ALGORYTMY MATURALNE:

1. Czy liczba jest pierwsza:

```
bool czy_pierwsza(int n)
{
	if(n<2) return false; //gdy liczba jest mniejsza ni¿ 2 to nie jest pierwsz¹

	for(int i=2;i*i<=n;i++)
	{
		if(n%i==0) return false; //gdy znajdziemy dzielnik, to dana liczba nie jest pierwsza
	}
	return true;
}
```

2. Rozklad liczby na czynniki pierwsze:

```
int n;
cin>>n;
int k=2; //ustawiamy k na pierwsz¹ liczbê pierwsz¹
//rozk³ad liczby na czynniki pierwsze
while(n>1&&k*k<=n)
{
   while(n%k==0) //dopóki liczba jest podzielna przez k
   {
      cout<<k<<" ";
      n/=k;
   }
	++k;
}
if(n>1) cout<<n;
cout<<endl;
```

3. Zamiana liczby n z systemu dziesietnego na dowolny system p (od 2 do 16):

```
void zamiana(long long n, int p , string &wynik)
{
   wynik = "";
   if(n>0)
   {
      zamiana(n/p,p,wynik);
      if(n%p>9)
      {                                          //dla systemów o podstawie wiêkszej ni¿ 9 cyfry s¹ literami
         switch(n%p)
        {
           case 10:
                wynik += 'A';
                break;
           case 11:
                wynik += 'B';
                break;
           case 12:
              wynik += 'C';
              break;
           case 13:
              wynik += 'D';
```

```
                break;
             case 14:
                wynik += 'E';
                break;
             case 15:
                wynik += 'F';
                break;
          }
           }
      else wynik += (char) (n%p + 48);
           }
}

int main()
{
    string w;
    zamiana (100 , 16 , w );
    cout << w ;
}

////////////////////////////////

int nadziesietne(string liczba)
{
          int x; //z jakiego systemu
          int wynik=liczba[0]-48;
          for (int i=1; i<liczba.size(); i++)
          {
          wynik=wynik*x+liczba[i]-48;
          }
          return wynik;
}

////////////////////////////////

string zdziesietnego (int licz, int p)
{
   string pom = "";
   while (licz > 0)
   {
      pom = (char) (licz % p + 48 ) + pom;
      licz /= p;
   }
   return pom;
}
```

4. Ciag Fibonacciego:

```
void fibonacci(int n)
{
   long long a = 0, b = 1;
   for(int i=0;i<n;i++)
   {
      cout<<b<<" ";
      b += a; //pod zmienn¹ b przypisujemy wyraz nastêpny czyli a+b
      a = b-a; //pod zmienn¹ a przypisujemy wartoœæ zmiennej b
   }
```

```
}
```

5. Wyszukiwanie binarne:

```
long long tab[1000000]; //tablica z posortowanymi elementami

//l - lewy index tablicy, p - prawy index tablicy
int szukaj(int l, int p, long szukana)
{
	int sr;
	while(l<=p)
	{
		sr = (l + p)/2;
		if(tab[sr] == szukana) return sr;
		if(tab[sr] > szukana) p = sr - 1;
		else l = sr + 1;
	}
	return -1; //zwracamy -1, gdy nie znajdziemy elementu
}

int main()
{
	long long n,szukana;

	cin>>n; //podajemy iloœæ elementów do wczytania n < 1000000

	for(int i=0;i<n;i++)
	{
		cin>>tab[i]; //wczytanie elementów tablicy
	)

	cin>>szukana;

	int pozycja = szukaj(0,n-1,szukana);

	if(pozycja==-1) cout<<"Liczba "<<szukana<<" nie wystêpuje w zbiorze"<<endl;
	else cout<<"Liczba "<<szukana<<" wystêpuje w zbiorze w komórce o numerze "<<pozycja<<endl;

	system("pause");
	return 0;
}
```

