

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет "Запорізька політехніка"

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи із дисципліни

"Програмування "

для студентів спеціальності 123 "Комп'ютерна інженерія"
денної форми навчання

Структури та об'єднання

2020

Методичні вказівки до виконання лабораторної роботи із дисципліни "Програмування" для студентів спеціальності 123 "Комп'ютерна інженерія" денної форми навчання. Структури та об'єднання / Укл.: Н.В. Луценко, О.В. Польська. – Запоріжжя: НУ "Запорізька політехніка", 2020. – 14 с..

Укладачі:

Н.В. Луценко, ст. викладач
О.В. Польська, ст. викладач

Рецензент:

Р.К.Кудерметов, к.т.н., доцент

Відповідальний за випуск

Н.В. Луценко, ст. викладач

Затверджено
на засіданні кафедри КСМ
Протокол № 5 від 29.01.2020

Рекомендовано до видання
на засіданні НМК КНТ
Протокол № 6 від 30.01.2020

ЗМІСТ

Порядок виконання лабораторних робіт.....	3
Загальні та теоретичні відомості	4
Лабораторна робота	
Структури та об'єднання.	7
Контрольні питання до лабораторної роботи.....	13
Рекомендована література.....	14

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота складається з декількох програм, завдань до них і контрольних питань. Студент повинен розробити, відпрацювати і зберегти у робочому каталозі програму, що наведена, й програму, розроблену згідно заданого варіанта, та проаналізувати результати їхньої роботи.

За результатами виконання кожної лабораторної роботи студент складає й захищає письмовий звіт, вимоги до якого викладені нижче.

Зміст звіту

1. Титульний лист, виконаний відповідно до СТП.
2. Мета роботи.
3. Відповіді на контрольні питання.

ЗАГАЛЬНІ ТА ТЕОРЕТИЧНІ ВІДОМОСТІ

Структури – це сукупність поіменованих об'єктів у загальному випадку різних типів. Кожна структура включає в себе один або декілька об'єктів (змінних, масивів, покажчиків, структур), які називають **елементами** (полями) структури.

Структурний тип (*шаблон*) визначає, скільки елементів і якого типу входять в структуру. Визначення шаблону структури починається із слова **struct**, за яким у фігурних дужках розміщуються описи елементів, що входять в структуру. Шаблон структури має права типу. Під шаблон пам'ять не виділяється. Наприклад:

```
struct STR{ int a[2]; float r};
```

Після визначення структурного типу (шаблону) за його допомогою можна визначити конкретні структури. Наприклад:

```
STR st1, st2[3], *ptrst=&st1;
```

Пам'ять для структур виділяється згідно структурного типу, тобто для структури **st1** виділиться **12** байт ($4 \cdot 2 + 4$), для масиву структур **st2** – **36** байт, для покажчика на структуру **ptrst** – **4** байти.

При визначенні структури можлива її ініціалізація, тобто при-власнення початкових значень елементам структури. Для цього після визначення ставиться знак '=' та у фігурних дужках через кому задаються значення елементів згідно структурного типу. Наприклад:

```
STR st3={78,20,4.9};
```

```
STR st4[2]={ {23,20,4.4}, {15,19,4.3} };
```

Для звернення (доступу) до елементів структури використовуються уточнені імена. Наприклад:

```
st1.a[0]=19; st1.r=4.3;
```

Для звернення до елементів масиву структур використовується індекс, який вказують у квадратних дужках, наприклад:

```
st2[0].a[0]=20; st2[1].r=4.7;
```

При зверненні до елементів структури, зв'язаної з покажчиком, використовується операції стрілка або розіменування покажчика з ім'ям елемента структури після крапки. Нижче наведені обидві форми:

```
ptrst→a[1]=18; (*ptrst).r=4.5;
```

Можна створювати і знищувати структури з динамічною тривалістю існування. Для цього використовуються функції **malloc()** і **free()** або операції **new** і **delete**. Наприклад:

```

STR *st; // визначення покажчика на структуру типа STD
st=(STR*)malloc(sizeof(STR)); //Запит пам'яті
//st=new STR; //Запит пам'яті
if(st==NULL) // перевірка виділення пам'яті
    {puts("Error");return -1;}
scanf("%d %d %f",&st->a[0],&st->a[1],&st->r);
printf("%d %d %.2f\n",st->a[0],st->a[1],st->r);
free(st); // звільнення пам'яті
//delete st; // звільнення пам'яті

