СТРУКТУРИ.
Оголошення та ініціалізація структур.

Частина 2.

Структури

Структури

Найзагальніший метод отримання складених типів полягає в об'єднанні елементів довільних типів.

Причому самі ці елементи можуть бути у свою чергу складеними.

Структура – це сукупність логічно зв'язаних змінних, об'єднаних під одним іменем.

Структура в мові C/C++ – це тип даних, який складається з визначеної кількості елементів, що називаються *членами структури* (інколи – *полями структури*).

Структури

Структура –

це об'єднання кількох об'єктів, можливо, різного типу під одним ім'ям, яке є типом структури.

Як об’єкти можуть виступати:

змінні, масиви, покажчики та інші структури.

Оголошення та декларування

Структура – це похідний тип даних.

Вона створюється з об'єктів інших типів.

Загальний формат оголошення нової структури такий:

```
struct назва_структури {
        елементи_структури
};
```

Оголошення та декларування

Синтаксис визначення структури

struct [<назва>] {
<список полів структури>}
<декларація1>
[, <декларація2>...];

або

struct <назва> <декларація1> [,<декларація2>...];

Після закриття фігурної дужки } в оголошенні структури обов'язково ставиться крапка з комою.

Декларація **структури** задає ім'я типу структури і специфікує послідовність змінних величин, що називаються полями (елементами, членами) структури.

Ці поля структури можуть мати різні типи.

Оголошення та декларування

Ключове слово **struct** визначає структуру,

назва структури надає назву структурі,

елементи структури - це список оголошен змінних, що включені в структури.

Елементи однієї структури повинні мати унікальні імена.

Кожне визначення структури повинно закінчуватися “;”

Оголошення та декларування

Приклад оголошення структури

```
struct date
{
  int day;      // 4 байта
  char *month; // 4 байта
  int year;     // 4 байта
};
```

Поля структури розміщуються у пам'яті в порядку оголошення

У цьому прикладі структура date займає у пам'яті 12 байт. Крім того, покажчик *month під час ініціалізації буде початком текстового рядка з назвою місяця, розміщеного в пам'яті.

Масив & Структура

int A[n];

A[0]	A[1]	A[2]	…	A[i]			A[n-1]

Пам’ять = (n*4) байт

Якщо n=5, зайнято 5*4=20 байт

```
struct  ST{
        int A;
        double Q;
        char S[25];
        char M[255];
        float f;
};
```

int A	double Q	char S[25]	char M[255]	float f

Пам’ять= 1*4+1*8+25*1+255*1+1*4=296 байт

Оголошення та декларування

Структуру в С можна оголосити у два способи:

```
struct T_Struct {
   T1 s1;
   T2 s2;
   ...
   Tn sn;
};
```

або

```
typedef struct {
   T1 s1;
   T2 s2;
   ...
   Tn sn;
} T_Struct;
```

де ***T_Struct*** – ім'я структури, s_1, s_2,…, s_n – поля структури, відповідно типів T_1, T_2,…,

Оголошення та декларування

При оголошенні структури можна одночасно оголосити одну або кілька змінних.

1 спосіб	2 спосіб

struct **Address** {
char name [64];
char street [56];
char city [24];
int code;
};
 	
struct **Address** **a_info**, **b_info**, **c_info**;
	struct {
	char name [64];
	char street [56];
	char city [24];
	int code;
	} **a_info**, **b_info**, **c_info**;

оголошує структуру **Address** і оголошує змінні **a_info**, **b_info**, **c_info** даного типу.

Оголошення та декларування

Декларація структури починається з ключового слова **struct** і має дві форми подання.

У першій формі подання типи і імена елементів структури специфікуються в списку декларацій елементів <**список полів структури**>.

<**назва**> - це ідентифікатор, який іменує тип структури, визначений у списку декларацій елементів.

Кожна змінна **<декларації>** задає ім'я змінної типу структури.

Тип змінної в декларатері може відповідати вказівнику до структури, на масив структур або на функцію, яка повертає структуру.

Оголошення та декларування

Друга синтаксична форма використовує тег <**назва**> структури для посилання на тип структури.

У цій формі декларації відсутній список декларацій елементів, оскільки тип структури визначений в іншому місці.