6. Znajdowanie najmniejszego lub najwiekszego elementu w zbiorze:

```
int main()
{
  //inicjacja tablicy z przyk³adowymi danymi
	int min, tab[]={2, 3, 6, 7, 8, 7, 4, 3, 1, 7};

	min = tab[0]; //pierwsz¹ liczbê przypisujemy do zmiennej min

	for(int i=1;i<10;i++) //przeszukanie pozosta³ych 9 liczb
		if(min>tab[i])
			min = tab[i];

	cout<<"Najmniejsz¹ wczytan¹ liczb¹ jest "<<min<<endl;
```

```
	system("pause");
	return 0;
}

//analogicznue dla MAX ^
```

7. Znajdowanie najmniejszego i najwiekszego elementu w zbiorze:

```
int main()
{
	int tab[] = {1, 3, 4, 5, 2, 0, 6, 9, 8}, MIN, MAX;

	MIN = MAX = tab[0];

	for(int i=1;i<9;i++)
	{
		if(tab[i]>MAX)
			MAX=tab[i];
		if(tab[i]<MIN)
			MIN=tab[i];
	}

	cout<<"MAX: "<<MAX<<"\nMIN: "<<MIN<<endl;

	system("pause");
	return 0;
}
```

8. NWD - Algorytm Euklidesa:

```
int NWD(int a, int b)
{
  int pom;
	while(b!=0)
  {
		pom=b;
		b=a%b;
		a=pom;
	}
  return a;
}

//NWW(a,b)=(a*b)/NWD(a,b)
```

9. Palindrom:

```
bool czy_palindrom(char tab[])
{
	//ustawiam liczniki "i" i "j" na pierwszy i ostatni znak wyrazu tab
	int i=0, j = strlen(tab)-1; //pamiêtajmy, ¿e indeksujemy tablicê od 0

	while(i<j) //pêtla wykonuje siê do momentu spotkania liczników
	{
		if(tab[i]!=tab[j]) //gdy znaki nie bêd¹ siê zgadzaæ, to wyraz nie jest palindromem
			return false;

		++i; //przejscie do nastêpnej litery id¹c w praw¹ stronê
```

```
		--j; //przejscie do poprzedniej litry id¹c w lew¹ stronê
	}

	return true; //wyraz jest palindromem
}

int main()
{
	char tab[100];
	cout<<"Podaj wyraz: ";
	cin>>tab;

	if(czy_palindrom(tab)) //lub if(czy_palindrom(tab)==true) lub if(czy_palindrom(tab)==1)
		cout<<"Wyraz "<<tab<<" jest palindromem"<<endl;
	else
		cout<<"Wyraz "<<tab<<" nie jest palindromem"<<endl;

	system("pause");
	return 0;
}
```

10. Wyszukiwanie wzorca w tekscie:

```
//Definicja. Wzorzec to spójny podci¹g (podtekst), który wystêpuje w danym ci¹gu znaków.

int main()
{
	char tekst[100], wzorzec[100];
	cout<<"Podaj tekst: ";
	cin.getline(tekst,100);
	cout<<"Podaj wzorzec: ";
	cin.getline(wzorzec,100);

	//naiwne wyszukiwanie wzorca w tekscie

	int t = 0, w; //do poruszania siê po tablicach znaków
	int dl_t = strlen(tekst), dl_w = strlen(wzorzec);
	bool ok = 0;

	for(int i=0; i <= dl_t - dl_w; i++)
	{
	ok = 1;
	//sprawdzamy, czy zgadzaj¹ siê pozosta³e znaki
	for(int j=0; j<dl_w; j++)
	if(tekst[j+i]!=wzorzec[j]) //jesli nie zgadzaj¹ siê
	{
		ok = 0;  //gdy tu wejdziemy, to ok = 0
		break;
	}
	if(ok) //jesli wszystkie litery siê zgadzaj¹ (ok = 1)
	{
		cout<<"Wzorzec znaleziono. Pocz¹tek na "<<i+1<<" pozycji\n";
			cout<<tekst<<endl;
	cout.fill(' ');
		cout.width(i+dl_w);
			cout<<wzorzec<<endl;
		break;
```

```
	}
	}
	if(!ok) cout<<"Wzorca nie znaleziono!";

	cin.get();
	return 0;
}
```

