int t[3][2] dwuwymiarowa tablica liczb całkowitych
int (*t)[3] wskaźnik 3elementowej tablicy liczb całkowitych
int *t[3] tablica 3 wskaźników liczb całkowitych
int (*t)(int) wskaźnik funkcji o parametrze i wartościach typu int
int (*t[7])[2][3] tablica 7 wskaźników dwuwymiarowych tablic liczb rzeczywistych
int ((*t)[2][3])(int) wskaźnik dwuwymiarowej tablicy wskaźników funkcji o parametrze i wartościach typu int
typ ident; typ
typ tab[5]; tablica 5 elementów typu typ
typ tab[]; tablica typu typ, rozmiar nieokreślony
typ *ident; wskaźnik typu typ
typ *tab[]; tablica wskaźników typu typ
typ (*tab[]); tablica wskaźników typu typ
typ (*ptr)[]; wskaźnik tablicy elementów typu typ
typ *(*ptr)[]; wskaźnik tablicy wskaźników typu typ
typ &ref; referencja do typu typ;
typ (*ptr)(); wskaźnik funkcji o wartościach typ
typ (*(*ptr)())[]; wskaźnik funkcji o wartościach wskazujących tablicę elementów typu typ
const typ ident; stała typu typ
typ *const ptr; stały wskaźnik do typu typ
typ const *ptr; wskaźnik stałej typu typ
const typ *ptr; wskaźnik stałej typu typ
const * ident; wskaźnik stałej typu int

Napisz deklarację wskaźnika do funkcji zweracającej stały wskaźnik do wartości typu int przyjmującej jako argumenty wskaźnik do stałej typu int i stały wskaźnik do typu char

```
int * const (*fun1) ( const int  * , char * const );
int * const (*fun2) ( int const * , char * const );
```

ctype.h
wybrane funkcje

int isalnum(int c);
Zwraca wartość różną od zera, jeśli c jest znakiem alfanumerycznym, tj. literą (znaki od ‘A’ do ‘Z’ lub od ‘a’ do ‘z’) lub cyfrą (znaki od’0’ do ‘9’).

int isalpha(int c);
Zwraca wartość różną od zera, jeśli c jest literą (znaki od ‘A’ do ‘Z’ lub od ‘a’ do ‘z’).

int isascii(int c);
Zwraca wartość różną od zera, ghdy młodszy bajt parametru c ma wartość z przedziału od zera do 127.

int isdigit(int c);
Zwraca wartość różną od zera, jeśli c jest cyfrą (znaki od’0’ do ‘9’).

int isgraph(int c);
Zwraca wartość różną od zera, jeśli c jest kodem znaku drukarskiego (kody od 0x20 do 0xfe) i nie jest spacją (kod ox 20)

int islower(int c);
Zwraca wartość różną od zera, jeśli c zawiera kod małej litery (znaki od ‘a’ do ‘z’)

int isupper (int c);
Zwraca wartość różną od zera, jeśli c zawiera kod wielkiej litery (znaki od ‘A’ do ‘Z’)

string.h
wybrane funkcje

```
char *strcpy( char *strDestination, const char *strSource );
char *strcat( char *strDestination, const char *strSource );
char *strdup( const char *strSource );
char *strncpy( char *strDest, const char *strSource, size_t count );
char *strncat( char *strDest, const char *strSource, size_t count );
char *strchr( const char *string, int c );
```

Napisz implementację funkcji strcat() nie posługując się żadnymi funkcjami standardowymi z wyjątkiem malloc();

