


SPRAWOZDANIE NR 2

Nazwa ćwiczenia	Instrukcje sterujące – część 1		 POLITECHNIKA BYDGOSKA Wydział Telekomunikacji, Informatyki i Elektrotechniki
Przedmiot	Podstawy programowania - laboratorium		
Student grupa	Marcin Ogórkiewicz, grupa 7		
Data ćwiczeń	19.10.2022	31.10.2022	Data oddania sprawozdania

```
1 #include <stdio.h>
2 #include <math.h>
3
4 void zad1() {
5     printf( format: "Zadanie 1.\nWprowadz 4 liczby rzeczywiste\n");
6     float a;
7     int i = 0, dod = 0, uje = 0;
8     do {
9         printf( format: "Podaj liczbe rzeczywista:\n");
10        scanf( format: "%f", &a);
11        if (a >= 0){
12            dod++;
13        }else uje++;
14        i++;
15    }while (i<4);
16    if (dod>uje){
17        printf( format: "Wprowadzono wiecej liczb dodatnich\n");
18    } else if(dod == uje){
19        printf( format: "Wprowadzono tyle samo liczb dodatnich i ujemnych\n");
20    }else (printf( format: "Wprowadzono wiecej liczb ujemnych\n"));
21    }
22
23 void zad2(){
24     int n;
25     printf( format: "Zadanie 2.\nWprowadz liczbe n aby uzyskac wartosc bezwzgledna\n");
26     scanf( format: "%d", &n);
27     if (n<0){
28         n=n*-1;
29     }
30     printf( format: "Wartosc bezwzgledna z n: %d\n", n);
31 }
```

```
32
33 void zad3(){
34     float x, y;
35     printf( format: "Zadanie 3.\nPodaj y aby otrzymac x\n");
36     scanf( format: "%f", &y);
37     if (y<0){
38         x=-y;
39     } else{
40         x=y;
41     }
42     printf( format: "%f\n",x);
43 }
44
45 void zad4(){
46     printf( format: "Zadanie 4. Kalkulator\n");
47     float x, y, dzialanie;
48     char z;
49     printf( format: "Podaj dwie liczby rzeczywiste oraz operacje, ktora chcesz na nich wykonac(znak) wedlug wzoru: liczba, liczba, znak\n");
50     scanf( format: "%f, %f, %c", &x, &y, &z);
51     switch (z) {
52         case '+':
53             dzialanie=x+y;
54             printf( format: "Twoje dzialanie %f %c %f = %f\n", x, z, y, dzialanie);
55             break;
56         case '-':
57             dzialanie=x-y;
58             printf( format: "Twoje dzialanie %f %c %f = %f\n", x, z, y, dzialanie);
59             break;
60         case '*':
61             dzialanie=x*y;
62             printf( format: "Twoje dzialanie %f %c %f = %f\n", x, z, y, dzialanie);
63             break;
64         case '/':
65             dzialanie=x/y;
66             printf( format: "Twoje dzialanie %f %c %f = %f\n", x, z, y, dzialanie);
67             break;
68     }
```

