Міністерство освіти і науки України

Національний університет “Львівська Політехніка”

Лабораторна робота №10-11

Виконав:
Студент групи АП-11
Білий Анатолій Іванович

Прийняв:

Чайковський І.Б.

Львів – 2024

“Умовні оператори та оператори переходу у мові С”

Мета роботи: навчитися використовувати умовні оператори та оператори переходу під час програмування на мові С.

Теоретичні відомості

Для розробки алгоритмів розв'язку задач (якщо задачі не зовсім примітивні) потрібна умовна конструкція, або розгалуження: в деякій точці алгоритму перевіряється певна умова, і в залежності від неї обирається один з двох можливих подальших шляхів.

Оператор if у загальному випадку має наступний вигляд:

```
if( логічний_вираз )
  оператор1;
else
  оператор2;
```

Слова if та else є ключовими словами мови С, тобто вони не можуть використовуватисядля інших цілей, крім умовного оператору (наприклад, не можна називати так змінні або функції) .

Виконання умовного оператору відбувається таким чином. Спочатку обчислюється значення виразу в дужках. Якщо це значення є логічною істиною (не дорівнює 0) , то виконується перший оператор, в протилежному випадку (логічна хиба, число 0) виконується другий оператор.

Наприклад, розглянемо задачу: знайти найбільше з двох заданих чисел. Нехай ці числа зберігаються у змінних a та b, а результат потрібно помістити у змінну m та вивести на дисплей. Тоді задачу розв'язує такий оператор:

```
if( a > b )
  m = a;
else
  m = b;
printf( "найбільше␣число␣%d\n", m );
```

Треба звернути увагу на те, що виклик функції printf в цьому прикладі не належить умовному оператору, а стоїть після нього.

Обидві гілки умовного оператору (як слово if, так і слово else) зв 'язують лише один наступний оператор. Якщо ж треба включити до умовної конструкції не один, а кілька операторів, то їх треба об'єднати у блок, взявши у фігурні дужки { . . . }.

Наприклад, нехай у програмі потрібно не лише присвоїти значення змінній m, але й надрукувати, яке саме з двох чисел виявилося більшим. Тоді програма набуває вигляду:

```
if( a > b ) {
 m = a;
 printf( "перше␣число␣більше\n" );
}
else {
 m = b;
 printf( "друге␣число␣більше\n" );
}
printf( "найбільше␣число␣%d\n", m );
```

Нерідко також виникає потреба виконати деякі дії, якщо умова виконується, але не робити нічого (тобто переходити далі до виконання наступного оператора) , якщо вона не виконується. Для цього слугує скорочена форма умовного оператору без гілки else:

```
if( логічний_вираз )
 оператор;
```

Наприклад, нехай треба взяти значення змінної x, знайти його модуль (абсолютне значення) та присвоїти його знову змінній x. Згідно означення,

$$|x| = \begin{cases} -x, & \text{якщо } x < 0; \\ x & \text{в іншому випадку.} \end{cases}$$

Тоді задачу розв’язує фрагмент:

```
if( x < 0 )
 x = -x;
```

Якщо значення змінної не від'ємне, то воно просто залишиться незмінним, що і треба.

Слід зробити тривіальне зауваження, що в складі гілки умовного оператору може стояти будь-який оператор, в тому числі ще один умовний оператор. Це можна використовувати для побудови складних розгалужених алгоритмів.

Наприклад, в програмі потрібно присвоїти змінній s значення 1, якщо змінна x має додатнє значення, або значення −1, якщо значення змінної x від'ємне, або значення x є 0, якщо значення змінної x є 0.

```
if( x == 0 )
 s = 0;
else {
 if( x > 0 )
  s = 1;
 else
 s = -1;
}
```

Умовний вираз, що входить в if, має мати скалярний результат. Це означає, що результатом має бути ціле число, символ або число з плаваючою точкою, але ним не може бути масив або структура.

