```
//pierwiastek
    float a;
    scanf("%f",&a);
    printf("Pierwiastek z x %f\n",sqrt(a));

//bezwzgledna bibioteka
    int a;
    scanf("%i",&a);
    printf("Wartosc bezwzgledna z x %i\n",abs(a));

//bezwzgledna funkcja
    int bezw(int a){
    int bezwzgledna;
    if(a<0) bezwzgledna=-a;
        else bezwzgledna=a;
}

//silnia rekurencja
    int silnia(int x){
      if(x<2)    return 1;
           else return x*silnia(x-1);
}

//zaokraglenie
    float a;
    scanf("%f",&a);
    printf("Zaokraglenie do 2 miejsc %2.2f\n",a);

//notacja
    float a;
    scanf("%f",&a);
    printf("Zapis w notacji wykladniczej %.0e\n",a);

// NWD odejmowanie
    int a,b,NWD=0;
    printf("Podaj liczbe a= ");
    scanf("%d",&a);
    printf("Podaj liczbe b= ");
    scanf("%d",&b);

    while(a!=b){
        if(a>b){
        a=a-b;
        }
        else if(a<b){
            b=b-a;
        }
    }
    NWD=a;
    printf("NWD podanych liczb %d",NWD);

//NWD odejmowanie rekurencja
    int NWD(int a, int b){
    if (a!=b)
    if (a>b) return NWD(a-b, b);
        else return NWD(a, b-a);
}
```

```
//NWD dzielenie
    int NWD(int a, int b){
      int temp;
      while(b>0)
      {
            temp=b;
            b=a%b;
            a=temp;
      }
      return a;
}

//NWD dzielenie rekurencja
    int NWD(int a, int b){
      if (b>0)
            return NWD_r (b, a%b);
}

//rownania 2 niewiadomych
    float a1,b1,c1;
    float a2,b2,c2;
    float W,Wx,Wy;
    float x,y;
    printf("Podaj wspolczynnki pierwszego ukladu\n");
    scanf("%f",&a1);
    scanf("%f",&b1);
    scanf("%f",&c1);
    printf("Podaj wspolczynnki drugiego ukladu\n");
    scanf("%f",&a2);
    scanf("%f",&b2);
    scanf("%f",&c2);

    W=a1*b2-a2*b1;
    Wx=c1*b2-c2*b1;
    Wy=a1*c2-a2*c1;
    x=Wx/W;
    y=Wy/W;
    if(W==0&&Wx==0&&Wy==0) printf("Podany uklad ma nieskonczenie wiele
rozwiazan\n");
        else if(W==0&&(Wx!=0||Wy!=0)) printf("Podany uklad jest
sprzeczny\n");
            else{
        printf("Wynik podanego ukladu rownan %f\n",x);
        printf("Wynik podanego ukladu rownan %f",y);

//wielokrotnosc w przdzialach
    int m,n,k;
    printf("Podaj liczbe ograniczajaca z lewej strony m= ");
    scanf("%d",&m);
    printf("Podaj liczbe ograniczajaca z prawej strony k= ");
    scanf("%d",&k);
    printf("Podaj liczbe do wielokrotnosci n= ");
    scanf("%d",&n);
    if(m>=0&&n>0&&k>0){
        for(int i=n; i<k; i=i+n)
        {
            if (i>m&&i<k) {
                printf(" %d\n",i);
```

```
            }
        }
    }
    else printf("Ktoras z podanej liczby nie jest dodatnia");

//liczba dzieli n i jest <n
    int liczba(int x){
    for(int i=2;i<x;i++){
        if(!(x%i)){
         return x/i;
        }
    return 1;
    }
}

//potega
    int potega(int n)
    {
    int pot=1;
    for(int i=1;i<=2;i++)
    {
        pot*=n;
    }
    return pot;
    }

// n do potegi m
    int potega(int n,int m){
    int wynik=1;
    if((n>0)&&(m>0)){
        for(int i=0;i<m;i++){
            wynik=wynik*n;
        }
    }
    else if((n>0)&&(m==0)){
       wynik=1;
        }
    else if((n==0)&&(m==0)){
        wynik=0;
    }
    return wynik;
    }

//czesc calkowita w dol
    int czcalk(int n)
{
    int i=0;
    while(potega(i)<=n)
    {
        i++;
    }
    return i-1;
}

//zliczanie wywolan funkcji
    void fzlicz(){
    static int S=0;
    S++;
```

```
    printf("Funkcja zostala wywolana %d  razy\n",S);
}

    int main()
{
    fzlicz();
    fzlicz();
    fzlicz();

//zliczanie sumy od ilosci wywolan
    int suma(){
    int n;
    static int S=0;
    printf("Podaj liczbe: ");
    scanf("%d",&n);
    S=S+n;
    printf("Obecna suma %d\n",S);
}

int main()
{
    suma();
    suma();

//ciag fibonacciego rekurencja wypisuje wartosc o indeksie n
    int fib(int n)
{
      if(n<3){
         return 1;
       }
    else{
        return fib(n-1)+fib(n-2);
    }
}

//rekurencja jakiegos przypadku
    int rek(int n,int m){
    if(m==0){
        return n;
    }
        else if(m>n){
                return rek(m,n);
             }
            else{
                return n-m+rek(n-1,m)+rek(n,m-1);
            }
}

//wartosc mniejsza po wskazniku
    int wskaznik(int *x,int *y){

