ЛЕКЦІЯ 13

ТИПИ, ЩО ВИЗНАЧАЮТЬСЯ КОРИСТУВАЧЕМ. ПРОДОВЖЕННЯ

1. Побітові операції
2. Поля бітів (bit fields)
3. Об’єднання (unions)
4. Тип даних – перелічування (enumeration)
5. Перейменування типів –typedef

ПОБІТОВІ ОПЕРАЦІЇ

Порозрядні операції можна проводити з будь-якими цілими перемінними і константами. Не можна використовувати ці операції зі змінними типу float, double та long double.
Результатом побітової операції буде ціле значення.

Побітовими (порозрядними) операціями є:

& AND,
I OR,
^ XOR,
~ NOT,
<< зсув ліворуч,
>> зсув праворуч.

Деяка інформація щодо логічних операцій AND, OR, NOT та XOR вже обговорювалась. Основні знання щодо роботи з ними будуть надані в курсах дискретної математики та математичної логіки.
Аналогічні за назвою побітові операції за своєю суттю повністю аналогічні відповідним логічним операціям. Тільки в цьому випадку порівнюються не значення виразів, а значення кожної пари розрядів (бітів). Порозрядні операції дозволяють, зокрема, забезпечити доступ до кожного біта інформації.

ЛОГІЧНИЙ ОПЕРАТОР І (ДІЗ'ЮНКЦІЯ)

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

ЛОГІЧНИЙ ОПЕРАТОР АБО (КОН'ЮНКЦІЯ)

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

ЛОГІЧНИЙ ОПЕРАТОР ВИКЛЮЧНЕ АБО

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

ЛОГІЧНИЙ ОПЕРАТОР НІ

X	NOT X
0	1
1	0

Часто порозрядні операції знаходять застосування у драйверах пристроїв, програмах, пов'язаних із принтером, модемом та іншими пристроями. При виконанні порозрядної операції над двома змінними, наприклад, типу char, операція проводиться над кожною парою відповідних розрядів. Відмінність порозрядних операцій від логічних та операцій відношення полягає в тому, що логічні операції та операції відношення завжди в результаті дають 0 або 1. Для порозрядних операцій це не так.

Розглянемо 1 байт, який складається с 8 бітів (розрядів). Запис чисел у комп'ютері проходить у двійковій системі числення, коли всі числа представляються за степенями числа два.
Наприклад,
3_{10}= 0*2^7+ 0*2^6 + 0*2^5+ 0*2^4 + 0*2^3+ 0*2^2 + 1*2^1+ 1*2^0 =00000011

У побітових (bit-wise) операціях значення біта, що дорівнює 1, розглядається як логічна істина, а 0 як хибне значеня. Побітове AND (оператор &) бере два числа і логічно множить відповідні біти. Наприклад, якщо логічно помножити 3 на 8, то отримаємо 0.

3_{10}=00000011

8_{10}=00001000

Перший біт результату дорівнює логічному добутку першого біта числа a=3 та першого біта числа b=8. І так для кожного біта.

```
00000011
00001000
↓↓↓↓↓↓↓↓
00000000
```

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5  unsigned char a=255, b=90,c;
6  c=a & b;
7  cout<<c;
8  }
```

Доступ до окремого біта

На відміну від інших мов програмування, мова С забезпечує доступ до одного або кількох біт у байті або слові. Це має переваги. Якщо багато змінних набирають лише два значення, такі змінні іноді називають прапорами (наприклад, логічні)Для них можна використовувати 1 біт. Один із методів, що вбудованих у мову С, дозволяє мати доступ до біта, - це поля бітів (bit-fields). Насправді поля бітів - це спеціальний тип членів структури, у якому визначено, з скількох біт складається кожен елемент. Основна форма оголошення такої структури така:

```
struct імя_структури {
тип   ім’я1: довжина_в_бітах;
тип   ім’я2: довжина_в_бітах;
…
тип   ім’я N: довжина_в_бітах;
};
```

У цьому оголошенні тип може бути одним з наступних: int, unsigned або signed.
Ім'я1 може бути пропущеним, тоді відповідна кількість біт не використовується (пропускається).

Довжина структури завжди кратна восьми. Так, якщо вказати

struct onebit {unsigned one_bit: 1;} obj;

то для змінної obj буде виділено 8 біт, але використовуватиметься тільки перший. У структурі можуть бути змішані звичайні змінні та поля бітів.

Об'єднання (unions)

У мові С визначено ще один тип розміщення у пам'яті кількох змінних різного типу. Це об'єднання. Оголошується об'єднання так само, як і структура, наприклад:

```
union u {int i; char ch;};
```

Це оголошення не задає будь-яку змінну. Воно задає шаблон об'єднання. Можна оголосити змінну

```
union u alfa, beta;
```

Можна було оголосити змінні одночасно із завданням шаблону.
На відміну від структури для змінної типу union місця в пам'яті виділяється рівно стільки, скільки потрібно елементу об'єднання, що має найбільший розмір у байтах.
У наведеному вище прикладі під змінну alfa буде виділено 4 байти пам'яті. Насправді елемент і вимагає 4 байти, елемент ch - 1 байт.
Інша змінна буде розміщена в тому ж місці пам'яті.
Синтаксис використання елементів об'єднання такий самий, як і для структури:
u.ch = '5';
Для об'єднань також дозволено операцію ->, якщо ми звертаємося до об'єднання за допомогою покажчика.

Наведена нижче програма видає на екран двійковий код ASCII символу, що вводиться з клавіатури:

```
1   #include <iostream>
2   using namespace std;
3   /** Використання полів бітів та об'єднань */
4   struct byte {
5   int b1: 1;
6   int b2: 1;
7   int b3: 1;
8   int b4: 1;
9   int b5: 1;
10  int b6: 1;
11  int b7: 1;
12  int b8: 1;
13  };          /** Визначена структура - бітове поле */
14  union bits {
15  char ch;
16  struct byte b;
17  } u;        /** Визначено об'єднання*/
18  void decode (union bits b1);
19  int main ()
20  { do{
21  cin>>u.ch;
22  decode (u);
23  }while (u.ch != 'q');
24  return 0;
25  }
```

ПРОГРАМУВАННЯ

```
26    void decode (union bits bl)
27    {
28    if (bl.b.b8) cout<<"1";
29    else cout<<"0";
30    if (bl.b.b7) cout<<"1";
31    else cout<<"0";
32    if (bl.b.b6) cout<<"1";
33    else cout<<"0";
34    if (bl.b.b5) cout<<"1";
35    else cout<<"0";
36    if (bl.b.b4) cout<<"1";
37    else cout<<"0";
38    if (bl.b.b3) cout<<"1";
39    else cout<<"0";
40    if (bl.b.b2) cout<<"1";
41    else cout<<"0";
42    if (bl.b.bl) cout<<"1";
43    else cout<<"0";
44
45    cout<<"\n";
46    }
```

```
g
01100111
a
01100001
b
01100010
q
01110001
```

Тип перелічення (enumeration)

– це низка поіменованих цілих констант. Цей тип визначає всі припустимі значення, які можуть мати його змінні.

Основна форма оголошення типу наступна:

enum имя_типа (список_назв) список змінних;

Список змінних може бути пустим. Приклад визначення перерахованого типу та змінної даного типу:

```
enum seasons {win, spr, sum, aut};
enum seasons s;
```

Ключем до розуміння сутності типу перелічення є те, що кожне з імен win, spr, sum і aut є цілою величиною. Якщо ці величини не визначені інакше, то за замовчуванням вони відповідно дорівнюють нулю, одиниці, двом і трьом.

Під час оголошення типу можливо одному або декільком символам надати інші значення, наприклад:
enum value {one=1, two, three, ten=10, thousand=1000, next};
Якщо тепер надрукувати значення
one, 2, ten, thousand, next ;
то екрані з'являться числа 1 2 10 1000 1001, тобто кожен наступний символ збільшується на одиницю проти попереднього, якщо немає іншого присвоювання.

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5  enum value {one=1, two, three, ten=10, thousand=1000, next};
6  cout<<one<<" "<<2<<" "<<ten<<" "<<thousand<<" "<<next<<" "<<next+1;
7  }
```

```
1 2 10 1000 1001 1002
```

Зі змінними цього типу можливо проводити такі операції:
• присвоїти змінну типу enum іншій змінній того ж типу;
• провести порівняння з метою з'ясування рівності чи нерівності;
• арифметичні операції із константами типу enum (i = win - aut).

Не можна використовувати арифметичні операції та операції ++ і - для змінних типу enum.

Основна причина використання типу перелічення - це покращення читання програм.

Перейменування типів: typedef

Мова С дозволяє, крім того, дати нову назву вже існуючим типам даних. Для цього використовується ключове слово typedef. При цьому немає нового типу даних.
Наприклад:
typedef char SYMBOL;
typedef unsigned UNSIGN;

Досить часто використовується оператор typedef із застосуванням структур:
typedef struct st_tag {char name [30];int kurs;char group [3];int stip;} STUDENT;

Тепер для визначення змінної можна використовувати
struct st_tag avar;
а можна використовувати
STUDENT avar;

ДЯКУЮ ЗА УВАГУ!