У виразі-умови оператора if результат плаваючого типу використовується рідко, тому що це істотно сповільнює обчислювальний процес. Пояснюється це тим, що для виконання операцій над плаваючими операндами необхідно виконати більше команд процесора, ніж для виконання операцій над цілими числами або символами. У наступній програмі ілюструється використання оператора if. У ній запрограмована дуже проста гра "вгадай магічне число". Якщо гравець вгадав число, на екран виводиться повідомлення ** Вірно **. Програма генерує "магічне число" за допомогою стандартного генератора випадкових чисел rand (). Генератор повертає випадкове число в діапазоні між 0 і RAND_MAX (зазвичай це число не менше 32767). Функція rand () оголошена в <stdlib.h>.

```
/* Магічне число */
#include <stdio.h>
#include <stdlib.h>
main()
{
  int magic; /* магічне число */
  int guess; /* спроба гравця */
  magic = rand(); /* генерація магічного числа */
  printf("Vgaday magichne chuslo: ");
  scanf("%d", &guess);
  if(guess == magic) printf("** Virno **");
  return 0;
}
```

У наступній версії програми для гри в "магічне число" ілюструється

використання оператора else. У цій версії виводиться додаткове повідомлення в разі помилкової відповіді.

```
/* Магічне число 2*/
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int magic;
  int guess;
  magic = rand();
  printf("Vgaday magichne chuslo: ");
  scanf("%d", &guess);
  if(guess == magic) printf("** Virno **");
  else printf("Nevirno");
  return 0;
}
```

Оператор if є вкладеним, якщо він вкладений, тобто знаходиться всередині іншого оператора if або else. У практиці програмування вкладені умовні оператори використовуються досить часто. У вкладеному умовному операторі фраза else завжди асоціюється з найближчим if в тому ж блоці, якщо цей if не асоційований з іншого фразою else. Наприклад:

```
if(i)
{
  if(j) statement 1;
  if(k) statement 2; /* цей if */
  else  statement 3; /* асоційований з цим else */
}
else statement 4; /* асоційований з if(i) */
```

Остання фраза else не асоціюється з if (j) тому, що вона знаходиться в іншому блоці. Ця фраза else асоційована з if (i). Внутрішня фраза else асоційована з if (k), тому що цей if - найближчий.

Стандарт С89 допускає 15 рівнів вкладеності умовних операторів, С99 - 127 рівнів. В даний час більшість компіляторів допускають значно більшу кількість рівнів вкладеності. Однак на практиці необхідність в глибині вкладеності, більшою, ніж кілька рівнів, виникає досить рідко, так як збільшення глибини укладення швидко заплутує програму і робить її нечитаною.

У наступному прикладі вкладений оператор if використовується в

модернізованій програмі для гри в магічне число. З його допомогою гравець отримує повідомлення про характер помилки:

```
/* Магічне число 3*/
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int magic;
  int guess;
  magic = rand();
  printf("Vgaday magichne chuslo: ");
  scanf("%d", &guess);

  if (guess == magic) {
    printf("** Virno **");
    printf("Magichne chuslo rivne %d\n", magic);
  }
  else {
    printf("** Nevirno, ");
    if(guess > magic) printf("zanabto veluke\n");
                    /* вкладений if */
    else printf("zanabto male\n");
  }
  return 0;
}
```

У програмах часто використовується конструкція, яку називають сходами if-else-if [1].

Загальна форма сходів має вигляд

if (вираз) оператор;
else
if (вираз) оператор;
else
if (вираз) оператор;
.
.
else оператор;

Працює ця конструкція наступним чином. Умовні вирази операторів if обчислюються зверху вниз. Після виконання деякої умови, тобто коли зустрінеться вираз, що приймає значення істина, виконується асоційований з цим виразом оператор, а частина, що залишилася, пропускається.

Якщо всі умови помилкові, то виконується оператор в останній фразі else, а якщо остання фраза else відсутня, то в цьому випадку не виконується жоден оператор. Недолік попереднього запису сходинок полягає в тому, що з ростом глибини укладення збільшується кількість відступів в рядку. Це стає незручним з технічної точки зору.

Використовуючи сходи if-else-if, програму для гри в "магічне число" можна записати так:

```
/* Магічне число, програма 4 */
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
 int magic;
 int guess;
 magic = rand();
 printf("Vgaday magiche chyslo : ");
 scanf("%d", &guess);

 if(guess == magic) {
  printf("** Virno ** ");
  printf("Magichne chyslo rivne %d\n", magic);
 }
 else if(guess > magic)
  printf("Nevirno, zanabto veluke");
 else printf("Nevirno, zanabto male");
 return 0;
}
```

Оператор "?", альтернативний умовному. Оператор ? можна використовувати замість оператора if-else, записаного в формі:

if (умова) змінна = вираз;

else змінна = вираз;

Оператор ? є тернарним, тому що він має три операнда. Його загальна форма наступна:

Вираз1 ? Вираз2: Вираз3;