11. Schemat Hornera:

```
int horner(int wsp[],int st, int x)
{
	int wynik = wsp[0];

	for(int i=1;i<=st;i++)
		wynik = wynik*x + wsp[i];

	return wynik;
}

int main()
{
	int *wspolczynniki;
	int stopien, argument;

	cout<<"Podaj stopieñ wielomianu: ";
	cin>>stopien;

	wspolczynniki = new int [stopien+1];

	//wczytanie wspó³czynników
	for(int i=0;i<=stopien;i++)
	{
		cout<<"Podaj wspó³czynnik stoj¹cy przy potêdze "<<stopien-i<<": ";
		cin>>wspolczynniki[i];
	}
	cout<<"Podaj argument: ";
	cin>>argument;

	cout<<"W( "<<argument<<" ) = "<<horner(wspolczynniki,stopien,argument)<<endl;

	delete [] wspolczynniki;
	system("pause");
	return 0;
}
```

12. Sortowanie babelkowe:

```
void sortowanie_babelkowe(int tab[],int n)
{
	for(int i=0;i<n;i++)
		for(int j=1;j<n-i;j++) //pêtla wewnêtrzna
		if(tab[j-1]>tab[j])
		{
			//zamiana miejscami
			swap(tab[j-1],tab[j]);
		}
```

```
}

int main()
{
	int *tab, n;
	cout<<"Ile liczb bêdziesz chcia³ posortowaæ? ";
	cin>>n;
	tab = new int [n]; //przydzielenie pamiêci na elementy tablicy
	//wczytanie liczb
	for(int i=0;i<n;i++)
	{
		cin>>tab[i];
	}
	sortowanie_babelkowe(tab,n);
	//wypisanie posortowanych elementów
	for(int i=0;i<n;i++)
	{
		cout<<tab[i]<<" ";
	}
	cout<<endl;
	system("pause");
	return 0;
}
```

13. Szybkie sortowanie:

```
void quick_sort(int *tab, int lewy, int prawy)
{
	if(prawy <= lewy) return;

	int i = lewy - 1, j = prawy + 1,
	pivot = tab[(lewy+prawy)/2]; //wybieramy punkt odniesienia

	while(1)
	{
		//szukam elementu wiekszego lub rownego piwot stojacego
		//po prawej stronie wartosci pivot
		while(pivot>tab[++i]);

		//szukam elementu mniejszego lub rownego pivot stojacego
		//po lewej stronie wartosci pivot
		while(pivot<tab[--j]);

		//jesli liczniki sie nie minely to zamieñ elementy ze soba
		//stojace po niewlasciwej stronie elementu pivot
		if( i <= j)
			//funkcja swap zamienia wartosciami tab[i] z tab[j]
			swap(tab[i],tab[j]);
		else
			break;
	}

	if(j > lewy)
	quick_sort(tab, lewy, j);
	if(i < prawy)
	quick_sort(tab, i, prawy);
}
```

```
int main()
{
	int *tab, n;
	//cout<<"Ile liczb bêdziesz chcia³ posortowaæ? ";
	cin>>n;
	tab = new int [n]; //przydzielenie pamiêci na elementy tablicy
	//wczytanie liczb
	for(int i=0;i<n;i++)
	{
		cin>>tab[i];
	}
	quick_sort(tab,0, n-1);

	//wypisanie posortowanych elementów
	for(int i=0;i<n;i++)
       cout<<tab[i]<<" ";

	cin.ignore();
	cin.get();
	return 0;
}
```

14. Szyfr przestawieniowy - Zamiana sasiadujacych liter:

```
void kodowanie(char *napis)
{
	int dl = strlen(napis); //wyznaczenie liczby znaków

	for(int i=0; i<dl-1; i+=2) //przesuwamy siê o dwa znaki
	//zamiana s¹siaduj¹cych znaków
	{
		char pom = napis[i];
		napis[i] = napis[i+1]; //dlatego w pêtli i<dl-1
		napis[i+1] = pom;
	}
}

int main()
{
	char napis[100];

	cout<<"Podaj napis do zaszyfrowania: ";
	cin.getline(napis, 100);

	cout<<"Przed szyfrowaniem: ";
	cout<<napis<<endl;

	//szyfrujemy
	kodowanie(napis);

	cout<<"Szyfrogram: ";
	cout<<napis<<endl;

	//deszyfrujemy
	kodowanie(napis);

	cout<<"Tekst jawny: ";
```

```
	cout<<napis<<endl;

	cin.get();
	return 0;
}
```