    if(*x<*y) return *x;
        else return *y;
}
    printf("%d\n",wskaznik(&a,&b));

//wskaznik na wartosc mniejsza
    int* wskaznik(int *x,int *y){
```

```
    if(*x<*y) return x;
        else return y;
}
    printf("%p\n",wskaznik(&a,&b));

//suma warto�ci zmiennych wskazywanych przez argumenty
    int suma(int const *x, int const *y){

    return *x+*y;
}
    printf("%d\n",suma(&a,&b));

//przepisuje warto�� n do zmiennej wskazywanej przez w np 2,8 -> 2,2
    void przep(int n,int *w){
    *w=n;
}
    przep(a,&b);
    printf("%d, %d",a,b);

//bezargumentowa funckja rezerwuje pami�� dla pojedynczej zmiennej typu
int i zwraca jako warto�� wska�nik do niej
    int* bf(){
    return malloc(sizeof(int));
}
//rezerwuje w pami�ci blok n zmiennych typu int i zwraca jako warto��
wska�nik do pocz�tku zarezerwowanego bloku pami�ci
    int* bf(int n){
    return malloc(n*sizeof(int[n]));
}

    int* bf(int n){
    return malloc(n*sizeof(double[n]));
}

//przepisuje zawarto�� sta�ej wskazywanej przez pierwszy argument do
zmiennej wskazywanej przez drugi argument
    void funk(const int *x,int *y){
    *y=*x;
}
//zamiana wartosci 2 zmiennych
    void zamiana(int *a,int *b){
    int temp=0;
    temp=*a;
    *a=*b;
    *b=temp;
}
// zamiana i sortowanie niemalejace 3 wartosci
    void zamiana(int *a,int *b){
    int temp=0;
    temp=*a;
    *a=*b;
    *b=temp;
}
void sort(int *a, int *b, int *c){
        if(*a>*b) zamiana(a,b);
        if(*a>*c) zamiana(a,c);
```

```
        if(*b>*c) zamiana(b,c);
}

//przepisywanie z jednej tablicy do drugiej
    void FuncA(int n, int tab1[], int tab2[]) {
      for (int i = 0; i < n; i++)
            tab2[i] = tab1[i];
}
//przepisywanie z jednej tablicy do drugiej odwrotnie
    void FuncB(int n, int tab1[], int tab2[]) {
      for (int i = 0; i < n; i++)
            tab2[i] = tab1[n - i - 1];
}

//scalanie 2 tablic
    void scal(int n,double tab1[], double tab2[], double tab3[]){
    for(int i=0;i<n;i++){
        tab3[i]=tab1[i];
        tab3[i+5]=tab2[i];
    }
}
    void pisz(double tab3[]){
    for(int i=0;i<10;i++){
        printf("%f ",tab3[i]);
    }
}

//scalanie 2 tablic dla jednej indeksy parzyste dla drugiej nie
    void scal(int n, double tab1[], double tab2[], double tab3[]){
    for(int i=0;i<2*n;i++){
        if(i%2==0) tab3[i]=tab2[i/2];
            else tab3[i]=tab1[i/2];
    }
}

//iloczyn skalarny wektorow
    double skal(int n,double tab1[], double tab2[]){
    double S=0;
    for(int i=0;i<n;i++){
        S=S+tab1[i]*tab2[i];
    }
    return S;
}

//sortowanie tablicy rosnaco
    void sort(int n, int tab[]){
    int temp=0;
    for(int i=0; i<n-1; i++){
            for(int j=0; j<n-i-1; j++){
                  if (tab[j] > tab[j + 1]){
                        temp = tab[j];
                        tab[j] = tab[j + 1];
                        tab[j + 1] = temp;
                  }
            }
      }
}
```

```
//sortowanie tablicy malejaco
    void sort(int n, int tab[]){
    int temp=0;
    for(int i=0; i<n-1; i++){
           for(int j=0; j<n-i-1; j++){
                 if (tab[j] < tab[j + 1]){
                       temp = tab[j];
                       tab[j] = tab[j + 1];
                       tab[j + 1] = temp;
                 }
           }
      }
}

//dynamiczna tablica n elementowa i zwraca wskaznik na 1 element
    double* dyna(int n){
    double *tab=malloc(n*sizeof(double));
    return tab;
}

//zwalnia pami[illegible] zajmowan[illegible] przez przekazan[illegible] w argumencie tablic[illegible]
    void dyna(double*wskaznik){
    free(wskaznik);
}

//dostaje tablice, tworzy jej kopie i zwraca wskaznik do nowej kopii
    double* dyna(int n, double tab1[]){
    double *tab2=malloc(n*sizeof(double));
    for(int i=0;i<n;i++){
        tab2[i]=tab1[i];
    }
    return &tab2[0];
}

//przepisuje do nowo utworzonej tablicy elementy rozne od 0, zwraca
wskaznik na pierwszy element tablicy 2
    int* dyna(int n, int tab1[]){
    int S=0;
    for(int i=0;i<n;i++){
        if(tab1[i]!=0) S+=1;
    }

    int *tab2=malloc(S*sizeof(int));
    for(int i=0;i<S;i++){
        if(tab1[i]!=0) tab2[i]=tab1[i];
    }
     return &tab2[0];
}
```