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help main.c [C:\Users\kokos\CLionProjects\PodsProgSpr2] - main.c
PodsProgSpr2 main.c
main.c x CL\_main.c x
61     dzialanie=x*y;
62     printf( format: "Twoje dzialanie %f %c %f = %f\n", x, z, y, dzialanie);
63     break;
64     case '/':
65         if(y==0){
66             printf( format: "Nie moze wykonac dzielenia\n");
67             break;
68         } else{
69             dzialanie=x/y;
70             printf( format: "Twoje dzialanie %f %c %f = %f\n", x, z, y, dzialanie);
71             break;
72         }
73     default:
74         printf( format: "Niepoprawne dane");
75         break;
76 }
77 }
78
79 void zad5(){
80     char fig;
81     printf( format: "Zadanie 5\nPodaj jakiej figury geometrycznej pole chcesz obliczyc(k - kwadrat, p - prostokat, t - trojkat)\n");
82     scanf( format: "%c", &fig);
83     float a, b, c;
84     while(fig!='k' && fig!='p' && fig!='t'){
85         printf( format: "Podaj poprawny symbol\n");
86         scanf( format: "%c", &fig);
87     }
88     if (fig == 'k'){
89         printf( format: "Podaj dlugosc boku kwadratu\n");
90         scanf( format: "%f",&a);
91     }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
```

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help main.c [C:\Users\kokos\CLionProjects\PodsProgSpr2] - main.c
PodsProgSpr2 main.c
main.c x CL\_main.c x
91     float kwadrat = pow( X, a, Y, 2);
92     printf( format: "Pole kwadratu wynosi: %f\n", kwadrat);
93 }
94 if (fig == 'p'){
95     printf( format: "Podaj dlugosc bokow prostokata\n");
96     scanf( format: "%f, %f", &a, &b);
97     float prostokat = a*b;
98     printf( format: "Pole prostokatu wynosi: %f\n", prostokat);
99 }
100 if (fig == 't'){
101     float najw, mniej;
102     printf( format: "Podaj dlugosc bokow trojkata\n");
103     scanf( format: "%f, %f, %f", &a, &b, &c);
104     if (a > b && a > c){
105         najw = a;
106         mniej = b+c;
107     } else if (b > a && b > c){
108         najw = b;
109         mniej = a+c;
110     } else {
111         najw = c;
112         mniej = a+b;
113     }
114     if (najw < mniej){
115         float obwod = a+b+c;
116         float pole = pow( X, 0.5*obwod*(0.5*obwod-a)*(0.5*obwod-b)*(0.5*obwod-c), Y, 0.5);
117         printf( format: "Pole trojkata zlozonego z podanych bokow wynosi: %f\n", pole);
118     } else{
119         printf( format: "Z podanych bokow nie da sie zlozyc trojkata");
120     }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
```

```
105     najw = a;
106     mniej = b+c;
107 } else if (b > a && b > c){
108     najw = b;
109     mniej = a+c;
110 } else {
111     najw = c;
112     mniej = a+b;
113 }
114 if (najw < mniej){
115     float obwod = a+b+c;
116     float pole = pow((0.5*obwod*(0.5*obwod-a)*(0.5*obwod-b)*(0.5*obwod-c)), 0.5);
117     printf("Pole trojkata zlozonego z podanych bokow wynosi: %f\n", pole);
118 } else{
119     printf("Z podanych bokow nie da sie zlozyc trojkata");
120 }
121 }
122 }
123 }
124
125 int main() {
126     zad1();
127     zad2();
128     zad3();
129     zad4();
130     zad5();
131     return 0;
132 }
133 }
```

```
Run: PodProgSpr2
C:\Users\koku\CLionProjects\PodProgSpr2\cmake-build-debug\PodProgSpr2.exe
Zadanie 1.
Wprowadz 4 liczby rzeczywiste
Podaj liczbe rzeczywista:
1
Podaj liczbe rzeczywista:
2
Podaj liczbe rzeczywista:
3
Podaj liczbe rzeczywista:
4
Wprowadzono tyle samo liczb dodatnich i ujemnych
Zadanie 2.
Wprowadz liczbe n aby uzyskac wartosc bezwzglezna
5
Wartosc bezwzglezna z n: 5
Zadanie 3.
Podaj y aby otrzymac x
12
6.780000
Zadanie 4. Kalkulator
Podaj dwie liczby rzeczywiste oraz operacje, ktora chcesz na nich wykonac(znak) wedlug wzoru: liczba, liczba, znak
18.000000 / -12.000000 = -1.500000
Zadanie 5
Podaj jakiej figury geometrycznej pole chcesz obliczyc(k - kwadrat, p - prostokat, t - trojkat)
Podaj poprawny symbol
t
Podaj dlugosc bokow trojkata
3 4 5
```

```
Podaj liczbe rzeczywista:
Podaj liczbe rzeczywista:
Podaj liczbe rzeczywista:
Wprowadzono tyle samo liczb dodatnich i ujemnych
Zadanie 2.
Wprowadz liczbe n aby uzyskac wartosc bezwzglezna
Wartosc bezwzglezna z n: 5
Zadanie 3.
Podaj y aby otrzymac x
6.700000
Zadanie 4. Kalkulator
Podaj dwie liczby rzeczywiste oraz operacje, ktora chcesz na nich wykonac(znak) wedlug wzoru: liczba, liczba, znak
Twoje dzialanie 18.000000 / -12.000000 = -1.500000
Zadanie 5
Podaj jakiej figury geometrycznej pole chcesz obliczyc(k - kwadrat, p - prostokat, t - trojkat)
Podaj poprawny symbol
Podaj dlugosc bokow trojkata
Pole trojkata zlozonego z podanych bokow wynosi: 6.000000
Process finished with exit code 0
```