```

Об'єднання – це сукупність поіменованих об'єктів різних типів, в яких однакова початкова адреса, тобто всі елементи об'єднання мають нульовий зсув. Тип об'єднання (*шаблон*) визначає, скільки елементів і якого типу входять в нього. Визначення типу об'єднання починається зі слова **union**, за яким розміщуються описи елементів, поміщені у фігурні дужки. Розмір пам'яті, що виділена під об'єднання, визначається розміром елемента з максимальною довжиною. Визначення шаблону об'єднання і самого об'єднання, а також доступ до елементів об'єднання виконується аналогічно структурам. Наприклад:

```

union UN{int a[2];char c[4];} un1,*ptrun=&un1;
un1.a[0]=25; // звернення до елемента об'єднання
ptrun->c[3]='e'; // звернення до un1.c[3] через покажчик

```

При визначенні об'єднання можна ініціалізувати тільки перший елемент. Наприклад:

```

UN un2={1, 12}; // ініціалізується перший елемент (масив a)

```

Елементами об'єднань можуть бути структури. Наприклад :

```

struct WORD{unsigned short ax; };
struct BYTE{unsigned char al; unsigned char ah; };
union REGS{WORD x; BYTE h;}rg;
rg.x.ax=0x4300; //запис в ax
printf("\n ah=%#x al=%#x\n", rg.h.ah rg.h.al);

```

Об'єднання можуть бути елементами структур. Наприклад:

```

union UN1{short b;
    char c[2];}; //виділиться пам'ять 2 байти
struct STR{char st;
    int a[3];
    UN1 my_u;}str1;
str1.my_u.b=500;
str1.st='z';

```

Бітові поля структур та об'єднань. Це цілі значення (типу `int`, `char`, `long`, `short`), що займають у пам'яті фіксоване число бітів (від 1 до 16 біт), призначені для доступу до окремих біт даних, а також для раціональнішого використання пам'яті.

Бітові поля можуть бути тільки елементами структур і об'єднань. Їх використовують для доступу до окремих бітів даних.

Розмір бітових полів задається через двокрапку після імені елемента структури або об'єднання

Визначення структури з бітовими полями має формат:

```
struct ім'я_типу_структури{тип_1 ім'я_1: ширина_1;
                           тип_2 ім'я_2: ширина_2;
                           . . . . . ;};
```

ім'я_типу_структури ім'я_структури;
де `тип_1`, `тип_2` - один з базових типів : `int`, `char`, `long`, `short`, `unsigned int`, `unsigned char`, `unsigned long`, `unsigned short`;
`ширина` - число від 1 до 16.

Наприклад:

```
struct SS{int a:5;    // для елемента a виділиться 5 бітів
          int b:3;}; // для елемента b виділиться 3 біта
```

`SS stb,*pb=&stb;` //визначення структури, покажчика на структуру

Для доступу до бітових полів використовуються уточнені імена.

Наприклад:

```
stb.a=31; stb.b=3; (*pb).a=1; pb->b=3;
```

Членами об'єднання можуть бути структури з бітовими полями.

Наприклад, об'єднання `un3` з змінною `ch` і структурою `str1`:

```
union UB{char ch; struct V{int a:5; //для a відвели 5 бітів
                          int b:3; //під b відвели 3 біта
                          }str1;
        } un3; //виділиться 8 біт пам'яті (1 байт)
un3.str1.a=4; un3.str1.b=2;
```

Для об'єднання `un3` виділиться 1 байт (8 біт) пам'яті.

Приклад об'єднання з бітовими полями:

```
union {char ch; unsigned a :1; unsigned b :2;}y;
y.ch=43; // 0010 1011 в двійковому коді або 2b в 16CC
printf("y.ch=%#x y.a=%#x y.b=%#x\n", y.ch, y.a, y.b);
```

На екран виведеться: `y.ch=0x2b y.a=0x1 y.b=0x3`

ЛАБОРАТОРНА РОБОТА

Структури та об'єднання

Мета роботи – вивчити способи визначення та ініціалізації структур, покажчиків на структуру і масивів структур; вивчити прийоми використання об'єднань, бітових полів структур і об'єднань.