Визначення типу структури має бути видимим для тегу, який використовується в декларації і визначення повинне передувати декларації через тег, якщо тег не використовується для декларації вказівника або структурного типу ***typedef***.

Оголошення та декларування

На відміну від масивів чи множин, усі елементи яких однотипні, структура може містити елементи різних типів.

Елементи структури називаються полями структури і можуть мати довільний тип, крім типу цієї ж структури, але можуть бути покажчиками на неї.

Якщо при описі структури відсутній тип структури, обов’язково повинен бути вказаний список змінних, покажчиків або масивів визначеної структури.

Приклад


```
struct Coord {
        char * namePoint;
        float X;
        float Y;
};
```

Визначення **struct Coord** складається з трьох елементів:

одного елементу символьного типу **char* (namePoint)** та двох елементів типу **float (X, Y)**.

Оголошення та декларування

Елементами структури можуть бути як змінні основних типів даних, так і агрегати, такі, як масиви, структури.

Елемент структури **не може** бути структурою такого ж типу, як структура, в якій він описаний.

Але такий елемент може бути **покажчиком** на тип структури, в яку він входить.

Структура, яка містить **покажчик**, що посилається на **структуру** такого ж типу, називається структурою, що посилається на саму себе.

Оголошення та декларування

Вищенаведене визначення структури не резервує місця в оперативній пам'яті.

Воно тільки створює **новий тип даних**, який можна використати для оголошення змінних.

Наприклад, в оголошенні

struct Coord myCoord, arrayCoord[52], *cPtr;

myCoord – змінна типу **struct Coord**;

arrayCoord – масив, що складається із 52 елементів типу **struct Coord**,

***cPtr** – покажчик на **struct Coord**.

Оголошення та декларування

Ім'я для структури не є обов'язковим.

Якщо воно відсутнє, то змінні для цієї структури можуть бути оголошенні тільки у визначенні структури.

Наступні **операції є допустимими** для структур:

→ присвоєння змінних-структур змінним того ж типу;

→ узяття адреси (&);

→ використання операції **sizeof** для визначення розміру структури.

Оголошення та декларування

Структури можуть бути ініційовані, як і масиви, із використанням списку ініціалізації.

Наприклад, оголошення

struct Coord myCoord = {“Точка 1”, 5.6, 10};

створює змінну **myCoord** типу **struct Coord** і присвоює елементу

namePoint значення **“Точка 1”**,

елементу **X – 5.6**,

елементу **Y– 10**.

Оголошення та декларування

Якщо значень у списку ініціалізації менше кількості елементів у структурі, решті присвоюється 0 (або **NULL**, якщо елемент – покажчик).

Змінним-структурам, якщо вони оголошенні як глобальні, присвоюється значення **0** або **NULL**.

Структури можна ініціалізувати і за допомогою оператора присвоєння.

При цьому можна або присвоїти змінній-структурі змінну того ж типу, або присвоїти значення окремим елементам структури.

Доступ до елементів структур

Для використання в програмі окремого елемента структури використовуються дві операції:

операція елемента структури (.), яка ще називається **операція-крапка**,

і **операція покажчика структури** (->), яка ще називається **операція-стрілка**.

Наступна програма демонструє ці дві операції.