Зверніть увагу на використання і розташування двокрапки. Результат операції? визначається наступним чином. Спочатку обчислюється Вираз1. Якщо воно має значення істина, обчислюється Вираз2 і його значення стає результатом

операції ?. Якщо Вираз1 має значення брехня, обчислюється вираз3 і його значення стає результатом операції ?. наприклад:

```
x = 10;
y = x>9 ? 100 : 200;
```

У цьому прикладі змінній у присвоюється значення 100. Якби х було менше 9, то змінна у отримала б значення 200. Те ж саме можна записати, використовуючи оператор if-else:

```
x = 10;
if (x> 9) y = 100;
else y = 200;
```

Оператор вибору – switch. Оператор вибору switch (часто його називають перемикачем) призначений для вибору гілки обчислювального процесу, виходячи з значення керуючого виразу (при цьому значення керуючого виразу порівнюється зі значеннями в списку цілих або символьних констант. Якщо буде знайдено збіг, то виконається асоційований із співпавшою константою оператор.) Загальна форма оператора switch наступна:

```
switch  (вираз)  {
  case постійна1:
    послідовність операторів
    break;
  case постійна2:
    послідовність операторів
    break;
  case постійна3:
    послідовність операторів
    break;
  default:
    послідовність операторів;
}
```

Значення виразу оператора switch має бути таким, щоб його можна було висловити цілим числом. Це означає, що в керуючому виразі можна використовувати змінні цілого або символьного типу, але тільки не з плаваючою точкою.

Значення керуючого виразу по черзі порівнюється з постійними (константами) в операторах case. Якщо значення керуючого виразу співпаде з якоюсь із констант,

управління передається на відповідну мітку case і виконується послідовність операторів до оператора break.

Якщо оператор break відсутній, виконання послідовності операторів триває до тих пір, поки не зустрінеться break (в інший мітці) або не скінчиться тіло оператора switch (тобто блок, наступний за switch). Оператор default виконується в тому випадку, коли значення керуючого виразу не співпало з жодною постійною (константою). Оператор default також може бути відсутнім. В цьому випадку при відсутності збігів не виконується жоден оператор.

Оператори переходу

У мові С визначені чотири оператора переходу: return, goto, break і continue. Оператори return і goto можна використовувати в будь-якому місці всередині функції. Оператори break і continue можна використовувати в будь-якому з операторів циклу.

Для оператора goto завжди необхідна мітка. Мітка - це ідентифікатор з подальшою двокрапкою. Мітка повинна знаходиться в тій же функції, що і goto, перехід в іншу функцію неможливий. Загальна форма оператора goto наступна:

goto мітка;

мітка:

Мітка може знаходитися як до, так і після оператора goto. Наприклад, використовуючи оператор goto, можна виконати цикл від 1 до 100:

```
x = 1;
loop1:
  x++;
  if(x<=100) goto loop1;
```

Цей оператор може бути корисним, наприклад, якщо потрібно залишити глибоко вкладені цикли.

Оператор break застосовується в двох випадках. По-перше, в операторі switch з його допомогою переривається виконання послідовності case . У цьому випадку оператор break не передає управління за межі блоку. По-друге, оператор break використовується для негайного припинення виконання циклу без перевірки його умови, в цьому випадку оператор break передає управління оператору, наступному після оператора циклу.

Коли всередині циклу зустрічається оператор break, виконання циклу безумовно (тобто без перевірки будь-яких умов) зупиняється і керування передається оператору, наступному за ним. Наприклад, програма виводить на екран числа від 0 до 10.

```
#include <stdio.h>
int main(void)
{
  int t;

  for(t=0; t<100; t++) {
    printf("%d ", t);
    if(t==10) break;
  }
  return 0;
}
```

Після цього виконання циклу припиняється оператором break, умова t <100 при цьому ігнорується.

Оператор break часто використовується в циклах, в яких деяка подія має викликати негайне припинення виконання циклу. Якщо оператор break присутній всередині оператора switch, який вкладений в будь-які цикли, то break відноситься тільки до switch, вихід з циклу не відбувається.

Важлива особливість циклів полягає в тому, що при деяких (а можливо і всіх) вхідних даних цикл ніколи не закінчиться. Розглянемо, наприклад, програму:

```
int x, s;
s = 0;
while( s < 100 ) {
 printf( "введіть␣число␣" );
 scanf( "%d", &x );
}
```

Цикл має продовжуватися доти, доки значення змінної s не перевищить 100. Перед початком циклу ця змінна має значення 0, отже програма увійде до циклу. Але в тілі циклу жоден оператор не змінює значення змінної s, отже воно так і залишається рівним 0, тому цикл буде виконуватися вічно.