0	NUL	Zero :)
1	SQH	(Start Of Heading) - początek nagłówka = SOM
2	STX	(Start Of Text) - początek tekstu = EOA
3	ETX	(End Of Text) - koniec tekstu = EOM
4	EOT	(End Of Transmission) - koniec transmisji
5	ENQ	(Enquiry) - wywołanie stacji
6	ACK	(Acknowledge) - potwierdzenie
7	BEL	(Bell) - dzwonek
8	BS	(Back Space) - powrót o 1 pozycję
9	HT	(Horizontal Tab) - tabulacja pozioma
10	LF	(Line Feed) - przesuw o 1 wiersz
11	VT	(Vertical Tab) - tabulacja pionowa
12	FF	(Form Feed) - przesuw o 1 stronę
13	CR	(Carriage Return) - powrót karetki
14	SO	(Switch Output) - wyjście (przełączenie trwałe)
15	SI	(Switch Input) wejście (przełączenie powrotne)
16	DLE	(Data Link Escape) - pominięcie znaków sterujących
17	DC1	(Device Control 1) - sterowanie urządządzienia 1 / start transmisji = XON
18	DC2	(Device Control 2) - sterowanie urządzenia 2
19	DC3	(Device Control 3) - sterowanie urządzenia 3 / stop transmisji = XOFF
20	DC4	(Device Control 4) - sterowanie urządzenia 4
21	NAK	(Negative Acknowledge) - potwierdzenie negatywne (gdy wystąpił błąd)

22	SYN	(Sync) - synchronizacja
23	ETB	(End Transmission Blok) - koniec bloku
24	CAN	(Cancel) - anulowanie
25	EM	(End of Medium) - koniec nośnika (zapisu)
26	SUB	(Substitute) - zastąpienie
27	ESC	(Escape) - przełączenie
28	FS	(File Separator) - poprzedza dane alfanumeryczne
29	GS	(Group Separator) - poprzedza dane binarne
30	RS	(Record Separator) - separator rekordów
31	US	(Unit Separator) - separator pozycji
32	SP	(Space) - spacja (odstęp)
33	!	64
	@	96
	`	
34	"	65
	A	97
	a	
35	#	66
	B	98
	b	
36	$	67
	C	99
	c	
37	%	68
	D	100
	d	
38	&	69
	E	101
	e	
39	'	70
	F	102
	f	

40	(	71
	G	103
	g	
41	)	72
	H	104
	h	
42	*	73
	I	105
	i	
43	+	74
	J	106
	j	
44	,	75
	K	107
	k	
45	-	76
	L	108
	l	
46	.	77
	M	109
	m	
47	/	78
	N	110
	n	
48	0	79
	O	111
	o	
49	1	80
	P	112
	p	
40	2	81
	Q	113
	q	
51	3	82
	R	114
	r	
52	4	83
	S	115
	s	
53	5	84
	T	116
	t	

54	6	85
	U	117
	u	
55	7	86
	V	118
	v	
56	8	87
	W	119
	w	
57	9	88
	X	120
	x	
58	:	89
	Y	121
	y	
59	;	90
	Z	122
	z	
60	<	91
	[	123
	{	
61	=	92
	\	124
	|	
62	>	93
	]	125
	}	
63	?	94
	^	126
	~	
-	-	95
		127
	DEL	