Програма 1. В програмі визначаються проста структура, масив структур та покажчик на структуру. Елементом всіх структур привласнюються певні значення.

```
#include <stdio.h>
#include <stdlib.h>
#include<iostream>
using namespace std;
#pragma pack(push,1)
struct REC{char *name;    // визначення структурного типу
           int age;
           };

int main()
{REC rec1; // визначення структури rec1 типу REC
  rec1.name="Petrov";
  rec1.age=19;
  cout<< rec1.name<<" " <<rec1.age<<endl;
//Визначення масиву з 2-х структур типу REC і ініціалізація першої з них
  REC str[2]={"Sidorov",19 };
  str[1].name="Semenov";
  str[1].age=20;
  for(int i=0;i<2;i++)
    cout<<str[i].name<<" " <<str[i].age<<endl;
  REC *ptrr;    // визначення покажчика на структуру типу REC
  ptrr=new REC; // запит пам'яті для структури типу REC
  if (ptrr==NULL) {puts ("Error") ;
                  return -1;}
  ptrr->name="Ivanov";
  ptrr->age=18;
  cout<<(*ptrr).name<<" " <<ptrr->age<<endl;
  delete ptrr; // звільнення пам'яті
#pragma pack(pop)
  system("pause");
  return 0; }
```

Завдання до програми 1. Створіть заданий структурний тип, за допомогою якого визначить просту структуру, масив з 3-х структур та покажчик на структуру, під яку запросить динамічну пам'ять. Перші дві структури з масиву структур проініціалізуйте, елементам усіх інших структур привласніть будь-які значення. Всі значення структур виведіть їх на екран. Визначить розмір пам'яті, який займають ці структури.

1. Створіть структурний тип, членами якого є символ та масив з 3-х чисел типу **float**.
2. Створіть структурний тип, членами якого є масив з 2-х чисел типу **int** та покажчик на рядок.
3. Створіть структурний тип, членами якого є число типу **int** та масив з 3-х чисел типу **double**.
4. Створіть структурний тип, членами якого є число типу **float** та масив з 4-х чисел типу **int**.
5. Створіть структурний тип, членами якого є число типу **int** та масив з 3-х символів.

Програма 2. В програмі визначається структурний тип, елементом якого є раніш визначений структурний тип, тобто визначається вкладена структура. Далі визначається масив з сьома таких структур. Перші три структури з цього масиву ініціалізуються явно, а іншим елементам привласнюються певні значення.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{struct CHARACT{int rg;    // розрядність
                char *mem; // об'єм доступної пам'яті
            };
    struct INTEL {int year; // рік створення
                  char *mean; // позначення
                  CHARACT ch; // вкладена структура
            };
    INTEL in[7] ={{1974,"8080",8,"64K"},
                  {1978,"8086",16,"1M"},
                  {1983,"80186",16,"1M"}
    };
}
```

```

in[3].year=1983;
in[3].mean="80286";
in[3].ch.rg=16;
in[3].ch.mem="16M";
in[4].year=1987;
in[4].mean="i386";
in[4].ch.rg=32;
in[4].ch.mem="4G";
in[5].year=1990;
in[5].mean="i486";
in[5].ch.rg=32;
in[5].ch.mem="4G";
in[6].year=1993;
in[6].mean="Pentium";
in[6].ch.rg=64;
in[6].ch.mem="4G";
puts("History Of 80x86");
for(int i=0;i<7;i++)
{cout<<in[i].year<<" "<<in[i].mean<<" ";
  cout<<in[i].ch.rg<<" "<<in[i].ch.mem<<endl;
}
system("pause"); return 0;}

```

Завдання до програми 2 Визначить масив з 4-х структур заданого типу. Елементи перших двох структур проініціалізуйте, елементам інших структур привласніть довільні значення. Значення всіх елементів масиву структур виведіть на екран. Визначить розмір пам'яті, який займають весь масив структур та одна структура з масиву.

1. Структурний тип містить дані про книгу (автор, назва книги, рік видання) та вкладену структуру, що вказує місце збереження книги (стелаж, ряд і місце на полиці).
2. Структурний тип містить дані про місто (назва, країна, область) та вкладену структуру з даними про кількість мешканців міста, кількість середніх шкіл та кількість вузів.
3. Структурний тип містить дані про студента (прізвище, ім'я, по батькові) та вкладену структуру з назвою факультету, номером курсу, назвою групи.
4. Структурний тип містить дані про вищі учебні заклади: місто знаходження, назва та вкладену структуру з даними про кількість факультетів, спеціальностей, кількість бюджетних місць для вступу.

5. Структурний тип містить дані про працівника: прізвище, рік народження та вкладену структуру з даними про посаду, оклад, стаж.

Програма 3. В програмі демонструється використання об'єднання **union** для дослідження внутрішнього зображення дійсних чисел. Число типу **float** виводиться у вигляді цілого числа типу **int** в 10-ій та 16-ій системі числення .

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
//визначення типу UN об'єднання
union UN{float F;
        unsigned int L;};
int main()
{UN un1; // визначення об'єднання un1 типу UN
  puts("Enter real number");// введіть дійсне число
  scanf("%f",&un1.F);
  puts("Real number ");
  printf("F=%.3f\n", un1.F);
  puts("Real number in 10th, 8th, 16th system");
  printf("L=%lu=%#lo %#lx \n", un1.L, un1.L, un1.L);
  system("pause");
  return 0; }
```

Завдання до програми 3. Визначте об'єднання заданого типу. Визначить розмір пам'яті, який займає це об'єднання.

