Міністерство освіти і науки України

Національний університет “Львівська Політехніка”

Лабораторна робота №18

Виконав:
Студент групи АП-11
Білий Анатолій Іванович

Прийняв:

Чайковський І.Б.

Львів – 2024

“Структури та об’єднання даних”

Мета роботи: ознайомитися з поняттями структури та об’єднання даних , навчитися їх використовувати у процесі програмування.

Теоретичні відомості

Структура – це сукупність змінних, об'єднаних під одним ім'ям. За допомогою структур зручно розміщувати в суміжних полях пов'язані між собою елементи інформації.

Перед будь-яким використанням структур треба оголосити структурний тип. Оголошення структурного типу має такий вигляд:

```
struct ім'я _ структурного_типу {
   тип_поля ім'я_поля ;
   . . .
  тип_поля ім'я_поля ;
} ;
```

Елементами структури вважаються змінні, декларовані в списку, що обмежується фігурними дужками.

Оголошення структури створює шаблон, який можна використовувати для створення її об'єктів (тобто примірників цієї структури). Змінні, з яких складається структура, називаються членами (члени структури ще називаються елементами або полями.)

Як правило, члени структури пов'язані один з одним за змістом. Наприклад, елемент списку розсилки, що складається з імені та адреси логічно представити у вигляді структури.

У нижченаведеному фрагменті коду показано, як оголосити структуру, в якій визначені поля імені і адреси. Ключове слово struct повідомляє компілятору, що оголошується (ще кажуть, "декларується") структура.

```
struct addr
{
  char name[30];
  char street[40];
  char city[20];
  char state[3];
  unsigned long int zip;
};
```

Дане оголошення завершується крапкою з комою, оскільки оголошення структури - це оператор. Ім'я структури addr ідентифікує структуру даних.

Ім'я структури часто використовують як ярлик. На даний момент насправді не створено жодної змінної, визначена лише форма даних. Для оголошення справжньої змінної, відповідної даній структурі, слід написати:

struct addr addr_info;

У цьому рядку відбувається оголошення змінної addr_info типу addr.

Присвоювання структур. Інформація, яка знаходиться в одній структурі, може бути присвоєна іншій структурі того ж типу за допомогою єдиного оператора присвоювання. Немає необхідності присвоювати значення кожного члена окремо. Як виконується присвоювання структур, показує нижченаведена програма.

```
#include <stdio.h>
int main(void)
{
  struct {
    int a;
    int b;
  } x, y;
  x.a = 10;
  y = x;  /* присвоювання одної структури другій */
  printf("%d", y.a);
  return 0;
}
```

Структури в мові С не можна порівнювати, тобто спроба застосувати до них такі операції,як ==, => та подібні їм, компілятор вважає синтаксичною помилкою, як у фрагменті нижче:

```
struct tPoint A , B ;
. . .
if ( A = = B ) /* помилка ! */
        printf ( "співпадають" ) ;
else
        printf ( "різні" ) ;
```

Якщо виникає потреба в програмі порівнювати об'єкти структурного типу, програмісту потрібно в явному вигляді записати порівняння кожного поля:

```
if ((A . x = = B . x ) & & ( A . y = = B . y ))
```

```
        printf (" співпадають ") ;
else
        printf (" різні ") ;
```

Розглянемо нижче представлену програму для виведення певних видів даних про студента шляхом використання структури.

```
#include <stdio.h>
/* визначення структури*/
struct student
{
   char name[30];
   int kurs;
   int age;
};
int main()
{
   /* оголошення змінної stud1 типу struct student*/
   struct student stud1;
   printf("Vvedit imya:");
   gets(stud1.name);
   printf("Vvedit vik:");
   scanf("%d", &stud1.age);
   printf("Vvedit kyrs:");
   scanf("%d", &stud1.kurs);
   printf("Student %s\n", stud1.name);
   printf("Kyrs %d\n", stud1.kurs);
   printf("Vik %d\n", stud1.age);
}
```

gets - функція, що входить в стандартну бібліотеку мови С, оголошується в файлі stdio.h, яка зчитує рядок стандартного вводу і поміщає її в буфер, створений викликаючою функцією.

Структури можуть бути об'єднані в масиви структур. Оголошення масиву структур робиться аналогічно оголошенню масиву змінних. Наприклад, якщо потрібно зберігати інформацію про 10 студентів, то оголошення масиву буде наступним

struct student stud1 [10];

Тут stud1 - ім'я масиву структур, а stud1 [0] - це перша структура, stud1 [1] - друга ... stud1 [9] - десята структура.