15. Szyfr przestawieniowy - Przesuniecie spolglosek o jedno miejsce:

```
bool czy_spolgloska(char litera)
{
	//sprawdzamy czy samog³oska - jest ich mniej
	switch(litera)
	{
		case 'a':
		case 'e':
		case 'i':
		case 'o':
		case 'u':
		case 'y':
			return 0;
	}
	return 1;
}

void kodowanie(char *napis)
{
	int dl = strlen(napis); //wyznaczenie liczby znaków

	//nale¿y zapamiêtaæ pozycjê pierwszej spó³g³oski
	bool f=1;	//czy dana spó³g³oska jest pierwsza
	int nr;	//pozycja pierwszej spó³g³oski
	char first;	//przechowujemy spó³g³oskê do podmiany

	for(int i=0;i<dl;i++)
	{
		if(czy_spolgloska(napis[i]))
		{
			if(f)	//jesli wczytana spó³g³oska jest pierwsza
			{		//to j¹ zapamiêtujemy
				nr = i;
				first = napis[i];
				f = 0;
			}
			else //podmiana
			{
				char pom = napis[i];
				napis[i] = first;
				first = pom;
			}
		}

	}
	if(!f) napis[nr] = first;
}

void dekodowanie(char *napis)
{
```

```
	int dl = strlen(napis); //wyznaczenie liczby znaków

	//nale¿y zapamiêtaæ pozycjê pierwszej spó³g³oski
	bool f=1;	//czy dana spó³g³oska jest pierwsza
	int nr;	//pozycja pierwszej spó³g³oski
	char first;	//przechowujemy spó³g³oskê do podmiany

	for(int i=dl-1;i>=0;i--)
		{
			if(czy_spolgloska(napis[i]))
			{
				if(f)	//jesli wczytana spó³g³oska jest ostatnia
				{		//to j¹ spamiêtujemy
					nr = i;
					first = napis[i];
					f = 0;
				}
				else //podmiana
				{
					char pom = napis[i];
					napis[i] = first;
					first = pom;
				}
			}

		}
		if(!f)
			napis[nr] = first;
}

int main()
{
	char napis[100];
	cout<<"Podaj napis do zaszyfrowania: ";
	cin>>napis;
	cout<<"Przed szyfrowaniem: ";
	cout<<napis<<endl;
	//szyfrujemy
	kodowanie(napis);
	cout<<"Szyfrogram: ";
	cout<<napis<<endl;
	//deszyfrujemy
	dekodowanie(napis);
	cout<<"Tekst jawny: ";
	cout<<napis<<endl;
	cin.get();
	return 0;
}
```

16. Szyfr Cezara - male litery:

```
void szyfruj(int klucz, char tab[])
{
	int dl = strlen(tab); //okreœlenie iloœci znaków wyrazu
	//sprawdzenie, czy klucz miesci sie w zakresie
	if(!(klucz >= -26 && klucz <= 26)) return;
```

```
	if(klucz >= 0)
	{
		for(int i=0;i<dl;i++)
		{
			if(tab[i] + klucz <= 'Z') tab[i] += klucz;
			else tab[i] = tab[i] + klucz - 26;
		}
	}
	else
	{
		for(int i=0;i<dl;i++)
		{
			if(tab[i] + klucz >= 'A') tab[i] += klucz;
			else tab[i] = tab[i] + klucz + 26;
		}

	}
}

int main()
{
	char tab[1001]; //tablica znaków - max 1000 znaków.
	int klucz;
	cout<<"Podaj wyraz sk³adaj¹cy siê z du¿ych liter: ";
	cin>>tab;
	cout<<"Podaj klucz z przedzia³u [-26..26]: ";
	cin>>klucz;
	szyfruj(klucz,tab); //szyfrowanie
	cout<<"Po zaszyfrowaniu: "<<tab<<endl;
	szyfruj(-klucz,tab); //deszyfrowanie
	cout<<"Po rozszyfrowaniu: "<<tab<<endl;
	system("pause");
	return 0;
}
```