Приклад

```
Point 1 has coordinates: 5.000000 ; 4.000000
Point 1  has coordinates: 5.000000 ; 4.000000
Point 1  has coordinates: 5.000000 ; 4.000000
```

```c
#include <stdio.h>
#include <conio.h>

 struct Coord {
   char * namePoint;
   float X;
   float Y;
 };

 int main() {
   struct Coord myCoord;
   struct Coord *cPtr;
   myCoord.namePoint="Point 1";
   myCoord.X=5;
   myCoord.Y=4;
   cPtr=&myCoord; // Присвоєння адреси структури змінній-покажчику
      // Використання елементів структури за допомогою операції-крапки
      // Використання елементів структури за допомогою операції-стрілки
      // Використання елементів структури за допомогою операції-точки та непрямого адресування
   printf("%s%s%f ; %f\n %s%s%f ; %f\n %s%s%f ; %f\n",
      myCoord.namePoint," has coordinates: ",myCoord.X,myCoord.Y,
      cPtr->namePoint, "  has coordinates: ", cPtr->X,cPtr->Y,
      (*cPtr).namePoint, "  has coordinates: ", (*cPtr).X,(*cPtr).Y);

   return 0;
}
```

```
#include <stdio.h>
#include <conio.h>

 struct Coord {
   char * namePoint;
   float X;
   float Y;
 };
```

```
int main() {
   struct Coord myCoord;
   struct Coord *cPtr;
   myCoord.namePoint="Point 1";
   myCoord.X=5;
   myCoord.Y=4;
   cPtr=&myCoord;
// Присвоєння адреси структури змінній-покажчику
```

```
// Використання елементів структури за допомогою операції-крапки
      // Використання елементів структури за допомогою операції-стрілки
      // Використання елементів структури за допомогою операції-точки та
непрямого адресування
   printf("%s%s%f ; %f\n %s%s%f ; %f\n %s%s%f ; %f\n",
      myCoord.namePoint," has coordinates: ",myCoord.X,myCoord.Y,
      cPtr->namePoint, "  has coordinates: ", cPtr->X,cPtr->Y,
      (*cPtr).namePoint, "  has coordinates: ", (*cPtr).X,(*cPtr).Y);
   return 0;
}
```

Стандартный вигляд оголошення структури


```
struct назва {
        тип ім'я змінної;
        тип ім'я змінної;
        тип ім'я змінної;
} структурні змінні;
```

назва - це ім'я **типу** структури, а не ім'я змінної.

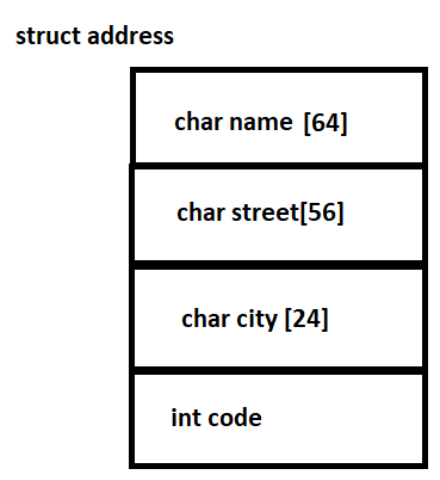
Структурні змінні - це список імен змінних.

Слід пам'ятати, що або назва, або структурні змінні можуть бути відсутніми, але не обидва.

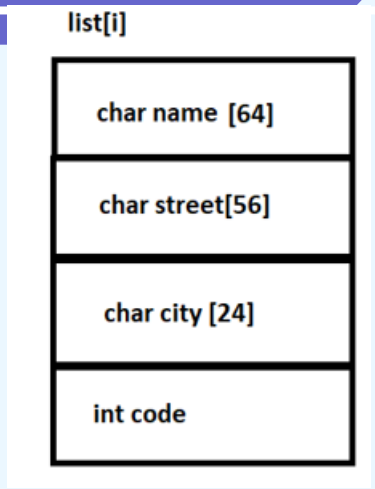
Приклад. Введення та виведення елементів структури

```
#include <stdio.h>
#include <stdlib.h>

struct Address {
      char name [64];
      char street [56];
      char city [24];
      int code;
};
```