Щоб отримати доступ до поля age п'ятої структури потрібно написати:

stud1[4].age

Щоб отримати доступ до поля kurs першої структури потрібно написати :

stud1[0].kurs

Нижче представлено програму, яка демонструє використання масиву структур.

```
#include <stdio.h>
/* визначення структури*/
struct student
{
   char name[30];
   int kurs;
   int age;
};
int main()
{
   /* оголошення масиву на 10 структур */
   struct student stud[10];
   int i, n;

   printf("Kilkict studentiv:");
   scanf("%d", &n);
   for(i=0;i<n;i++)
   {
      printf("Vvedit imya:");
      scanf("%s", stud[i].name);
      printf("Vvedit vik:");
      scanf("%d", &stud[i].age);
      printf("Vvedit kurs:");
      scanf("%d", &stud[i].kurs);
   }

   /* Виведення */
   for(i=0;i<n;i++)
   {
      printf("Student %s\n", stud[i].name);
      printf("Kurs %d\n", stud[i].kurs);
      printf("Vik %d\n", stud[i].age);
   }
}
```

Вказівники на структури використовуються , коли структура передається функції за допомогою виклику за посиланням.

У такого способу, як передача будь-яких (крім найпростіших) структур функцій, є один великий недолік: при виконанні виклику функції, щоб

помістити структуру в стек, необхідні істотні ресурси. Проте для простих структур з декількома членами ці ресурси є не такими вже й великими.

Але якщо в структурі є велика кількість членів або деякі члени самі є масивами, то при передачі структур функцій продуктивність може впасти до дуже низького рівня. Для вирішення цієї проблеми потрібно передавати не саму структуру, а вказівник на неї.

Коли функції передається вказівник на структуру, то в стек потрапляє тільки адреса структури. В результаті виклики функції виконуються дуже швидко. Щоб отримати адресу змінної-структури, необхідно перед її ім'ям помістити оператор &.

Об'єднання. Мова С підтримує також конструкцію даних, що називається об'єднанням.

Об'єднання дозволяє зберігати в одному об'єкті значення різних типів. При оголошенні об'єднання з ним асоціюється набір типів значень, які можуть в ньому зберігатися. В кожен момент часу об'єднання може зберігати значення тільки одного типу з набору. Контроль за тим, який тип зберігається, покладається на програміста. Оголошення об'єднання дуже схоже на оголошення структури, потрібно тільки замінити ключове слово "struct" на "union".

Синтаксис:

union [<тег>]{< список декларацій елементів>}<ім'я>[,...]

Пам'ять, що виділяється змінній типу об'єднання, визначається розміром найбільш довгого елементу. Усі елементи об'єднання розміщуються в тій самій області пам'яті з однією й тією ж адресою.

Приклади:

```
union sign { int svar;
   unsigned uvar;} number;
```

Це об'єднання дозволяє зберігати цілочисельне значення у знаковому та беззнаковому вигляді.

```
union test {
  uint8_t   c;
  uint32_t  i;
```

};

Даний код визначає шаблон, який має два члена: "c", який займає один байт, і "i", який займає чотири байти. Тепер ми можемо створити змінну цього шаблону об'єднання:

union test u1;

Отримати доступ до членів об'єднання "u1" можна, використовуючи оператор члена (.). Наприклад, наступний код привласнює значення 10 другого члену наведеного вище об'єднання і копіює значення "c" в змінну "m" (яка повинна мати тип uint8_t).

u1.i=10;

m=u1.c;

Беручи до уваги, що розмір структури, щонайменше, дорівнює сумі розмірів її членів, розмір об'єднання дорівнює розміру його найбільшою змінної. Простір пам'яті, виділений об'єднанню, буде спільно використовуватися всіма членами об'єднання. У наведеному вище прикладі розмір "u1" дорівнює розміру uint32_t, тобто чотири байти. Цей простір пам'яті розподіляється між "i" і "c". Отже, привласнення значення одному з цих двох членів, змінить значення іншого члена.

Хід роботи:

1. Ознайомитися з теоретичними відомостями.

2. Здійснити виконання прикладів, представлених у теоретичних відомостях, після чого представити скріни їх коду та результати виконання у звіті.

3. Написати програму для виведення нижчепредставленої інформації шляхом використання структури. Ім’я, вага, висота, вік – вводяться з клавіатури (вказати довільні дані).

Інформація про працівника Ім’я

- -

Вага

Висота

Вік

4. Оформити звіт.