17. Szyfr Cezara - duze litery:

```
void szyfruj(int klucz, char tab[])
{
	int dl = strlen(tab); //okreœlenie iloœci znaków wyrazu
	//sprawdzenie, czy klucz miesci sie w zakresie
	if(!(klucz >= -26 && klucz <= 26)) return;

	if(klucz >= 0)
	{
		for(int i=0;i<dl;i++)
		{
			if(tab[i] + klucz <= 'z') tab[i] += klucz;
			else tab[i] = tab[i] + klucz - 26;
		}
	}
	else
	{
		for(int i=0;i<dl;i++)
		{
			if(tab[i] + klucz >= 'a') tab[i] += klucz;
			else tab[i] = tab[i] + klucz + 26;
```

```
                }

        }
}
```

18. Szyfr Cezara - uniwersalny:

```
inline int sprawdz(char znak)
{
        //jesli jest mala litera
        if(znak >= 'a' && znak <= 'z') return 0;
        //jesli jest duza litera
        if(znak >= 'A' && znak <= 'Z') return 1;
        //inna ni¿ litera
        return 2;
}

void szyfruj(int klucz, string &tab)
{
        //sprawdzenie, czy klucz miesci sie w zakresie
        if(!(klucz >= -26 && klucz <= 26)) return;

        int pom;
        char a, z;

        for(int i = 0; i < tab.size(); i++)
        {
                pom = sprawdz(tab[i]);
                //ustalienie wielkosci litery
                if(pom < 2)
                {
                        if(pom == 0)
                                a = 'a', z = 'z';
                        else
                                a = 'A', z = 'Z';

                        if(klucz >= 0)
                        {
                                if(tab[i] + klucz <= z) tab[i] += klucz;
                                else tab[i] = tab[i] + klucz - 26;
                        }
                        else
                        {
                                if(tab[i] + klucz >= a) tab[i] += klucz;
                                else tab[i] = tab[i] + klucz + 26;
                        }
                }
        }
}

int main()
{
        string tab;
        int klucz;
        cout<<"Podaj zdanie do zaszyfrowania: ";
        getline(cin, tab);
        cout<<"Podaj klucz z przedzia³u [-26..26]: ";
```

```
        cin>>klucz;
        szyfruj(klucz,tab); //szyfrowanie
        cout<<"Po zaszyfrowaniu: "<<tab<<endl;
        szyfruj(-klucz,tab); //deszyfrowanie
        cout<<"Po rozszyfrowaniu: "<<tab<<endl;
        system("pause");
        return 0;
}
```

19. Sito Eratostenesa:

```
void sito(bool *tab, unsigned int n)
{
        for (int i=2; i*i<=n; i++) //przeszukujemy kolejnych kandydatów na pierwsze
  {                                                    //wystarczy sprawdziæ do pierwiastka z n
                                                       // i<=sqrt(n) - podnosz¹c do kwadratu mamy
                                                       // i*i <= n
    if(!tab[i])                          //jesli liczba jest pierwsza(ma wartosc 0)
                for (int j = i*i ; j<=n; j+=i) //to wykreslamy jej wielokrotnosci
    {
                        tab[j] = 1;             //ustawiaj¹c wartosæ na 1
                }
  }
}

int main()
{
        int n;
        bool *tab;
        cout<<"Podaj zakres górny przedzia³u: ";
        cin>>n;
        tab = new bool [n+1];
        for(int i=2; i<=n; i++) //zerowanie tablicy
        {
                tab[i] = 0;
        }
        sito(tab, n); //przesianie liczb
        cout<<"Kolejne liczby pierwsze z przedzia³u [2.."<<n<<"]: ";
        for(int i=2;i<=n;i++)
        {
                if(!tab[i]) cout<<i<<" ";
        }
        cout<<endl;
        delete []tab;
        system("pause");
  return 0;
}
```