Wyjaśnienie kodu:

Zadanie 1

Za pomocą zmiennej `a` typu `float`, pętli `do while` oraz licznika `i`, program pobiera od użytkownika 4 liczby rzeczywiste. Licznik `i` kontroluje ilość wprowadzanych liczb. Po każdej podanej liczbie program sprawdza czy liczba jest dodatnia bądź ujemna i odpowiednio zwiększa wartość licznika dodając (liczby dodatnie) lub ujemnie (liczby ujemne). Na sam koniec, program porównuje wartości tych dwóch liczników i ustala którego typu liczb jest więcej.

Zadanie 2

Program pobiera od użytkownika liczbę całkowitą `n`, aby wypisać jej wartość bezwzględna. Jeżeli `n` jest dodatnie to program wypisuje `n`, a jeśli jest ujemne, program wypisuje wartość $n * (-1)$.

Zadanie 3

Program działający na tej samej zasadzie co ten w Zadaniu 2. Rolę `n` pełni zmienna `y`, a wartość bezwzględną zmienna `x`.

Zadanie 4

Program pobiera od użytkownika dane na temat tego, jakie działanie matematyczne chce wykonać, na dwóch liczbach rzeczywistych. Następnie, przy użyciu polecenia `case`, program wykonuje odpowiednie działanie na liczbach i następnie wyświetla działanie, wraz z jego wynikiem, użytkownikowi.

Zadanie 5

Program prosi użytkownika o wybranie, jakiej figury pole chce obliczyć, poprzez wpisanie odpowiedniej litery. W przypadku podania złego znaku, program prosi użytkownika o poprawny symbol do skutku. Za pomocą warunku `if`, program weryfikuje jakie pole ma obliczyć. Kiedy program ma obliczyć pole kwadratu, wykorzystuje funkcję `pow`, w celu podniesienia boku kwadratu do potęgi drugiej, a następnie wypisuje wynik użytkownikowi. Dla prostokąta, program wykonuje proste mnożenie dwóch boków prostokąta i wypisuje wynik. Dla trójkąta, program wykorzystuje wzór

Herona aby obliczyć jego pole. Najpierw pobiera długości trzech boków trójkąta. Potem sprawdza, który jest najdłuższy, w celu zweryfikowania czy suma dwóch krótszych boków jest większa od długości najdłuższego boku. Jeżeli trójkąt jest poprawny, program liczy obwód trójkąta, a następnie wykorzystuje funkcję pow do wyliczenia pola trójkąta z użyciem wzoru Herona. Na koniec program wypisuje wartość pola trójkąta. W wypadku podania niepoprawnego trójkąta, program kończy działanie i wypisuje odpowiedni komunikat. Wydałbym rozwiązanie tego zadania było zastosowanie polecenia case, ale niestety nie umiałem go w tym wypadku wykorzystać.

Wnioski

Ćwiczenie drugie zaznajomiło mnie nieco z instrukcjami sterującymi. Pozwoliło mi na wykorzystanie polecenia case, którego chcę się nauczyć używać. Cel ćwiczenia został osiągnięty.