1. Визначте об'єднання, елементами якого є масиви з 3-х чисел типу **short** та з 6-и символів. Значення елементів об'єднання введіть з клавіатури у вигляді 16-ричних чисел типу **short**, а виведіть на екран спочатку у вигляді чисел типу **short**, потім у вигляді символів.
2. Визначте об'єднання, елементами якого є масив з 4-х символів і ціле число типу **long**. Значення елементів об'єднання введіть з клавіатури у вигляді символів, а виведіть на екран спочатку у вигляді символів, потім у вигляді 16-ричного і 10-тичного числа типу **long**.
3. Визначте об'єднання, елементами якого є масиви з 8-х символів та з 4-х чисел типу **short**. Значення елементів об'єднання введіть з клавіатури у вигляді символів, а виведіть на екран спочатку у вигляді символів, потім у вигляді 16-ричних та 10-тичних чисел типу **short**.

4. Визначте об'єднання, елементами якого є масиви з 8-х символів та 2-х чисел типу **long**, та проініціалізуйте його. Значення елементів об'єднання виведіть на екран спочатку у вигляді символів, потім у вигляді 16-ричних та 10-тичних чисел типу **long**.
5. Визначте об'єднання, елементами якого є масиви з 3-х чисел типу **int** та з 12-х символів, та проініціалізуйте його 16-ричними числами. Значення елементів об'єднання виведіть на екран спочатку у вигляді 16-ричних та 10-тичних чисел типу **int**, потім у вигляді символів.

Програма 4. В програмі показано, як за допомогою бітових полів структури можна закодувати дані в байті. Зверніть увагу, за допомогою функції **scanf()** не можна занести дані в бітове поле, тому використовується допоміжна змінна **flag**.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
#define M 3
#pragma pack(push,1)
struct CARD{char name[20];           // прізвище
            unsigned char sex:1;      //стать
            unsigned char family:1;  // сімейний стан
            unsigned char children:3; // кількість дітей
            unsigned char experience:3; //стаж роботи
            };
CARD card[M];
int main()
{int flag,i;
  for (i=0;i<M;i++)
    {puts("Enter details");
      puts("surname:"); // прізвище
      gets(card[i].name);
      puts("sex: woman -0, man -1");
      scanf("%d",&flag);    fflush(stdin);
      if(flag) card[i].sex=1;
      else card[i].sex=0;
      puts("married yes - 1, no - 0"); // женат(замужем)
      scanf("%d",&flag);    fflush(stdin);
      if(flag) card[i].family=1;
      else card[i].family=0;
```

```

    puts("number of children:");
    scanf("%d",&flag);    fflush(stdin);
    card[i].children=flag;
    puts("experience:"); // стаж роботи
    scanf("%d",&flag);    fflush(stdin);
    card[i].experience=flag;
}
puts("\n    The input data:");
for(int i=0; i<M;i++)
{printf("%d. %-15s ",i+1, card[i].name);
  if(card[i].sex==0)      printf("woman ");
  else printf("man ");
  printf("children - %d\n", card[i].children);
  printf("experience - %d\n\n",card[i].experience);
}
#pragma pack(pop)
system("pause"); return 0;}

```

Завдання до програми 4. Визначте масив з 3-х структур заданого типу. Елементи перших двох структур з масиву проініціалізуйте, елементам третьої структури привласніть довільні значення. Значення всіх елементів масиву структур виведіть на екран. Визначить розмір пам'яті, який займає масив структур та одна структура з масиву.

1. Структурний тип містить дані про працівника (прізвище, ім'я, по батькові) та бітові поля про стать та кількість повних років (до 100)
2. Структурний тип містить дані про студента (прізвище, рік народження) та бітові поля з даними бюджетна чи контрактна форма навчання, номер групи (до 4), номер за списком (до 30).
3. Структурний тип містить дані про книгу (прізвище автора, назва) та бітові поля з даними про місце знаходження книги: номер стелажу (до 4), номер полиці (до 100), номер місця на полиці (до 100).
4. Структурний тип містить дані про працівника: прізвище, рік народження та бітові поля з даними про стать, освіту (неповна середня, середня, середня спеціальна, вища), стаж (до 30).
5. Структурний тип містить дані про викладача (прізвище, ініціали) та бітові поля з даними про номер кафедри (до 16), посаду (завідувач кафедри, доцент, старший викладач, асистент), наявність ученого ступеня (магістр, ктн, дтн, немає)

Контрольні питання до лабораторної роботи

1. Дайте визначення структури та структурному типу (шаблону).
2. Визначте та проініціалізуйте структуру **r1** типу **REC** та масив з двох структур **r2** типу **REC**.