20. Przeszukiwanie liniowe z wartownikiem:

```
int main()
{
  int tab[1001], i = 0, szukana;
  //wczytanie elementów tablicy, ostatnim elementem jaki wczytamy jest wartownik = -1
  do
        {
    cin>>tab[i];
```

```
  }
        while(tab[i++]!=-1);
  //podanie liczby do wyszukania
  cin>>szukana;
  i = 0;
  //przeszukanie tablicy
  while(tab[i]!=szukana && tab[i]!=-1) ++i;
  //koniec wyszukiwania
  //jeœli zatrzymaliœmy siê na wartowniku, to oznacza,
  //¿e szukana liczba nie wystêpuje w zbiorze
  if(tab[i] == -1)
    cout<<"Szukany element nie wystêpuje w tablicy"<<endl;
  else
    cout<<"Liczba "<<szukana<<" znajduje siê na pozycji "<<i+1<<endl;

  system("pause");
  return 0;
}
```

21. Znajdowanie lidera w zbiorze (?):

```
int szukaj_lidera(int tab[],int n)
{
        int lider = tab[0], do_pary = 1;
        //wykreœlanie par o ró¿nych wartoœciach
        for(int i=1;i<n;i++)
        {
                if(do_pary > 0)
                {
                        if(tab[i]==lider) ++do_pary;
                        else --do_pary;
                }
                else
                {
                        ++do_pary;
                        lider = tab[i];
                }
        }
        //koniec wykreœlania
        if(do_pary==0) return -1; //zwrócenie -1 oznacza, ¿e zbiór nie posiada lidera
        int ile = 0; //zmienna zliczaj¹ca wyst¹pieñia potencjalnego lidera
        for(int i=0;i<n;i++)  //zliczamy wyst¹pienia lidera
        {
                if(tab[i]==lider) ++ile;
        }
        //sprawdzamy, czy potencjalny lider wystêpuje oczekiwan¹ iloœæ razy
        if(ile>n/2) return lider;
        return -1;
}

int main()
{
        int n, *tab, lider;
        cout<<"Ile liczb chcesz wczytaæ? ";
        cin>>n;
        tab = new int [n];
        for(int i=0;i<n;i++)
```

```
	{
		cin>>tab[i];
	}
	lider = szukaj_lidera(tab,n);
	if(lider==-1) cout<<"Zbiór nie posiada lidera"<<endl;
	else cout<<"Liderem zbioru jest "<<lider<<endl;
	delete [] tab;
	return 0;
}
```

22. Znajdowanie miejsca zerowego metoda polowienia przedzialow:

```
double f(double x)
{
	//rozpatrujemy wielomian f(x) = x^3 - 3x^2 + 2x - 6
	return x*(x*(x-3)+2)-6; //rozbijam schematem Hornera
}

double polowienie_przedzialow(double a, double b, double epsilon)
{
	if(f(a)==0.0)return a;
	if(f(b)==0.0)return b;

	double srodek;

	while(b-a > epsilon)
	{
		srodek = (a+b)/2;
		if(f(srodek) == 0.0) return srodek;  //jesli miejsce zerowe jest w srodku
		if(f(a)*f(srodek)<0) b = srodek;
		else a = srodek;
	}
	return (a+b)/2;
}

int main()
{
	double a = -10, b = 10, epsilon = 0.00001;

	cout<<"Znalezione miejsce zerowe wynosi: ";
	cout<<fixed<<setprecision(5)<<polowienie_przedzialow(a, b, epsilon);

	cin.get();
	return 0;
}
```