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	█	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
141	ì	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Source: www.LookupTables.com

```
#include <iostream>
#include <stdio.h>

using namespace std;

class MY_FILE
{
    FILE *m_pFile;

    public:
                                MY_FILE():m_pFile(NULL)
                                          {}

                    MY_FILE(const char* nazwa, const char * tryb)
                                          {
                                m_pFile = fopen(nazwa,tryb);
                                          }

                                       ~MY_FILE()
                                          {
                                         zamknij();
                                          }

                   bool otworz(const char* nazwa, const char * tryb)
                                          {
                    return (m_pFile = fopen(nazwa,tryb)) ? true : false;
                                          }

                                      void zamknij()
                                          {
                                         if(m_pFile)
                                              {
                                                                        fclose(m_pFile);
                                                                           m_pFile=NULL;
                                              }
                                          }

                               MY_FILE& operator<<(const char* pStr)
                                          {
                                        fprintf(m_pFile,"%s",pStr);
                                               return *this;
                                          }

                                MY_FILE& operator<<(int l)
                                          {
                                           fprintf(m_pFile,"%d",l);
                                                return *this;
                                          }

                               MY_FILE& operator<<(char c)
                                          {
                                           fprintf(m_pFile,"%c",c);
                                                return *this;
                                          }

             friend ostream& operator<<(ostream &o, const MY_FILE &f)
                                          {
                                            char c;
                                  while((c=fgetc(f.m_pFile))!=EOF)
                                                                                 o<<c;
                                            return o;
                                          }
};


int main()
```

```
{
    MY_FILE plik;
    int liczba=-80;

    if( plik.otworz("abc.txt","w") == true )
    {
      plik << "Wartosc liczby: " << liczba  << '\n';
      plik.zamknij();
    }

    MY_FILE plik2("abc.txt","r");
    cout << "Zawartosc pliku:\n" << plik2;

    return 0;
}
```

```
#include <iostream>
#include <stdio.h>

using namespace std;

class CRandom
{
    unsigned                        m_iIlosc;
    int                                                         *m_piDane;
    int                                                 m_iDolny, m_iGorny;

    public:
           CRandom():m_iIlosc(0), m_piDane(0), m_iDolny(0), m_iGorny(0)
                                        {}

                               CRandom(unsigned iIlosc)
                                        {
                                     m_iIlosc = iIlosc;
                                m_piDane = new int [m_iIlosc];
                                        }

                           void zakres(int iDolny, int iGorny)
                                        {
                                     m_iDolny = iDolny;
                                     m_iGorny = iGorny;
                                        }

                                       ~CRandom()
                                        {
                                    delete [] m_piDane;
                                        }

                                      void losuj()
                                        {
                                        if(m_iIlosc)
                                          {
                                            srand( (unsigned) time( NULL ) );
                                for(unsigned i=0; i< m_iIlosc; ++i)

                  m_piDane[i] = (rand()%(m_iGorny-m_iDolny))+m_iDolny;
                                          }
                                        }

                friend ostream& operator<<(ostream &o, const CRandom &r)
                                        {
                             for(unsigned i=0; i<r.m_iIlosc; ++i)
                                                 o << r.m_piDane[i] << endl;
                                     return o;
                                        }
};

int main()
{
    CRandom liczba(20);

    liczba.zakres(-4,7);
    liczba.losuj();

    cout << liczba;       // wyswietla 20 losowych liczb

    return 0;
}
```

Wyjątki

```
try
{

   ryzykowne_instrukcje

}

catch (...)

{

   kod_obsługi_wyjątków

}
```

Co można „wyrzucać”

```
throw 12u;
throw "Wystapil blad";
throw CException("Wyjatek!", __FILE__, __LINE__);
throw 14.5;
```

Kolejność bloków catch

```
try
{
   // rzucamy wyjątek

   throw 90;

}
catch (float fLiczba)       { /* ... */ }
catch (int nLiczba)         { /* ... */ }
catch (double fLiczba)      { /* ... */ }
```

Przykład zastosowania wyjątku

```
#include <stdio.h>
#include <iostream>
#include <string>

using namespace std;

class CExcept
{
    string                          m_szInfo;
    int                             m_iExNum;
public:
    CExcept():m_szInfo(),m_iExNum(0){}
    CExcept(const CExcept& rCExc){ m_szInfo=rCExc.m_szInfo;
m_iExNum=rCExc.m_iExNum;}
    string GetInfo() const { return m_szInfo; }
    int    GetExNum() const { return m_iExNum; }
    void   SetInfo(const string& szInfo) { m_szInfo = szInfo; }
    void   SetNum(int iNum) { m_iExNum = iNum; }
};
```

```
double Dziel(double da, double db)
{
    if (db==0.0)
    {
                                    CExcept e;
                          e.SetInfo("Dzielenie przez zero");
                                    e.SetNum(1);
                                      throw e;
    }

    return da/db;
}


int main ()
{
    double dNum1, dNum2, dWynik;


start:
    cout << "Dzielenie liczb z wyrzuceniem wyjatku" << endl;
    cout << "Podaj dzielna:" << endl;
    cin >> dNum1;
    cout << "Podaj dzielna:" << endl;
    cin >> dNum2;

    try
    {
                            dWynik = Dziel(dNum1, dNum2);
    }
    catch( CExcept& e )
    {
             cout << "Nastapilo wyrzucenie wyjatku CExcept:" << endl;
      cout << "Numer bledu: " << e.GetExNum() << " opis: " << e.GetInfo() <<
endl;
                  cout << "Ponowienie obliczen [T/N]?" << endl;
                                    char c;
                                   cin >> c;
                                  if(c=='T')
                                    goto start;
                                      else
                                     return 0;
    }

    cout << "Wynik: " << dWynik << endl;

    return 0;
}
```