```
struct REC {short a[2];
            float b;};
```

Схематично покажіть розподіл пам'яті цих структур.

3. Створіть динамічну структуру типу **REC** за допомогою функції **malloc()**. Перевірте, чи створилася динамічна структура. Привласніть першому елементу значення **99**. Звільніть пам'ять.

4. Створіть масив з 3-х динамічних структур типу **REC** за допомогою операції **new**. Перевірте, чи створився масив динамічних структур. Привласніть останнім елементам значення **0.5**. Звільніть пам'ять.

5. Дайте визначення об'єднанню та типу об'єднання.

6. Визначить об'єднання **un1** та **un2** типу **UN**. Проініціалізуйте об'єднання **un1**, а в об'єднанні **un2** другому елементу привласніть значення **51000**.

```
union UN{int a[2]; float d; char c;};
```

Схематично покажіть розподіл пам'яті об'єднання **un1**.

7. Для чого можна використовувати бітові поля? Як задається розмір бітового поля при описі його в структурі?

8. Схематично покажіть розподіл пам'яті для структури **rec**:

```
struct STR {short n;
            unsigned char a:5;
            unsigned char b:3;
            char ch;}rec;
```

9. Схематично покажіть розподіл пам'яті для об'єднання **un3** та привласніть кожному елементу об'єднання максимальне значення:

```
struct STB{ unsigned char b1:2;
            unsigned char b2:3;
            unsigned char b3:5;
            unsigned char b4:6;
            };
union UN2{short Lg;
            STB bit;
            }un3;
```

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Подбельский В.В. Язык Си++: Учеб. пособие. – М.: Финансы и статистика, 2007. – 560 с.
2. Подбельский В.В. Практикум по программированию на языке Си: Учебн. пособие. – М.: Финансы и статистика, 2004. – 576 с.
3. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учебн. пособие. – 2-е изд.,– М.: Финансы и статистика, 2009. – 600с.
4. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб: Питер, 2009.– 464 с.
5. Касаткин А.И., Вальвачев А.Н. Профессиональное программирование на языке Си: От Turbo к C Borland C++.-Мн.: Выш. Шк., 1992. – 240 с.
6. Березин Б.И., Березин С.Б. Начальный курс С и С++. – М.: ДИАЛОГ-МИФИ, 2007. – 288 с.
7. Фридман А.Л. Основы объектно-ориентированного программирования на языке Си++. –М: Горячая линия. – Телеком. Радио и связь, 1999. –208 с.
8. Уильям Топп, Уильям Форд. Структуры данных в С++. / Пер. с англ.. – М.: ЗАО “Издательство БИНОМ”, 1999. – 816с.: ил.
9. Крячков А.В., Сухинина И.В., Томшин В.К. Программирование на С и С++. Практикум: Учебное пособие для вузов/ А.В.Крячков, И.В.Сухинина, В.К.Томшин.- 2-е изд.,исправ.-М.: Горячая линия.- Телеком, 2000.-344с.
10. Крупник А. Изучаем Си.- СПб.: Питер.-2001. – 256с.
11. С/С++. Структурное программирование. Практикум/ Т.А. Павловская, Ю.А.Щупак. – СПб.: Питер, 2002. – 240с.
12. Культин Н.Б. С /С++ в задачах и примерах.- СПб.: БХВ: Петербург, 2001. – 288с.
13. Х.М. Дейтел, П.Дж. Дейтел. Как программировать на С++. – 5-е изд. / Пер. с англ. – М.: ООО "Бином-Пресс", 2008.– 1456 с.
14. Шилдт, Герберт. Полный справочник по С++. – 4-е изд. / Пер. с англ. – М.: Вильямс, 2008. – 800 с.