

1. Написав програму котра ілюструє роботі всіх видів операторів

```

1 #include <stdio.h>
2
3 // Функція для перетворення десяткового числа в двійковий формат
4 void decimalToBinary(unsigned int num, char *binaryStr) {
5     for (int i = 0; i < 32; i++) {
6         binaryStr[31 - i] = (num & (1 << i)) ? '1' : '0';
7     }
8     binaryStr[32] = '\0'; // Завершення рядка
9 }
10
11 int main()
12 {
13     int str;
14     printf("Choose the type of operator (arithmetical(1),
15         logical(2), by bit(3)): ");
16     scanf("%d", &str);
17
18     switch (str) {
19         case 1: {
20             double num1, num2, result;
21             int arithmeticalOperation;
22             printf("Choose the type of operator: (+(1), -(2), /(3)
23                 ), *(4), %(5)): ");
24             scanf("%d", &arithmeticalOperation);
25             printf("Select two operands: ");
26             scanf("%lf", &num1);
27             scanf("%lf", &num2);
28         }
29     }
30 }

```

```

26
27 - switch (arithmeticalOperation) {
28     case 1: // '+' operation
29         result = num1 + num2;
30         printf("%.21f + %.21f = %.21f\n", num1, num2
31             , result);
32         break;
33     case 2: // '-' operation
34         result = num1 - num2;
35         printf("%.21f - %.21f = %.21f\n", num1, num2
36             , result);
37         break;
38     case 3: // '*' operation
39         result = num1 * num2;
40         printf("%.21f * %.21f = %.21f\n", num1, num2
41             , result);
42         break;
43     case 4: // '/' operation
44         if (num2 != 0) {
45             result = num1 / num2;
46             printf("%.21f / %.21f = %.21f\n", num1,
47                 num2, result);
48         } else {
49             printf("Error! Division by zero is not
50                 allowed.\n");
51         }
52         break;
53 }
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

