміщаємо усі прототипи функцій, а у файлі `Prog1.cpp` - опис цих функцій. При цьому підключаємо потрібні заголовочні файли.

Тут зібрані функції, які мають доступ до глобальних змінних - масиву і кількості змінних у ньому. Тому саме тут міститься оголошення зовнішніх змінних, а також визначення функцій:

`check_number()` - перевірка правильності номеру елемента (0 - вірний, 1 - ні),

`del_item()` - видалення елементу із заданим номером з масиву,

`show_all()` - виведення на екран усього масиву. Ця функція викликатиме інші 3 функції, які не залежать від глобальних змінних і будуть описані у модулі `Prog2: print head(),show row(),print line()`.

3.3. Розробка модуля Prog2

Тут визначені функції, які не залежать від загальних змінних.

`get_number()` – введення з клавіатури номера елемента масиву;

`enter_data()` – введення з клавіатури значень для одного елемента масиву;

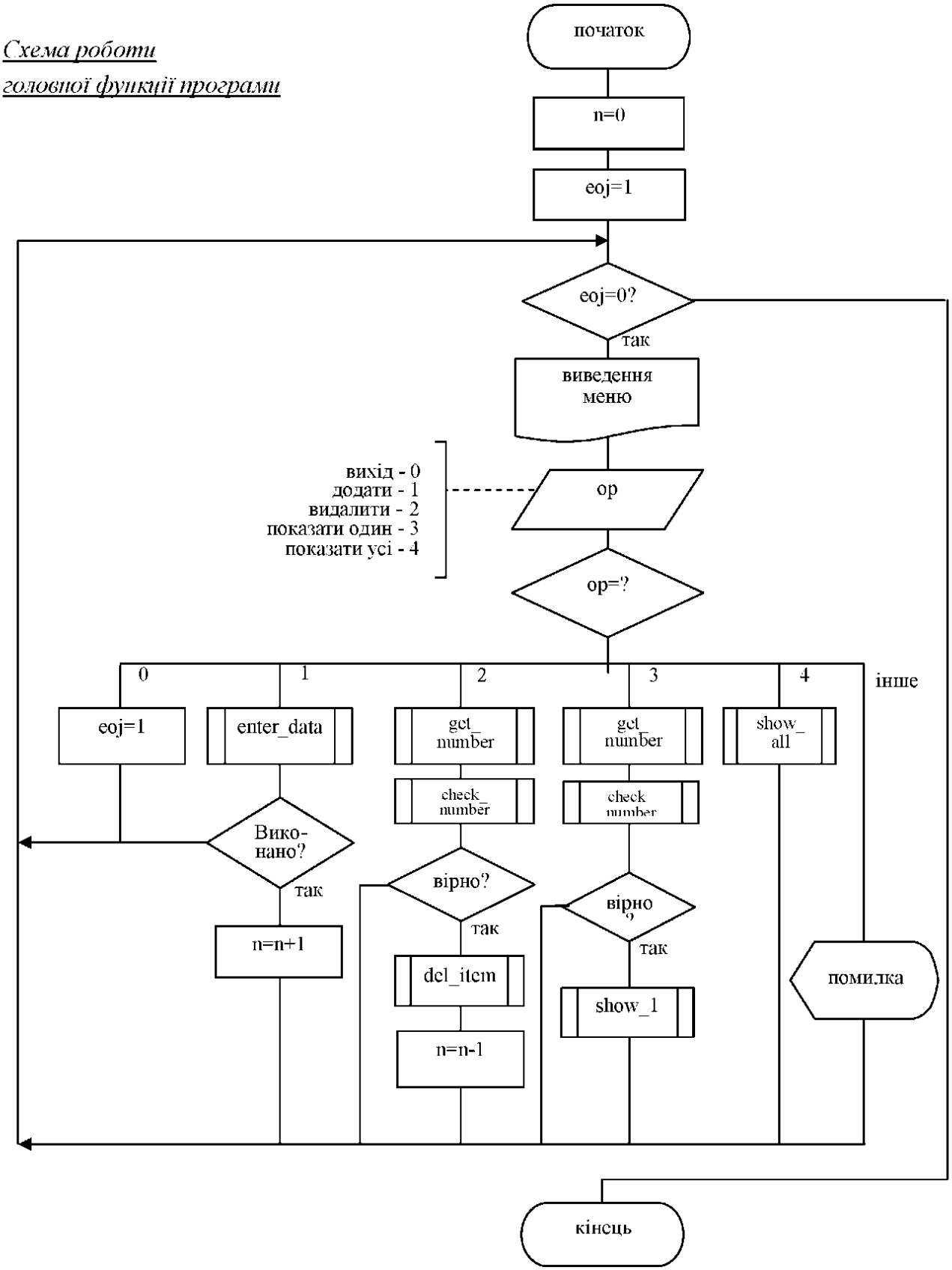
`show_1()` – виведення на екран одного елемента масиву;