23. Calkowanie numeryczne - metoda trapezow (dla f(x)=x2+x+2):

```
double f(double x)
{
	//funkcja zawsze przyjmuje wartosci dodatnie
	//wiêc mo¿na pomin¹æ wartosæ bezwzglêdn¹
	return x*x+x+2;
}

double Pole(int a, int b, int n)
{
```

```
	double h = (b-a)/(double)n; //wysokosæ trapezów
	double S = 0.0; //zmienna bêdzie przechowywaæ sumê pól trapezów
	double podstawa_a = f(a), podstawa_b;

	for(int i=1;i<=n;i++)
	{
		podstawa_b = f(a+h*i);
		S += (podstawa_a+podstawa_b);
		podstawa_a = podstawa_b;
	}
	return S*0.5*h;
}

int main()
{
	int a, b, n;
	cout<<"Podaj przedzia³ [a, b]\na = ";
	cin>>a;
	cout<<"b = ";
	cin>>b;
	cout<<"Podaj liczbê trapezów: ";
	cin>>n;
	if(!(a<b)) cout<<"To nie jest przedzia³!";
	else cout<<"Pole figury wynosi: "<<fixed<<setprecision(2)<<Pole(a, b, n);

	cin.ignore();
	cin.get();
	return 0;
}
```

24. Calkowanie numeryczne - metoda prostokatow (dla f(x)=x2+x+2):

```
double f(double x)
{
  //funkcja zawsze przyjmuje wartosci dodatnie
  //wiêc mo¿na pomin¹æ wartosæ bezwzglêdn¹
  return x*x+x+2;
}

double Pole(int a, int b, int n)
{
	double x = (b-a)/(double)n; //pierwszy bok - ka¿dy prostok¹t ma taki sam
	double S = 0.0; //zmienna bêdzie przechowywaæ sumê pól trapezów
	double srodek = a+(b-a)/(2.0*n); //œrodek pierwszego boku

	for(int i=0;i<n;i++)
	{
		S+=f(srodek); //obliczenie wysokoœci prostok¹ta
	srodek+=x; //przejœcie do nastêpnego œrodka
	}
	return S*x;
}

int main()
{
	int a, b, n;
	cout<<"Podaj przedzia³ [a, b]\na = ";
```

```
	cin>>a;
	cout<<"b = ";
	cin>>b;
	cout<<"Podaj liczbê trapezów: ";
	cin>>n;

	if(!(a<b)) cout<<"To nie jest przedzia³!";
	else cout<<"Pole figury wynosi: "<<fixed<<setprecision(2)<<Pole(a, b, n);

	cin.ignore();
	cin.get();
	return 0;
}
```

25. Wyznaczanie pierwiastka arytmetycznego - metoda Newtona - Raphsona:

```
//ustawienie prezycji pierwiastka
const double eps = 0.000001;

double pierwiastek(double P, double eps)
{
	double a = 1., b = P;
  //dopóki nie otrzymamy ¿¹danej precyzji
  while(fabs(a-b)>=eps)
  {
    a = (a+b)/2.;
    b = P/a;
  }
	return a;
}

int main()
{
  double x;
  cout<<"Podaj liczbê, z której chcesz wyznaczyæ pierwiastek: ";
  cin>>x;
  cout<<fixed<<pierwiastek(x, eps);
  cin.ignore();
  cin.get();
  return 0;
}
```

26. Anagramy:

```
bool czy_anagram(char *a, char *b)
{
	//wyznaczenie liczby liter w slowie a i w slowie b
	int dl1 = strlen(a), dl2 = strlen(b);
	//jesli dlugosci wyrazów nie sa równe, to nie moga byc anagramy
	if(dl1!=dl2) return false;

	int licz[127]={};  //zerujemy liczniki
	for(int i=0;i<dl1;i++)
	{
		licz[a[i]]++;  //zliczamy litery pierwszego wyrazu
	}
	for(int i=0;i<dl1;i++)
```

```
        {
                licz[b[i]]--;  //odejmowanie wystapien liter
        }
        for(int i=0;i<127;i++)
        {
                if(licz[i]!=0)  //jesli ktorys licznik sie nie wyzerowal
                {
                        return false;  //to oznacza, ze s³owa nie sa anagramami
                }
        }
        return true; //jesli wszystkie liczniki sie wyzerowa³y, to mamy anagramy
}

int main()
{

        char a[101], b[101]; //dwa s³owa, maksymalnie 100 znaków
        cout<<"Podaj dwa wyrazy: ";
        cin>>a>>b;
        if(czy_anagram(a,b)) cout<<"Podane wyrazy sa anagramami\n";
        else cout<<"Podane wyrazy nie s¹ anagramami\n";

   cin.ignore();
   cin.get();
   return 0;
}
```

