ЛЕКЦІЯ 12

1. ТИПИ, ЩО ВИЗНАЧАЮТЬСЯ КОРИСТУВАЧЕМ. СТРУКТУРИ
2. МАСИВИ СТРУКТУР
3. СТРУКТУРИ ТА ПОЛЯ СТРУКТУР ЯК ПАРАМЕТРИ ФУНКЦІЙ

ТИПИ, ЩО ВИЗНАЧАЮТЬСЯ КОРИСТУВАЧЕМ. СТРУКТУРИ

Розглянемо побудову типів, конкретна форма яких створюється користувачем. Відносяться до складених типів.
Перший з них - структура. Структура об’єднує декілька змінних, можливо різного типу.
Змінні, що об’єднані структурою, називають членами, елементами або, найчастіше, полями структури:

Приклад визначення структури:

```
struct student { char name[30]; int kurs; char group[5]; int stip;};
```

Оголошення структури є оператором, тому наприкінці має бути поставлена крапка з комою. При цьому поки жодної змінної не оголошено. Виділення пам'яті під змінну не відбулось. Під ім'ям *studen*t задано тільки вигляд структури, тобто її шаблон.
Визначено новий тип *student* !

Для того, щоб оголосити конкретні змінні типу *student*, в тексті програми необхідно написати

student stud1, stud2;

Тепер оголошено дві змінні – stud1 та stud2. Транслятор автоматично виділить під них місце в пам'яті комп'ютеру.

Під кожну зі змінних типу структура виділяється неперервна ділянка пам'яті.

Створення шаблону структури та оголошення змінних може відбуватись й одним оператором:

```
struct student {
char name[30]; char kurs; char group[5]; int stip; } stud1, stud2;
```

Тут одночасно задають структуру з ім’ям *student* та оголошують змінні *stud1* та *stud2*.

Доступ до конкретного поля структури здійснюється за допомогою операції точка (dot).
Наприклад:

strcpy(stud1.name, «Коваль А. А.");

Якщо потрібно вивести на екран зміст третього поля змінної stud2, необхідно написати

cout<<stud2.group;

Структури, як й змінні інших типів, можуть об’єднуватись у масиви структур.

Для того, щоб оголосити масив структур, необхідно спочатку задати шаблон структури (наприклад, вже в наявності шаблон student), а після цього оголосити масив:

student stud1kurs[200];

Цей оператор створить в пам’яті 200 змінних типу структура з шаблоном student та іменами *stud1kurs[0], stud1kurs[1*] … тощо.
Наприклад:
Для доступу до поля *kurs* 25-го елементу масиву: *stud1kurs[24].kurs*

Якщо необхідно отримати значення 5-го елементу поля *name* змінної *stud1*, необхідно написати *stud1.name[4]* .

Якщо оголошено дві змінні типу структура з однаковим шаблоном, можливо застосувати операцію присвоювання:

stud1=stud2;

При цьому відбудеться побітове копіювання кожного поля однієї змінної до відповідного поля іншої змінної - структури.
Заборонено використовувати операцію присвоювання змінних типу структури, шаблони яких оголошено під різними ідентифікаторами, нехай навіть зовсім ідентично.

Змінна типу структура може бути глобальною, локальною змінною та формальним параметром.
Можливо використовувати структуру або поле структури, як й будь-яку іншу змінну, як параметр функції. Наприклад:

Func1(first.a); func2(&second.b);

Наголосимо, що символ *&* розташовано перед ім’ям структури!

Можливо як формальний параметр використовувати всю структуру:

*main.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   struct st{int x; char y;};
5   void f(st param); /* прототип */
6   int main()
7   {
8   struct st arg;
9   arg.x=1;
10  arg.y='2';
11  f(arg);
12  return 0;
13  }
14  void f(st param)
15  {
16   cout<<param.x<<" "<<param.y;
17  }
```

Можливо також створити покажчик на структуру та передавати аргумент типу структури за посиланням. Оголосити покажчик на структуру можливо наступним чином :

*st *adr_pointer;*

adr_pointer – змінна типу покажчик на структуру *st*.

Якщо структура передається до функції за значенням, то всі її поля заносяться до стеку. Якщо структура проста й містить мало полів, то це є припустимим. Якщо ж структура своїм полем має масив, то стек може бути переповненим.
При передачі за посиланням до стеку буде занесено тільки адресу структури. При цьому копіювання структури не буде.
Також отримується можливість змінювання змісту полів структури.

main.cpp

```
1   #include <iostream>
2   using namespace std;
3
4   struct st{int x; char y;};
5   void f(st *param); /* прототип */
6   int main()
7   {
8   struct st arg;
9   arg.x=1;
10  arg.y='2';
11  f(&arg);
12  return 0;
13  }
14  void f(st *param)
15  {
16   cout<<(*param).x<<" "<<(*param).y;
17  }
```

Використання покажчиків на структуру зустрічається часто.
У зв'язку з цим, крім традиційного шляху отримання значення, розташованного за заданою адресою, використовуючи *(*a).x*, є ще один.

У мові C/C++ введено спеціальну операцію -> (стрілка, arrow).

Дана операція *стрілка* використовується замість операції точка, коли необхідно використати значення поля структур с застосуванням змінної – покажчика.

Замість (*a).x частіше використовують a -> x.

*main.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  struct st{int x; char y;};
5  void f(st *param); /* прототип */
6  int main()
7  {
8  struct st arg;
9  arg.x=1;
10 arg.y='2';
11 f(&arg);
12 return 0;
13 }
14 void f(st *param)
15 {
16   cout<<param->x<<" "<<param->y;
17 }
```

Нарешті, наголосимо, що як поля структури можливо використовувати масивы, структури та масиви структур.

*main.cpp

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4  struct addr {char city[30];int index; char street[30]; int housenumber;};
5  struct student{char name[30]; char group [5];int kurs; addr studaddr;};
6  int main()
7  {
8  student infiz [600];
9  int i;
10 for (i=1;i<140;i++)
11     infiz[i].kurs=1;
12
13     strcpy(infiz[1].studaddr.street,"Pushkinska");
14
15
16 return 0;
17 }
```

ДЯКУЮ ЗА УВАГУ!