Іноді зациклювання програми є не помилкою, а саме очікуваною та бажаною її поведінкою. Наприклад, програма автоматичного продажу авіаквитків працює за алгоритмом: отримати замовлення, зменшити на 1 кількість вільних місць на рейс, зняти гроші з банківського рахунку замовника, а потім знов повторити те ж саме.

Тобто програма повинна зазначену послідовність дій повторювати весь час в циклі. Такі цикли називають нескінченними.

Оператор циклу забезпечує виконання оператору або послідовності операторів, перевіряючи умову продовження або перед кожною ітерацією (цикл з передумовою), або після неї (з постумовою). Разом з тим, іноді буває зручно завершити цикл достроково, вийти з циклу на середині ітерації. Іншими словами, це означає розмістити в тілі циклу додаткову точку виходу із циклу.

Для цього слугує спеціальний оператор break. Зазвичай оператор break в тілі циклу застосовується у складі умовного оператора , тоді умова в операторі if грає роль умови виходу з середини циклу (як на наведеній блок-схемі) . Якщо під час виконання циклу десь всередині тіла циклу програма виконує оператор break, то цикл завершується, і управління передається далі, на наступний оператор

```
while( умова1 ) {
 оператори1;

 if( умова2 )
  break;
 оператори2;
}
```

Оператор виходу з середини циклу відкриває нові цікаві можливості. Наприклад, авторимови С безпосередньо заклали в неї цикли, в яких умова продовження перевіряється перед або після кожної ітерації, та не зробили окремого оператору циклу, в якому умова продовження перевіряється всередині ітерації. Попри відсутність в мові С спеціального оператору, таку структуру управління легко змоделювати власноруч як нескінченний цикл з додатковою точкою виходу з середини тіла (треба звернути увагу, що у звичайних операторах циклу умова є умовою продовження, а тут навпаки, умовою завершення).

Ще одна можливість циклів – оператор continue. Якщо він виконується всередині тіла циклу, то поточна ітерація циклу достроково закінчується, управління передається на точку перевірки умови продовження (перед тілом у циклі з

передумовою, після тіла у циклі з постумовою) . Розглянемо програму:

```
while( умова1 ) {
 оператори1;
 if( умова2 )
  continue;
 оператори2;
}
```

У ній під час виконання циклу, якщо виконується умова2, то оператори2 не виконуються, оператор continue одразу передає управління на перевірку умови1. Для циклу з постумовою аналогічно.

Слід зазначити, що оператори break та continue не є необхідними для мови програмування: будь-яку програму, де в циклах використовуються ці два оператори, можна переписати,замінивши їх більш чи менш витонченим використанням операторів if. Оператори переривання та продовження циклу запроваджені для зручності, для скороченого позначення тих структур програм, які без них виглядали б більш громіздкими.

Можна сказати, що оператор continue трохи схожий на break. Оператор break викликає переривання циклу, а continue - переривання поточної ітерації циклу і здійснює перехід до наступної ітерації.

У наступному прикладі програма підраховує кількість пробілів в рядку, введеним користувачем:

```
/* Підрахунок кількості пробілів */
#include <stdio.h>
int main(void)
{
  char s[80], *str;
  int space;
  printf("Vvedit riadok: ");
  gets(s);
  str = s;
  for(space=0; *str; str++) {
    if(*str != ' ') continue;
    space++;
  }
  printf("%d probiliv\n", space);

  return 0;
}
```

Кожен символ рядка порівнюється з пробілом. Якщо порівнюваний символ не є пропуском, оператор continue передає управління в кінець циклу for і виконується наступна ітерація. Якщо символ є пробілом, значення змінної space збільшується на 1.

Хід роботи:

1. Ознайомитися з теоретичними відомостями.