`print_head()` – виведення заголовка таблиці;

`show_row()` – виведення одного рядка таблиці;

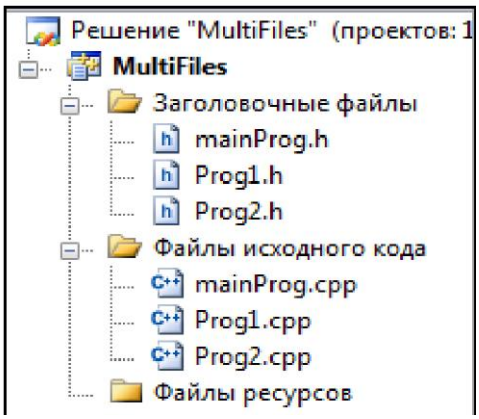
`print_line()` – виведення нижньої лінії таблиці.

Схема роботи головної функції програми



Створення програмного проекту

Проект, що реалізує програмний засіб, буде складатися з шести файлів.



Файл `mainProg.h`

```
#define STUD struct stud
#define SSTUD sizeof(struct stud)
struct stud {
  char name [25];    // Ф.І.Пб.
  char gend;   // стать
  int year;    // рік народження
  float w;    };// вага
```

Файл `mainProg.cpp`:

```
#include "mainProg.h"
#include "Prog1.h"
#include "Prog2.h"
#define N 100
STUD stud[N]; /* масив-таблиця */
int n=0;       /*   кількість елементів у масиві */
int main(void)  {
  int op;     /*    операція */
  int num;    /*    номер елементу */
  char eoj; /*  познака кінця */
  setlocale(0,"");
  for (eoj=0;  !eoj; )  {/* виведення меню */
     printf("1 - Додати елемент\n");
     printf("2 - Видалити елемент\n");
     printf("3 - Показать елемент за номером\n");
     printf("4 - Показати все\n");
     printf("0 - Вихід\n");
     printf("Введіть > ");
```

```
        scanf("%d",&op); /* вибір з меню */
        switch(op) {
         case 0: eoj=1;break; /* вихід */
         case 1: if (!enter_data(stud+n)) n++;/* додати */
                 break;
         case 2: if(!check_number(num=get_number())){/* видалити*/
                 del_item(num); n-- ; } break;
         case 3: if(!check_number(num=get_number()))/*показати один */
                 show_1(stud+num-1);
                 break;
         case 4: show_all();/* показать все */
                 break;
         default: printf ("Неправильна операція^");
                  break;
         }  /* switch */
        if (op) {
         printf("Натисніть будь-яку клавишу\n");
         getch();
         } /* if */
        } /* for */
        return 0;
    }    /* main */
```

Файл `Prog1.h`

```
int check_number(int);
void del_item(int);
void show_all(void);
```

Файл `Prog1.cpp`

```
#include <stdio.h>
#include <memory.h>
#include "mainProg.h"
#include "Prog2.h"
extern STUD stud[];
extern int n;
int check_number(int a) {/**** перевірка номера елемента ****/
 if (a<1) {printf("Мінімальний номер : 1\n");  return -1;}
 if (a>n) {printf("Максимальний номер : %d\n",n);return -1;}
 return 0;
}
```

```
void del_item(int m) {/**** видалення елемента ****/
  int i;
  for (; m<n; m++) memcpy(stud+m-1,stud+m,SSTUD);
}
void show all() {/**** вивід всього масиву ****/
  int i;
  print_head();
  for (i=0; i<n; i++) show_row(stud+i);
  print_line();
}
```

Файл Prog2.h

```
void    print_head(void);
void    print_line(void);
void   show_row(STUD   *);
int get_number(void);
void show_1(STUD *);
int enter_data(STUD *);
```