Przykład zastosowania wyjątku

```
#include <stdio.h>
#include <iostream>
#include <string>

using namespace std;

class CExcept
{
      string                          m_szInfo;
      int                                                          m_iExNum;

public:
      CExcept():m_szInfo(),m_iExNum(0)
```

```
      {
            cout << "Konstruktor bezparametrowy wyjatku CExcept" << endl;
      }
      ~CExcept()
      {
            cout << "Destruktor wyjatku CExcept" << endl;
      }

      CExcept(const CExcept& rCExc)
      {
            m_szInfo = rCExc.m_szInfo;
            m_iExNum = rCExc.m_iExNum;
            cout << "Konstruktor kopiujacy wyjatku CExcept" << endl;
      }
      string GetInfo() const { return m_szInfo; }
      int    GetExNum() const { return m_iExNum; }
      void   SetInfo(const string& szInfo) { m_szInfo = szInfo; }
      void   SetNum(int iNum) { m_iExNum = iNum; }
};

class KlasaA
{
    string m_szInfo;
public:
      KlasaA(const char* pC)
      {
            m_szInfo = pC;
            cout << "Konstruktor klasy KlasaA nazwa zmiennej:"
            << m_szInfo << endl;
      }

      ~KlasaA()
      {
            cout << "Destruktor klasy KlasaA nazwa zmiennej:"
            << m_szInfo << endl;
      }
};


class KlasaB
{
    string m_szInfo;
public:
      KlasaB(const char* pC)
      {
            m_szInfo = pC;
            cout << "Konstruktor klasy KlasaB nazwa zmiennej:"
            << m_szInfo << endl;
      }

      ~KlasaB()
      {
            cout << "Destruktor klasy KlasaB nazwa zmiennej:"
            << m_szInfo << endl;
      }
};

void funkcjaA(void)
{
    KlasaA ob1klA("ob1klA"), ob2klA("ob2klA");
    throw CExcept();
    cout << "koniec funkcji funkcjaA" << endl;
}

void funkcjaB(void)
{
```

```
    KlasaB ob1klB("ob1klB"), ob2klB("ob2klB");
    funkcjaA();
    cout << "koniec funkcji funkcjaB" << endl;
}
```

```
--------------------------------------------------------------------------
int main ()
{

    try
    {
                                    funkcjaB();
    }
    catch( CExcept& e )
    {
    }

    cout << "wyjscie z funkcji main" << endl;
    return 0;
}
```

ekran:

Konstruktor klasy KlasaB nazwa zmiennej:ob1klB
Konstruktor klasy KlasaB nazwa zmiennej:ob2klB
Konstruktor klasy KlasaA nazwa zmiennej:ob1klA
Konstruktor klasy KlasaA nazwa zmiennej:ob2klA
Konstruktor bezparametrowy wyjatku CExcept
Konstruktor kopiujacy wyjatku CExcept
Destruktor wyjatku CExcept
Destruktor klasy KlasaA nazwa zmiennej:ob2klA
Destruktor klasy KlasaA nazwa zmiennej:ob1klA
Destruktor klasy KlasaB nazwa zmiennej:ob2klB
Destruktor klasy KlasaB nazwa zmiennej:ob1klB
Destruktor wyjatku CExcept
wyjscie z funkcji main

Nieobsłużone wyjątki

```
int main ()
{
      funkcjaB();

      cout << "wyjscie z funkcji main" << endl;
      return 0;
}
```

ekran:

Konstruktor klasy KlasaB nazwa zmiennej:ob1klB
Konstruktor klasy KlasaB nazwa zmiennej:ob2klB
Konstruktor klasy KlasaA nazwa zmiennej:ob1klA
Konstruktor klasy KlasaA nazwa zmiennej:ob2klA
Konstruktor bezparametrowy wyjatku CExcept
Konstruktor kopiujacy wyjatku CExcept
Aborted

unexpected(), terminate(), abort()

Niezłapane wyjątki.

terminate(), abort()

unexpected_handler set_unexpected(unexpected_handler pfnFunction);
terminate_handler set_terminate(terminate_handler pfnFunction);

Przestrzeń nazw.