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

3. Виконати програму, яка ілюструє розгалуження умовними операторами та пояснити отримані результати:

```
#include <stdio.h>
#include <conio.h>
void main()
{int a=2,b=0,c=1;
printf("\n\n");
if(a > 0 && b < -3) c = b*b/a; printf("c=%d\n",c);     /*c=1*/
a = ++c/a + a%c;
b+= c*c;
if(a < b || a < 0) {c *= a; printf("c=%d\n",c);}       /*c=2*/
else if (c++ == 2) printf("c=%d\n",c);
if(b < a && a == 2) c = 2*a +1;
else { c = (b--) + a; a = 0; }
printf("c=%d\n",c);                                    /*c=5*/
a = b = 2;
if(c >= 3) if(a < 0 || a > c) c = 0;
          else { a = 1; c = 7; printf("c=%d\n",c); }     /*c=7*/
if(c > 0 && c < 10) {if(a > 0) printf("c=%d\n",c++);}  /*c=7*/
else c = 10;
if(c <= 5) if((a = b + 1) > 2) c %= 2;
printf("c=%d\n",c);                                    /*c=8*/
a = 3; b = -1;
if(b > 0) c = 1;
else if(b < -10) { c = -1;   printf("c=%d\n",c); }
    else if( b <= -3 ) c = 2;
        else c = b*b + 10; printf("c=%d\n",c);         /*c=11*/
getch();
}
```

4. Написати програму для здійснення базових арифметичних операцій (додавання, віднімання, множення, ділення) над двома числами, використовуючи

умовний оператор if. Врахувати, що на нуль ділити неможна. Значення чисел та знак операції вводяться з клавіатури.

4. Оформити звіт.

Приклад 1

```
      #include
<stdio.h>
#include
<stdlib.h> int
main(void)
{
 int magic;
 int guess;
 magic = rand();
 printf("Vgaday magichne chuslo: ");
 scanf("%d", &guess);

 if (guess == magic)
  { printf("** Virno
  **");
  printf("Magichne chuslo rivne %d\n", magic);
 }
 else {
  printf("** Nevirno, ");
  if(guess > magic) printf("zanabto veluke\n");
                /* вкладений if
  */ else printf("zanabto
  male\n");
 }
 return 0; }
```

```
Vgaday magichne chuslo: 1
** Virno **Magichne chuslo rivne 1
```

Приклад 3

```
      #include
      <stdio.h>
      #include
      <stdlib.h>
      #include <time.h> // Підключаємо бібліотеку для використання
      функції time() int main(void) {
         int magic;
         int guess;
         srand(time(NULL));
         magic = rand() % 10;
         printf("Вгадайте магічне число:
```

```
   "); scanf("%d", &guess);
   if (guess == magic) {
      printf("** Вірно
      **\n");
      printf("Магічне число рівне %d\n", magic);
   } else {
      printf("** Невірно,
      "); if(guess >
      magic)
         printf("занадто
      велике\n"); else
         printf("занадто мале\n");
   }
      return 0;}
```

Вгадайте магічне число: 1
** Невірно, занадто велике

Приклад 5

```
#include
<stdio.h> int
main(void)
{
 int t;
 for(t=0; t<100; t++)
 { printf("%d ", t);
 if(t==10) break;
 }
 return 0;
}
```

---

0 1 2 3 4 5 6 7 8 9 10

Приклад 6

```
/* Підрахунок кількості пробілів */
#include <stdio.h>
int main(void)
{
 char s[80],
 *str; int space;
 printf("Vvedit riadok:
 "); gets(s);
 str = s;
 for(space=0; *str;
 str++) { if(*str != ' ')
 continue; space++;
 }
 printf("%d probiliv\n",
 space); return 0;
```

```
}
```

```
Vvedit riadok: Hello
world 1 probiliv
```

Приклад 7

```
#include <stdio.h>
#include <conio.h> void main() {
   int a = 2, b = 0, c = 1;
   printf("\n\n");
   if (a > 0 && b < -3) c = b * b / a;
   printf("c=%d\n", c); // c=1
   a = ++c / a + a % c; //
   b = +c * c;
   if (a < b || a < 0) {
      printf("c=%d\n", c); // c=2
   } else if (c++ = = 2) printf("c=%d\n", c);
   if (b < a && a == 2) c = 2 * a + 1;
```

```
    else {
        c = (b--) + a; a = 0;
    }
    printf("c=%d\n", c); // c=5 a = b
    = 2;
    if (c >= 3) {
        if (a < 0 || a > c);
        else {
            a = 1;
            c = 7;
            printf("c=%d\n", c); // c=7 if (c >
            0 && c < 10) {
                if (a > 0)
                    printf("c=%d\n", c++);
            }
            else
                c = 10;
        }
    }
    if (c <= 5) {
        if ((a = b + 1) > 2);
    }
    printf("c=%d\n", c);

    a = 3;
    b = -3;

    if (b > 0) c = 1;
    else if (b < -10) { c = -1;
        printf("c=%d\n", c);
    }
    else if (b <= -3) c = 2;
    else
        c = b * b + 10; printf("c=%d\n",
    c);

    getch();
}
```

```
c=1
c=2
c=5
c=7
c=7
c=8
c=2
```