27. Liczby doskonale:

```
bool czy_doskonala(int n)
{
        int s = 1, p = sqrt(n);
        for(int i=2; i<=p; i++)
        {
                if(n%i == 0) s+= i + n/i;   //dodajemy do sumy dwa dzielniki
        }
        //jesli mamy do czynienia z liczb¹ kwadratow¹
        //to dwa razy dodalismy jej pierwiastek
        //wiêc musimy go raz odj¹æ
        if(n == p*p) s-=p;
        //jesli suma dzielników jest równa danej liczbie
        //do podana liczba jest doskona³a
        if(n == s) return 1;

        return 0;
}

int main()
{
        int n;
        cout<<"Podaj liczbê: ";
        cin>>n;
        if(czy_doskonala(n)) cout<<"Liczba "<<n<<" jest doskona³a";
        else cout<<"Liczba "<<n<<" nie jest doskona³a";

        cin.ignore();
        cin.get();
```

```
	return 0;
}
```

28. Nierownosc trojkata:

```
bool trojkat(int a, int b, int c)
{
	return a>0 && b>0 && c>0 && a+b>c && a+c>b && b+c>a;
}
```

29. Sprawdzenie, czy dwa odcinki przecinaj sie:

```
int iloczyn_wektorowy(pkt X, pkt Y, pkt Z)
{
	int x1 = Z.first - X.first, y1 = Z.second - X.second,
		x2 = Y.first - X.first, y2 = Y.second - X.second;
	return x1*y2 - x2*y1;
}
//sprawdzenie, czy punkt Z(koniec odcinka pierwszego)
//le¿y na odcinku XY
bool sprawdz(pkt X, pkt Y, pkt Z)
{
	return min(X.first, Y.first) <= Z.first && Z.first <= max(X.first, Y.first)
	&& min(X.second, Y.second) <= Z.second && Z.second <= max(X.second, Y.second);
}

bool czy_przecinaja(pkt A, pkt B, pkt C, pkt D)
{
	int v1 = iloczyn_wektorowy(C, D, A),
		v2 = iloczyn_wektorowy(C, D, B),
		v3 = iloczyn_wektorowy(A, B, C),
		v4 = iloczyn_wektorowy(A, B, D);

	//sprawdzenie czy siê przecinaj¹ - dla nieu¿ych liczb
	//if(v1*v2 < 0 && v3*v4 < 0) return 1;

	//sprawdzenie czy siê przecinaj¹ - dla wiêkszych liczb
	if((v1>0&&v2<0 || v1<0&&v2>0) && (v3>0&&v4<0 || v3<0&&v4>0)) return 1;

	//sprawdzenie, czy koniec odcinka le¿y na drugim
	if(v1 == 0 && sprawdz(C, D, A)) return 1;
	if(v2 == 0 && sprawdz(C, D, B)) return 1;
	if(v3 == 0 && sprawdz(A, B, C)) return 1;
	if(v4 == 0 && sprawdz(A, B, D)) return 1;

	//odcinki nie maj¹ punktów wspólnych
	return 0;
}
int main()
{
	pair<int, int> A, B, C, D;
	//definiujemy dwa odcinki skierowane A->B oraz C->D
	cout<<"Podaj wspólrzedne punktu A: ";
	cin>>A.first>>A.second;
	cout<<"Podaj wspólrzedne punktu B: ";
	cin>>B.first>>B.second;
	cout<<"Podaj wspólrzedne punktu C: ";
```

```
        cin>>C.first>>C.second;
        cout<<"Podaj wspólrzedne punktu D: ";
        cin>>D.first>>D.second;
        if(czy_przecinaja(A, B, C, D)) cout<<"Odcinki sie przecinaja\n";
        else cout<<"Odcinki sie nie przecinaja\n";

        cin.ignore();
        cin.get();
        return 0;
}
```