```

48         default:
49             printf("Error! Invalid operator.\n");
50         }
51         break;
52     }
53     case 2: {
54         double num3, num4;
55         int logicalOperation;
56         printf("Choose the type of operator: (&&(1), ||(2),
57             !(3)): ");
58         scanf("%d", &logicalOperation);
59         printf("Select two operands: ");
60         scanf("%lf", &num3);
61         scanf("%lf", &num4);
62
63         switch (logicalOperation) {
64             case 1: // Logical AND
65                 if (num3 > 0 && num4 > 0) {
66                     printf("Both num3 and num4 are positive
67                         numbers.\n");
68                 }
69                 break;
70             case 2: // Logical OR
71                 if (num3 > 0 || num4 > 0) {
72                     printf("At least one of num3 or num4 is
73                         positive.\n");
74                 }
75             }
76         }
77     }
78 }
79
80 // End of Program
81
82 // Author:
83 // Date:
84
85 // Compiler:
86
87 // Version:
88
89 // Tested on:
90
91 // Tested by:
92
93 // Tested on:
94
95 // Tested by:
96
97 // Tested on:
98
99 // Tested by:
100
101 // Tested on:
102
103 // Tested by:
104
105 // Tested on:
106
107 // Tested by:
108
109 // Tested on:
110
111 // Tested by:
112
113 // Tested on:
114
115 // Tested by:
116
117 // Tested on:
118
119 // Tested by:
120
121 // Tested on:
122
123 // Tested by:
124
125 // Tested on:
126
127 // Tested by:
128
129 // Tested on:
130
131 // Tested by:
132
133 // Tested on:
134
135 // Tested by:
136
137 // Tested on:
138
139 // Tested by:
140
141 // Tested on:
142
143 // Tested by:
144
145 // Tested on:
146
147 // Tested by:
148
149 // Tested on:
150
151 // Tested by:
152
153 // Tested on:
154
155 // Tested by:
156
157 // Tested on:
158
159 // Tested by:
160
161 // Tested on:
162
163 // Tested by:
164
165 // Tested on:
166
167 // Tested by:
168
169 // Tested on:
170
171 // Tested by:
172
173 // Tested on:
174
175 // Tested by:
176
177 // Tested on:
178
179 // Tested by:
180
181 // Tested on:
182
183 // Tested by:
184
185 // Tested on:
186
187 // Tested by:
188
189 // Tested on:
190
191 // Tested by:
192
193 // Tested on:
194
195 // Tested by:
196
197 // Tested on:
198
199 // Tested by:
200
201 // Tested on:
202
203 // Tested by:
204
205 // Tested on:
206
207 // Tested by:
208
209 // Tested on:
210
211 // Tested by:
212
213 // Tested on:
214
215 // Tested by:
216
217 // Tested on:
218
219 // Tested by:
220
221 // Tested on:
222
223 // Tested by:
224
225 // Tested on:
226
227 // Tested by:
228
229 // Tested on:
230
231 // Tested by:
232
233 // Tested on:
234
235 // Tested by:
236
237 // Tested on:
238
239 // Tested by:
240
241 // Tested on:
242
243 // Tested by:
244
245 // Tested on:
246
247 // Tested by:
248
249 // Tested on:
250
251 // Tested by:
252
253 // Tested on:
254
255 // Tested by:
256
257 // Tested on:
258
259 // Tested by:
260
261 // Tested on:
262
263 // Tested by:
264
265 // Tested on:
266
267 // Tested by:
268
269 // Tested on:
270
271 // Tested by:
272
273 // Tested on:
274
275 // Tested by:
276
277 // Tested on:
278
279 // Tested by:
280
281 // Tested on:
282
283 // Tested by:
284
285 // Tested on:
286
287 // Tested by:
288
289 // Tested on:
290
291 // Tested by:
292
293 // Tested on:
294
295 // Tested by:
296
297 // Tested on:
298
299 // Tested by:
300
301 // Tested on:
302
303 // Tested by:
304
305 // Tested on:
306
307 // Tested by:
308
309 // Tested on:
310
311 // Tested by:
312
313 // Tested on:
314
315 // Tested by:
316
317 // Tested on:
318
319 // Tested by:
320
321 // Tested on:
322
323 // Tested by:
324
325 // Tested on:
326
327 // Tested by:
328
329 // Tested on:
330
331 // Tested by:
332
333 // Tested on:
334
335 // Tested by:
336
337 // Tested on:
338
339 // Tested by:
340
341 // Tested on:
342
343 // Tested by:
344
345 // Tested on:
346
347 // Tested by:
348
349 // Tested on:
350
351 // Tested by:
352
353 // Tested on:
354
355 // Tested by:
356
357 // Tested on:
358
359 // Tested by:
360
361 // Tested on:
362
363 // Tested by:
364
365 // Tested on:
366
367 // Tested by:
368
369 // Tested on:
370
371 // Tested by:
372
373 // Tested on:
374
375 // Tested by:
376
377 // Tested on:
378
379 // Tested by:
380
381 // Tested on:
382
383 // Tested by:
384
385 // Tested on:
386
387 // Tested by:
388
389 // Tested on:
390
391 // Tested by:
392
393 // Tested on:
394
395 // Tested by:
396
397 // Tested on:
398
399 // Tested by:
400
401 // Tested on:
402
403 // Tested by:
404
405 // Tested on:
406
407 // Tested by:
408
409 // Tested on:
410
411 // Tested by:
412
413 // Tested on:
414
415 // Tested by:
416
417 // Tested on:
418
419 // Tested by:
420
421 // Tested on:
422
423 // Tested by:
424
425 // Tested on:
426
427 // Tested by:
428
429 // Tested on:
430
431 // Tested by:
432
433 // Tested on:
434
435 // Tested by:
436
437 // Tested on:
438
439 // Tested by:
440
441 // Tested on:
442
443 // Tested by:
444
445 // Tested on:
446
447 // Tested by:
448
449 // Tested on:
450
451 // Tested by:
452
453 // Tested on:
454
455 // Tested by:
456
457 // Tested on:
458
459 // Tested by:
460
461 // Tested on:
462
463 // Tested by:
464
465 // Tested on:
466
467 // Tested by:
468
469 // Tested on:
470
471 // Tested by:
472
473 // Tested on:
474
475 // Tested by:
476
477 // Tested on:
478
479 // Tested by:
480
481 // Tested on:
482
483 // Tested by:
484
485 // Tested on:
486
487 // Tested by:
488
489 // Tested on:
490
491 // Tested by:
492
493 // Tested on:
494
495 // Tested by:
496
497 // Tested on:
498
499 // Tested by:
500
501 // Tested on:
502
503 // Tested by:
504
505 // Tested on:
506
507 // Tested by:
508
509 // Tested on:
510
511 // Tested by:
512
513 // Tested on:
514
515 // Tested by:
516
517 // Tested on:
518
519 // Tested by:
520
521 // Tested on:
522
523 // Tested by:
524
525 // Tested on:
526
527 // Tested by:
528
529 // Tested on:
530
531 // Tested by:
532
533 // Tested on:
534
535 // Tested by:
536
537 // Tested on:
538
539 // Tested by:
540
541 // Tested on:
542
543 // Tested by:
544
545 // Tested on:
546
547 // Tested by:
548
549 // Tested on:
550
551 // Tested by:
552
553 // Tested on:
554
555 // Tested by:
556
557 // Tested on:
558
559 // Tested by:
560
561 // Tested on:
562
563 // Tested by:
564
565 // Tested on:
566
567 // Tested by:
568
569 // Tested on:
570
571 // Tested by:
572
573 // Tested on:
574
575 // Tested by:
576
577 // Tested on:
578
579 // Tested by:
580
581 // Tested on:
582
583 // Tested by:
584
585 // Tested on:
586
587 // Tested by:
588
589 // Tested on:
590
591 // Tested by:
592
593 // Tested on:
594
595 // Tested by:
596
597 // Tested on:
598
599 // Tested by:
600
601 // Tested on:
602
603 // Tested by:
604
605 // Tested on:
606
607 // Tested by:
608
609 // Tested on:
610
611 // Tested by:
612
613 // Tested on:
614
615 // Tested by:
616
617 // Tested on:
618
619 // Tested by:
620
621 // Tested on:
622
623 // Tested by:
624
625 // Tested on:
626
627 // Tested by:
628
629 // Tested on:
630
631 // Tested by:
632
633 // Tested on:
634
635 // Tested by:
636
637 // Tested on:
638
639 // Tested by:
640
641 // Tested on:
642
643 // Tested by:
644
645 // Tested on:
646
647 // Tested by:
648
649 // Tested on:
650
651 // Tested by:
652
653 // Tested on:
654
655 // Tested by:
656
657 // Tested on:
658
659 // Tested by:
660
661 // Tested on:
662
663 // Tested by:
664
665 // Tested on:
666
667 // Tested by:
668
669 // Tested on:
670
671 // Tested by:
672
673 // Tested on:
674
675 // Tested by:
676
677 // Tested on:
678
679 // Tested by:
680
681 // Tested on:
682
683 // Tested by:
684
685 // Tested on:
686
687 // Tested by:
688
689 // Tested on:
690
691 // Tested by:
692
693 // Tested on:
694
695 // Tested by:
696
697 // Tested on:
698
699 // Tested by:
700
701 // Tested on:
702
703 // Tested by:
704
705 // Tested on:
706
707 // Tested by:
708
709 // Tested on:
710
711 // Tested by:
712
7
```