Приклад 8

```
#include <stdio.h>

int main() {
```

```
    float num1, num2; char
    operator;

    printf("Введіть перше число: ");
    scanf("%f", &num1);

    printf("Введіть друге число: ");
    scanf("%f", &num2);

    printf("Введіть операцію (+, -, *, /): ");
    scanf(" %c", &operator);

    if (operator == '+') {
        printf("%.2f + %.2f = %.2f\n", num1, num2, num1 + num2);
    } else if (operator == '-') {
        printf("%.2f - %.2f = %.2f\n", num1, num2, num1 - num2);
    } else if (operator == '*') {
        printf("%.2f * %.2f = %.2f\n", num1, num2, num1 * num2);
    } else if (operator == '/') { if
        (num2 != 0) {
            printf("%.2f / %.2f = %.2f\n", num1, num2, num1 / num2);
        } else {
            printf("Помилка: ділення на нуль\n");
        }
    } else {
        printf("Помилка: невідома операція\n");
    }
    return 0;
}
    ------------------------------------------------------------
#include <stdio.h>

int main() {
    float num1, num2; char
    operator;

    printf("Введіть перше число: ");
    scanf("%f", &num1);

    printf("Введіть друге число: ");
    scanf("%f", &num2);

    printf("Введіть операцію (+, -, *, /): ");
    scanf(" %c", &operator);

    if (operator == '+') {
        printf("%.2f + %.2f = %.2f\n", num1, num2, num1 + num2);
```

```
    } else if (operator == '-') {
        printf("%.2f - %.2f = %.2f\n", num1, num2, num1 - num2);
    } else if (operator == '*') {
        printf("%.2f * %.2f = %.2f\n", num1, num2, num1 * num2);
    } else if (operator == '/') { if
        (num2 != 0) {
            printf("%.2f / %.2f = %.2f\n", num1, num2, num1 / num2);
        } else {
            printf("Помилка: ділення на нуль\n");
        }
    } else {
        printf("Помилка: невідома операція\n");
    }
    return 0;
}
```

```
Введіть
перше
число: 4

Введіть
друге
число: 2

Введіть
операці
ю  (+,  -,
*, /):
*

4.00    *
2.00    =
8.00
```

Контрольні питання

1. Назвіть умовні оператори у мові С.

   Умовні оператори у мові С включають:

   - `if` : Використовується для виконання коду, якщо певна умова істинна.

   - `else`: Використовується разом з `if` для виконання альтернативного коду, якщо умова `if` не виконується.

   - `else if` : Дозволяє перевіряти додаткові умови, якщо попередня умова `if` не виконується.

   - `switch`: Використовується для вибору виконання коду залежно від значення виразу.

2. Назвіть оператори переходу у мові С.

Оператори переходу в мові С включають:

- `break`: Використовується для виходу з циклу або `switch`.

- `continue`: Використовується для переходу до наступної ітерації циклу.

- `goto`: Використовується для безумовного переходу до мітки в програмі

3. Охарактеризуйте умовний оператор if.

   Умовний оператор `if` в мові С дозволяє виконувати певний блок коду, якщо певна умова істинна. Синтаксис виглядає наступним чином:

   if (умова) {

   // Код, який виконується, якщо умова істинна

4. Охарактеризуйте призначення оператора break.

   Оператор `break` використовується для негайного виходу з циклу або

   `switch`. Якщо `break` використовується у циклі, виконання циклу припиняється, і виконання програми продовжується з наступного оператора після циклу. У `switch`, `break` призводить до виходу з `switch`, запобігаючи виконанню інших `case` або `default`.

5. Поясніть призначення функції rand.

   Функція `rand` в мові С використовується для генерації випадкових чисел. Вона повертає псевдовипадкове ціле число у заданому діапазоні. Щоб коректно використовувати функцію `rand`, потрібно попередньо ініціалізувати генератор псевдовипадкових чисел за допомогою функції `srand`.

   Висновок: під час виконання цієї лабораторної роботи я навчився використовувати умовні оператори та оператори переходу під час програмування на мові С.