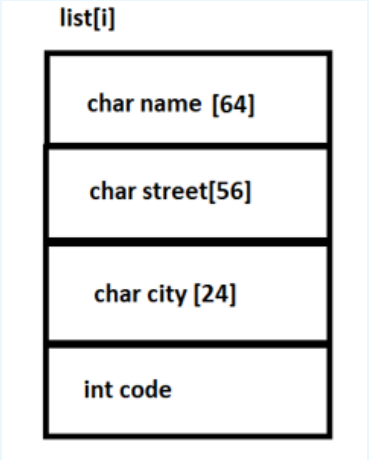


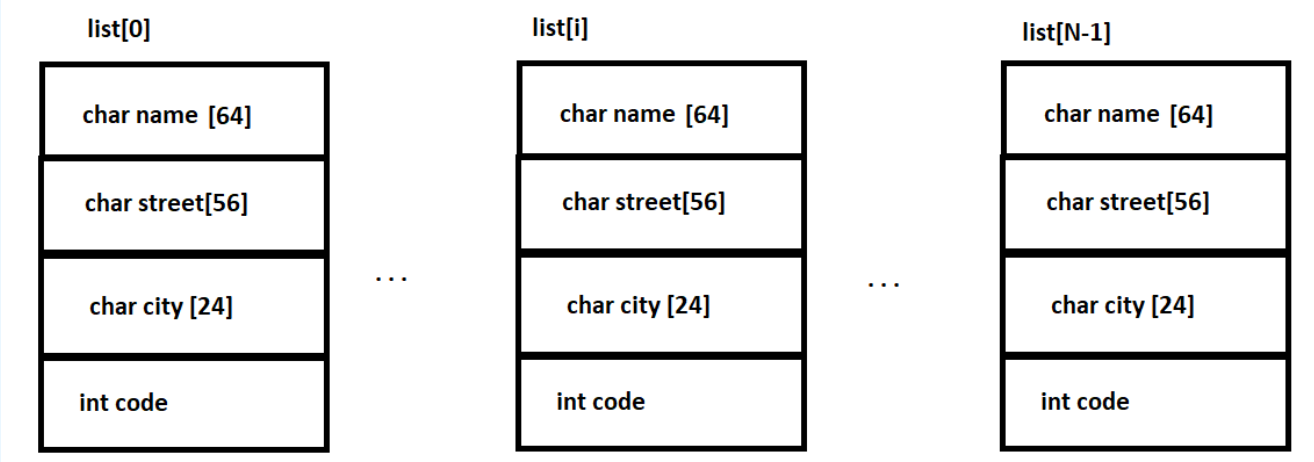

```
int main() {
    int N, i;
    struct address list[10];
     printf("Input quantity of elenents \n");
     scanf ("%d",&N);
     printf("\nInput  elenents of struct\n");
     for (i=0;i<N;i++)  {
          printf("\nInput  %d elenent of struct\n",i+1);
          scanf("%s",&list[i].name);
          scanf("%s",&list[i].street);
          scanf("%s",&list[i].city);
          scanf("%d",&list[i].code);
}
```



Приклад. Введення та виведення елементів структури

```
    printf("\nOutput  elenent of struct\n");
    for (i=0;i<N;i++) {
        printf("%s  ",list[i].name);
        printf("%s  ",list[i].street);
        printf("%s  ",list[i].city);
        printf("%d  ",list[i].code);
        printf("\n");
    }
    return 0;
}
```





Приклад. Введення та виведення елементів структури


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

struct Address {
  char name[64];
  char street[56];
  char city[24];
  int code;
};

int main() {
  int N, i;
  struct Address list[10];
  printf ("Input quantity of elenents \n");
  scanf ("%d", &N);
  printf ("\nInput  elenents of struct\n");
```

```
for (i = 0; i < N; i++) {
      printf ("\nInput  %d elenent of struct\n", i + 1);
      scanf ("%s", &list[i].name);
      scanf ("%s", &list[i].street);
      scanf ("%s", &list[i].city);
      scanf ("%d", &list[i].code);
}
```

Приклад. Введення та виведення елементів структури

```
    printf("\nOutput  elenent of struct\n");
    for (i=0;i<N;i++) {
       printf("%s  ",list[i].name);
       printf("%s  ",list[i].street);
       printf("%s  ",list[i].city);
       printf("%d  ",list[i].code);
       printf("\n");
    }
    return 0;
}
```

```
Input quantity of elenents
2

Input  elenents of struct

Input  1 elenent of struct
qwerty
werty
ytrewq
1234

Input  2 elenent of struct
asdfg
sdfghj
adgj
7654

Output  elenent of struct
qwerty  werty  ytrewq  1234
asdfg  sdfghj  adgj  7654
```

Структури можуть передаватися функціям шляхом передачі окремих елементів структури, передачі всієї структури або передачі покажчика на структуру.

Коли структури або їхні окремі елементи передаються функціям, вони передаються викликом за значенням.

Тому функція, яку викликають, не може змінювати елементи структури у функції, що викликає.

Масиви структур, як і інші структури, викликаються за посиланням.

Але, використовуючи структуру, в якій як елемент є масив, можна здійснити виклик масиву за значенням.

Викликаючи структуру за значенням, ми і масив викликаємо за значенням.

Дякую за увагу!