```

71         }
72         break;
73     case 3: // Logical NOT
74         if (!num3) {
75             printf("num3 is zero (logical NOT of\n\n");
76             num3).\\n");
77         }
78         break;
79     default:
80         printf("Error! Invalid operator.\\n");
81     }
82     break;
83 }
84 case 3: {
85     unsigned int num1, num2, result;
86     char binaryStr1[33], binaryStr2[33];
87
88     printf("Select two decimal numbers:\\n");
89     printf("Enter first decimal number: ");
90     scanf("%u", &num1);
91     printf("Enter second decimal number: ");
92     scanf("%u", &num2);
93
94     // Перетворення в двійковий формат
95     decimalToBinary(num1, binaryStr1);
96     decimalToBinary(num2, binaryStr2);
97

```

```

96     printf("Binary representation of first number: %s\\n",
97           , binaryStr1);
98     printf("Binary representation of second number:
99           %s\\n", binaryStr2);
100
101     int bitwiseOperation;
102     printf("Choose the type of bitwise operator: (AND(1
103           ), OR(2), XOR(3), NOT(4), LSHIFT(5), RSHIFT(6)):
104           ");
105     scanf("%d", &bitwiseOperation);
106
107     switch (bitwiseOperation) {
108     case 1: // Bitwise AND
109         result = num1 & num2;
110         printf("num1 AND num2: %u (Binary: ", result
111               );
112         decimalToBinary(result, binaryStr1);
113         printf("%s\\n", binaryStr1);
114         break;
115     case 2: // Bitwise OR
116         result = num1 | num2;
117         printf("num1 OR num2: %u (Binary: ", result
118               );
119         decimalToBinary(result, binaryStr1);
120         printf("%s\\n", binaryStr1);
121         break;
122     case 3: // Bitwise XOR
123

```