1. sposób

```
#include <iostream>
using namespace std;       // dyrektywa using namespace
int main ()
{
    cout << "wyjscie z funkcji main" << endl;
    return 0;
}
```

2. sposób

```
#include <iostream>
int main ()
{
    std::cout << "wyjscie z funkcji main" << std::endl;
    return 0;
}
```

3. sposób

```
#include <iostream>
using std::cout;
using std::endl;

int main ()
{
    cout << "wyjscie z funkcji main" << endl;
    return 0;
}
```

Oznaczenie globalnych zmiennych

// plik moj.h

```
#ifndef _MOJ__
#define _MOJ_

extern float liczbaFloat;
float funkcjaX(float);

#endif
```

// plik moj.cpp

```
#include "moj.h"

float liczbaFloat;
float funkcjaX(float f){ return f*20.0f;}
```

// plik main.cpp

```
#include <iostream>
#include "moj.h"

using std::cout;
using std::endl;

int main ()
{
    float liczbaFloat;
```

```
    liczbaFloat=27.5;       //odwolanie się do zmiennej lokalnej
    ::liczbaFloat = 89.4; //odwolanie się do zmiennej globalnej

    cout << "zmienna lokalna liczbaFloat " << liczbaFloat << endl;
    cout << "zmienna globalna liczbaFloat " << ::liczbaFloat << endl;
    cout << "funkcja globalna funkcjaX: " << funkcjaX(2.0f) << endl;
    cout << "funkcja globalna funkcjaX: " << ::funkcjaX(2.0f) << endl;

// funcje nie zadeklarowane w przestrzeni nazw są zawsze widoczne
// i posiadaja atrybut globalnosci

    return 0;
}
```

Własne przestrzenie nazw

// plik main.cpp zawiera przestrzen nazw oraz fukcje main

```
namespace MojaPrzestrzenNazw
{
    void fun1(void);
    float fun2(float);
    float const PI = 3.1415f;

    float liczba;
    class Klasa
    {
                                        int a;
                                       char c;
    public:
                                 void Set(int, char);
                                void Get(int&, char&);
                               class KlasaZagniezdzona
                                        {
                                       public:
                                       int Dummy();
                                          };
    };
}

void MojaPrzestrzenNazw::fun1(){}
float MojaPrzestrzenNazw::fun2(float x){ return x*PI;}
void MojaPrzestrzenNazw::Klasa::Set(int _a, char _c){ a=_a; c=_c;}
void MojaPrzestrzenNazw::Klasa::Get(int &_a, char &_c){ _a=a; _c=c;}
int MojaPrzestrzenNazw::Klasa::KlasaZagniezdzona::Dummy(){return -1;}

void main()
{
                            // tu uzywamy przestrzeni nazw
}
```

Deklaracja przestrzeni nazw powinna być umieszczona w pliku nagłówkowym (*.h), definicja składowych przestrzeni nazw w pliku źródłowym (*.cpp)

// plik moja.h zawiera deklarację przestrzeni nazw MojaPrzestrzenNazw

```
#ifndef _MOJA_PRZESTRZEN_NAZW_
#define _MOJA_PRZESTRZEN_NAZW_

namespace MojaPrzestrzenNazw
{
    void fun1(void);
    float fun2(float);
    extern float const PI;
    extern float liczba;

    class Klasa
    {
                                        int a;
                                       char c;
    public:
                                 void Set(int, char);
                                void Get(int&, char&);
                               class KlasaZagniezdzona
                                           {
                                          public:
                                          int Dummy();
                                             };
    };
}

#endif
```