Файл Prog2.cpp:

```
#include <stdio.h>
#include <string.h>
#include "mainProg.h"
#include "Prog2.h"
int get_number() {/**** введення номера ****/
  int b;
  printf("Введите номер>");
  scanf("%d",&b);
  return b;
}
int enter data(STUD *s) {/**** введення даних про одного студента ***/
  float f;
   printf("Введите имя, пол, год рождения, вес:\n");
   scanf("%s %c %d %f",s->name,&s->gend,&s->year,&f);
   s->w=f;
   if (!strcmp(s->name,"***"))  return -1;
   return 0;
}
void show_1(STUD *s){/**** виведення даних про одного студента ***/
printf("\n П.І.Б.     : %s\n",s->name);
```

```
  printf("Стать     : %c");
  switch(s->gend)
  {
       case  'м':  printf("(чол.)");  break;
       case 'ж' : printf("(Жін.)"); break;
  }
  printf("\nРік народження : %d\n",s->year);
  printf("Вага      : %6.2f\n",s->w);
  print_line();
}
void show_row(STUD *s)   {/**** виведення рядка таблиці ****/
printf("| %9s  |     %c     |     %3d | %-5.1f |\n",
   s->name,s->gend,s->year,s->w);
}
void print_line()
{
  printf("----------------------------------------\n");
}
void print_head()
{
 print_line();
 printf("|        СПИСОК СТУДЕНТІВ           |\n");
 printf("|-----------------------------------|\n");
 printf("|  Ім'я   | Стать | Рік        | Вага|\n");
 printf("|         |       | народження |     |\n");
 printf("| -------| -- ---|-- - -------|  ---|\n");
}
```

Результати роботи програми (фрагмент)

```
1 - Додати елемент
2 - Видалити елемент
3 - Показать елемент за номером
4 - Показати все
0 - Вихід
Введіть >  1
Введите имя, пол, год рождения, вес:
Sidorov m 1992 65


Нажмите любую клавишу
1 - Додати елемент
2 - Видалити елемент
3 - Показать елемент за номером
4 - Показати все
0 - Вихід
Введіть >  4
-----------------------------------------
|            СПИСОК СТУДЕНТІВ           |
|---------------------------------------|
|    ?м'я  | Стать |    Рік    |  Вага  |
|          |       |народження |        |
|----------|-------|-----------|--------|
|  Petrova |   w   |      1990 | 58,0   |
|  Sidorov |   m   |      1992 | 65,0   |
-----------------------------------------

Нажмите любую клавишу
```

Порядок виконання роботи

1. За результатами виконання *лабораторних робіт №№7-8* розробити у вигляді багатофайлового проекту програму, яка буде відповідати таким вимогам:
   а) заповнення масиву (таблиці) даних повинно здійснюватись із зовнішнього файлу;
   б) програма повинна забезпечувати роботу з даними у різних режимах:
   - додавання нових рядків у таблицю;
   - видалення рядка із заданим номером з таблиці;
   - виведення інформації, яка зберігається у рядку с заданим номером;
   - виведення на екран усієї таблиці.
2. Підготувати звіт:
   - варіант і текст завдання;
   - загальна схема функціонування програми;
   - схема взаємозв'язків функцій у програмі (з нанесенням на схему вхідних і вихідних даних для кожної з функцій;
   - лістинг програми (по файлах);
   - результати виконання з відображенням результатів роботи кожного режиму;
   - висновки.

Контрольні питання

1. Що таке директиви препроцесора?
2. Який синтаксис написання директив препроцесора?
3. Коли здійснюється обробка програми препроцесором?
4. Наведіть основні директиви препроцесора і поясніть їх дію.
5. Що таке "препроцесорна обгортка"? У яких випадках її використовують?
6. Що таке макровизначення? Як вони працюють?
7. У чому переваги використання багатофайлових програм?
8. Які класи пам'яті ви знаєте?
9. У яких випадках використовують регістровий тип даних?
10. Що означають локальні змінні? До якого класу пам'яті вони належать?
11. До якого класу пам'яті можна віднести формальні параметри функцій?
12. Як і для чого використовують зовнішні глобальні змінні?компіляція
13. Що таке роздільна компіляція?
14. Що рекомендується зберігати у заголовочних файлах, а що - у програмних файлах?