```

124         result = num1 ^ num2;
125         printf("num1 XOR num2: %u (Binary: ", result
126               );
127         decimalToBinary(result, binaryStr1);
128         printf("%s\\n", binaryStr1);
129         break;
130     case 4: // Bitwise NOT
131         result = ~num1;
132         printf("NOT num1: %u (Binary: ", result);
133         decimalToBinary(result, binaryStr1);
134         printf("%s\\n", binaryStr1);
135         break;
136     case 5: // Left Shift
137         result = num1 << 1; // Зсув вліво на 1
138         printf("num1 << 1: %u (Binary: ", result);
139         decimalToBinary(result, binaryStr1);
140         printf("%s\\n", binaryStr1);
141         break;
142     case 6: // Right Shift
143         result = num1 >> 1; // Зсув вправо на 1
144         printf("num1 >> 1: %u (Binary: ", result);
145         decimalToBinary(result, binaryStr1);
146         printf("%s\\n", binaryStr1);
147         break;
148     default:
149         printf("Error! Invalid operator.\\n");
150     }
151     break;

```

```

125         printf("NOT num1: %u (Binary: ", result);
126         decimalToBinary(result, binaryStr1);
127         printf("%s)\n", binaryStr1);
128         break;
129     case 5: // Left Shift
130         result = num1 << 1; // Зсув вліво на 1
131         printf("num1 << 1: %u (Binary: ", result);
132         decimalToBinary(result, binaryStr1);
133         printf("%s)\n", binaryStr1);
134         break;
135     case 6: // Right Shift
136         result = num1 >> 1; // Зсув вправо на 1
137         printf("num1 >> 1: %u (Binary: ", result);
138         decimalToBinary(result, binaryStr1);
139         printf("%s)\n", binaryStr1);
140         break;
141     default:
142         printf("Error! Invalid operator.\n");
143     }
144     break;
145 }
146 default:
147     printf("Error! Invalid operator.\n");
148 }
149
150 return 0;
151 }
152

```

2.написав програму з введенням числа та виведенням її адреси та значення через вказівник

```

1  #include <stdio.h>
2
3  int main() {
4      int number;
5      int *pointer = &number;
6
7
8      printf("Enter an integer: ");
9      scanf("%d", &number);
10
11     printf("The value of the variable 'number': %d\n", *pointer);
12     printf("The address of the variable 'number': %p\n", (void *)pointer);
13
14     return 0;
15 }
16

```

3.Написав програму розв'язку квадратного рівняння з коефіцієнтами які вводяться з консолі

```

1  #include <stdio.h>
2  #include <math.h>
3
4  int main() {
5      double a, b, c;
6      double discriminant, root1, root2, realPart, imaginaryPart;
7
8
9      printf("Enter coefficients a, b and c: ");
10     scanf("%lf %lf %lf", &a, &b, &c);
11
12
13     if (a == 0) {
14         printf("Coefficient 'a' cannot be zero in a quadratic
15             equation.\n");
16         return 1;
17     }
18
19     discriminant = b * b - 4 * a * c;
20
21
22     if (discriminant > 0) {
23
24         root1 = (-b + sqrt(discriminant)) / (2 * a);
25         root2 = (-b - sqrt(discriminant)) / (2 * a);
26         printf("Roots are real and different.\n");
27         printf("Root 1 = %.2lf\n", root1);

```

```

21
22     if (discriminant > 0) {
23
24         root1 = (-b + sqrt(discriminant)) / (2 * a);
25         root2 = (-b - sqrt(discriminant)) / (2 * a);
26         printf("Roots are real and different.\n");
27         printf("Root 1 = %.2lf\n", root1);
28         printf("Root 2 = %.2lf\n", root2);
29     } else if (discriminant == 0) {
30
31         root1 = -b / (2 * a);
32         printf("Roots are real and the same.\n");
33         printf("Root = %.2lf\n", root1);
34     } else {
35         i
36         realPart = -b / (2 * a);
37         imaginaryPart = sqrt(-discriminant) / (2 * a);
38         printf("Roots are complex and different.\n");
39         printf("Root 1 = %.2lf + %.2lfi\n", realPart,
40             imaginaryPart);
41         printf("Root 2 = %.2lf - %.2lfi\n", realPart,
42             imaginaryPart);
43     }
44     return 0;
45 }

```