// plik moja.cpp zawiera definicję składowych przestrzeni nazw MojaPrzestrzenNazw

```
#include "moja.h"

float MojaPrzestrzenNazw::liczba;
const float MojaPrzestrzenNazw::PI=3.1415f;
void MojaPrzestrzenNazw::fun1(){}
float MojaPrzestrzenNazw::fun2(float x){ return x*PI; }
void MojaPrzestrzenNazw::Klasa::Set(int _a, char _c){ a=_a; c=_c;}
void MojaPrzestrzenNazw::Klasa::Get(int &_a, char &_c){ _a=a; _c=c;}
int MojaPrzestrzenNazw::Klasa::KlasaZagniezdzona::Dummy(){return -1;}
```

// plik main.cpp

```
#include <iostream>
#include "moja.h"

using std::cout;
using std::endl;

int main ()
{
    MojaPrzestrzenNazw::liczba =4;
    cout << "liczba = " << MojaPrzestrzenNazw::liczba << endl;

    MojaPrzestrzenNazw::fun1();

    cout << "funkcja fun2: " <<  MojaPrzestrzenNazw::fun2(4.0f) << endl;

    MojaPrzestrzenNazw::Klasa obiekt;

    obiekt.Set(5,'r');
    int aa;
    char bb;
    obiekt.Get(aa,bb);
```

```
    cout << "aa: " <<  aa <<  endl << "bb: " << bb << endl;

    MojaPrzestrzenNazw::Klasa::KlasaZagniezdzona obiektKlasyZagniezdzonej;

    cout << "obiektKlasyZagniezdzonej.Dummy(): " <<
obiektKlasyZagniezdzonej.Dummy() << endl;

    return 0;
}
```

Ekran po wykonaniu programu:

liczba = 4
funkcja fun2: 12.566
aa: 5
bb: r
obiektKlasyZagniezdzonej.Dummy(): -1

Aliasy Przestrzeni nazw

```
#include <iostream>
#include "moja.h"

using std::cout;
using std::endl;

namespace mpn = MojaPrzestrzenNazw;

int main ()
{
    mpn::liczba =4;
    cout << "liczba = " << mpn::liczba << endl;

    MojaPrzestrzenNazw::fun1();

    cout << "funkcja fun2: " <<  mpn::fun2(4.0f) << endl;

    mpn::Klasa obiekt;

    obiekt.Set(5,'r');
    int aa;
    char bb;
    obiekt.Get(aa,bb);
    cout << "aa: " <<  aa <<  endl << "bb: " << bb << endl;


    mpn::Klasa::KlasaZagniezdzona obiektKlasyZagniezdzonej;

    cout << "obiektKlasyZagniezdzonej.Dummy(): " <<
obiektKlasyZagniezdzonej.Dummy() << endl;

    return 0;
}
```

Anonimowe przestrzenie nazw

// plik moja.h zawiera deklarację przestrzeni nazw MojaPrzestrzenNazw

```
#ifndef _MOJA_PRZESTRZEN_NAZW_
#define _MOJA_PRZESTRZEN_NAZW_
```

```
namespace MojaPrzestrzenNazw
{
    float mojaFunkcja(float);
}

#endif
```

// plik moja.cpp zawiera definicję składowych przestrzeni nazw MojaPrzestrzenNazw

```
#include "moja.h"

namespace                          // anonimowa przestrzen nazw
{
    float t;
    class Kl{};
    const float stala =12.12f;
}

float MojaPrzestrzenNazw::mojaFunkcja(float f)
{
    t = f-stala;
    return t*3.0;
}
```

// plik main.cpp

```
#include <iostream>
#include "moja.h"

using std::cout;
using std::endl;

using  MojaPrzestrzenNazw::mojaFunkcja;

int main ()
{
    cout << "mojaFunkcja: " << mojaFunkcja(15.0f) << endl;
    return 0;
}
```