Приклад 1

```
#include <stdio.h>
int main(void)
{
 struct {
 int a;
 int b;
 }
 x, y;
 x.a = 10;
 y = x; /* присвоювання одної структури другій */
 printf("%d", y.a);
 return 0;
}
```

10

Приклад 2

```
#include <stdio.h>
/* визначення структури*/
struct student
{
 char name[30];
 int kurs;
 int age;
};
int main()
{
 /* оголошення змінної stud1 типу struct student*/
 struct student stud1;
 printf("Vvedit imya:");
 gets(stud1.name);
 printf("Vvedit vik:");
 scanf("%d", &stud1.age);
 printf("Vvedit kyrs:");
 scanf("%d", &stud1.kurs);
 printf("Student %s\n", stud1.name);
 printf("Kyrs %d\n", stud1.kurs);
```

```
 printf("Vik %d\n", stud1.age);
}
```

```
// Vvedit imya:Анатолій
// Vvedit vik:18
// Vvedit kyrs:1
// Student Анатолій
// Kyrs 1
// Vik 18
```

Приклад 3

```
#include <stdio.h>
/* визначення структури*/
struct student
{
 char name[30];
 int kurs;
 int age;
};
int main()
{
 /* оголошення масиву на 10 структур */
 struct student stud[10];
 int i, n;
 printf("Kilkict studentiv:");
 scanf("%d", &n);
 for(i=0;i<n;i++)
 {
 printf("Vvedit imya:");
 scanf("%s", stud[i].name);
 printf("Vvedit vik:");
 scanf("%d", &stud[i].age);
 printf("Vvedit kurs:");
 scanf("%d", &stud[i].kurs);
 }
 /* Виведення */
 for(i=0;i<n;i++)
 {
 printf("Student %s\n", stud[i].name);
 printf("Kurs %d\n", stud[i].kurs);
```

```
        printf("Vik %d\n", stud[i].age);
        }
       }
```

```
        Kilkict studentiv:2
        Vvedit imya:Анатолій
        Vvedit vik:18
        Vvedit kurs:1
        Vvedit imya:Олег
        Vvedit vik:17
        Vvedit kurs:1
        Student Анатолій
        Kurs 1
        Vik 18
        Student Олег
        Kurs 1
        Vik 17
```

Завдання

```
#include <stdio.h>
// Визначення структури для зберігання інформації про працівника
struct Employee {
    char name[50];
    float weight;
    float height;
    int age;}
int main() {
    // Оголошення змінної типу структури Employee
    struct Employee emp;
    // Зчитування інформації з клавіатури
    printf("Ім'я працівника: ");
    scanf("%s", emp.name);

    printf("Вага працівника: ");
    scanf("%f", &emp.weight);

    printf("Висота працівника: ");
    scanf("%f", &emp.height);

    printf("Вік працівника: ");
```

```
    scanf("%d", &emp.age);

    // Виведення інформації про працівника
    printf("\nІнформація про працівника\n");
    printf("-------------------------------------\n");
    printf("Ім'я: %s\n", emp.name);
    printf("Вага: %.2f\n", emp.weight);
    printf("Висота: %.2f\n", emp.height);
    printf("Вік: %d\n", emp.age);
    return 0;}
```

```
Ім'я працівника: Анатолій
Вага працівника: 67
Висота працівника: 182
Вік працівника: 18

Інформація про працівника
-------------------------------------
Ім'я: Анатолій
Вага: 67.00
Висота: 182.00
Вік: 18
```

Контрольні питання

1. Дайте визначення поняття «структура».

   Структура - це складний тип даних в програмуванні, який дозволяє комбінувати різні типи даних під одним ім'ям. Вона може включати в себе змінні різних типів, об'єдуючи їх в логічний блок.

2. Яким чином здійснюється оголошення структури?

   Оголошення структури зазвичай відбувається шляхом визначення нового типу даних, який включає в себе різні поля або члени. У багатьох мовах програмування це здійснюється за допомогою ключового слова, такого як "struct" (у мові C або C++).

3. Охарактеризуйте синтаксис об’єднання даних.

   Синтаксис об'єднання даних (зазвичай відомий як "структури" або "об'єкти" в об'єктно-орієнтованих мовах програмування) включає в себе оголошення полів або членів, їх типи та доступ до них через оператори членства.

4. Які операції не можна застосовувати до структур?

До структур можна застосовувати багато операцій, але є деякі обмеження. Наприклад, неможливо виконати арифметичні операції безпосередньо над самою структурою (наприклад, додавання двох структур). Також, в деяких мовах програмування можуть бути обмеження на операції порівняння між структурами, які потрібно визначити явно.

Висновок: під час виконання даної лабораторної роботи я ознайомився з поняттями структури та об’єднання даних , навчився їх використовувати у процесі програмування.