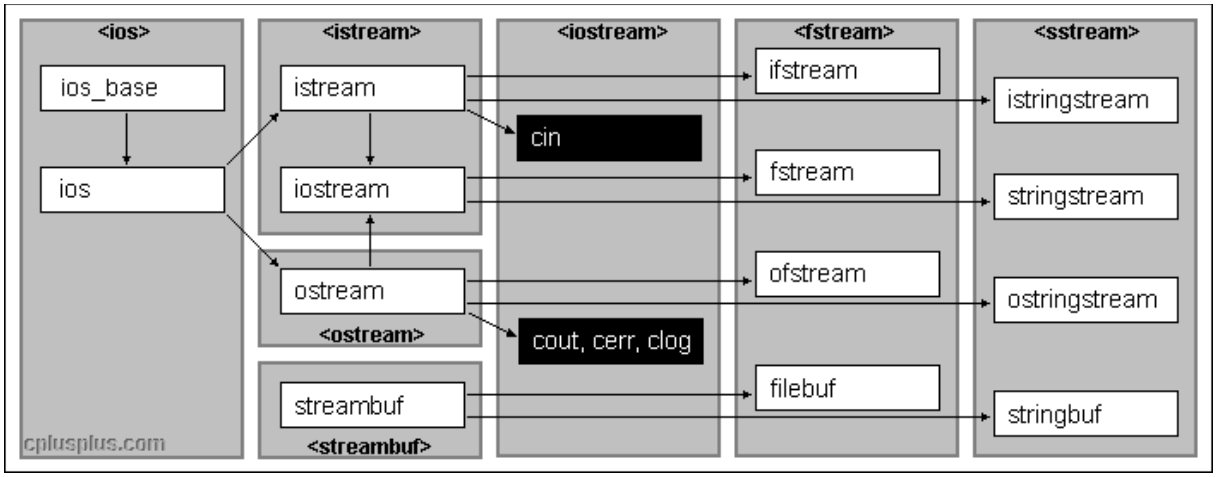
słowo kluczowe using a widoczność obiektów/funkcji w blokach

```
// niewidoczne std, MojaPrzestrzenNazw, TwojaPrzestrzenNazw
{
                                        using namespace std;
                                       // widoczne tylko std;
                                                {
                                    using namespace MojaPrzestrzenNazw;
                                     // widoczne std i MojaPrzestrzenNazw
                                                                                {
                          // widoczne std i MojaPrzestrzenNazw
                  }
        }

                                                {
                                    using namespace TwojaPrzestrzenNazw;
                                     // widoczne std i TwojaPrzestrzenNazw
                                                }

        // widoczne tylko std
}
```

Plikowe strumienie wejscia/wyjscia ifstream/ofstream




```
#include <string>
#include <iostream>
#include <fstream>
#include <iomanip>

using std::ostream;
using std::ofstream;
using std::ifstream;
using std::string;
using std::endl;
using std::cin;
using std::cout;
using std::setprecision;

class mojaKlasa
{
    int                             m_iLiczbaInt;
    string                            m_szNapis;
    float                           m_fLiczbaFloat;

public:
    mojaKlasa(int i, const string &str, float f):
    m_iLiczbaInt(i),
    m_szNapis(str),
    m_fLiczbaFloat(f)
    {}

    friend ostream& operator << (ostream& of, const mojaKlasa &kl)
    {
                          of << kl.m_iLiczbaInt << endl;
                            of << kl.m_szNapis << endl;
                         of << kl.m_fLiczbaFloat << endl;
                                   return of;
    }

    friend ofstream& operator << (ofstream& of, const mojaKlasa &kl)
    {
                          of << kl.m_iLiczbaInt << endl;
                            of << kl.m_szNapis << endl;
                         of << kl.m_fLiczbaFloat << endl;
                                   return of;
    }

    friend ifstream& operator >> (ifstream& ifs, mojaKlasa &kl)
    {
```

```
                                ifs >> kl.m_iLiczbaInt;
                                  ifs >> kl.m_szNapis;
                               ifs >> kl.m_fLiczbaFloat;
                                      return ifs;
    }
};

int main ()
{
    ifstream ifs;

    string szName;

    mojaKlasa kl1(-23,"hjklhjkl",0.234223777777f);
    mojaKlasa kl2(3,"rtrtrt",-0.456f);

//    cout << setprecision(8);
    cout << "Zawartosc kl1:" << endl << kl1;
    cout << "Zawartosc kl2:" << endl << kl2;

    cout << "Nazwe pliku zapisu:" << endl;
    cin >> szName;

    ofstream ofs(szName.c_str());

/*
ofstream ( );
explicit ofstream ( const char * filename, openmode mode = out );

ifstream ( );
explicit ifstream ( const char * filename, openmode mode = in );

fstream ( );
explicit fstream ( const char * filename, openmode mode = in | out );

ios_base::app (append) Seek to the end of the stream before each output
operation.
ios_base::ate  (at end) Seek to the end of the stream when opening.
ios_base::binary   Consider stream as binary rather than text.
ios_base::in           Allow input operations on a stream.
ios_base::out          Allow output operations on a stream.
ios_base::trunc   (truncate) Truncate file to zero when opening.
*/

    if(ofs.is_open())
    {
              cout << "Zapis zawartosci kl1 do " << szName << endl;
//                            ofs << setprecision(8);
                                  ofs << kl1;
      ofs.close();
    }
    else
                       cout << "nie mozna otworzyc pliku!";

    cout << "Nazwe pliku odczytu:" << endl;
    cin >> szName;
    ifs.open(szName.c_str());
    if(ifs.is_open())
    {
               cout << "Odczyt do kl2 z pliku " << szName << endl;
                                  ifs >> kl2;
      ifs.close();
    }
    else
                       cout << "nie mozna otworzyc pliku!";
```

```
    cout << "Zawartosc kl2:" << endl << kl2;

    return 0;
}
```

```
---------------------------------

    friend ofstream& operator << (ofstream& of, const mojaKlasa &kl)
    {
        of.write((const char*)&kl.m_iLiczbaInt, sizeof(kl.m_iLiczbaInt));
                    unsigned uStrLen = kl.m_szNapis.length();
                of.write((const char*)&uStrLen, sizeof(uStrLen));
                                    if(uStrLen)
                                                of.write(kl.m_szNapis.c_str(),
uStrLen);
      of.write((const char*)&kl.m_fLiczbaFloat, sizeof(kl.m_fLiczbaFloat));


                                     return of;
    }

---------------------------------

    friend ifstream& operator >> (ifstream& ifs, mojaKlasa &kl)
    {
           ifs.read((char*)&kl.m_iLiczbaInt, sizeof(kl.m_iLiczbaInt));
                                unsigned uStrLen;
                    ifs.read((char*)&uStrLen, sizeof(uStrLen));
                                    if(uStrLen)
                                         {
                               char *pSz = new char[uStrLen+1];
                                    ifs.read(pSz, uStrLen);
                                     pSz[uStrLen] = '\0';
                                      kl.m_szNapis = pSz;
                                         delete []pSz;
                                         }
         ifs.read((char*)&kl.m_fLiczbaFloat, sizeof(kl.m_fLiczbaFloat));

                                     return ifs;
    }

---------------------------------
```

Operatory:

- inkrementacji ++ dekrementacji – pre i post
- unarne
- rzutowania *typ*
- indeksowania []
- funkcyjne ()

Operatory, które mogą być przeciążane:

+	**-**	*****	**/**	**%**	**^**	**&**
|	**~**	**!**	**=**	**<**	**>**	**+=**
-=	***=**	**/=**	**%=**	**^=**	**&=**	**!=**
<<	**>>**	**>>=**	**<<=**	**==**	**!=**	**<=**
>=	**&&**	**||**	**++**	**--**	**->***	**,**
->	**[]**	**()**	**new**	**new[]**	**delete**	**delete[]**

Uwaga! Operatory =, ->, [], () muszą być metodami niestatycznymi.

```
#include <iostream>

using namespace std;

class mojaKlasa
{
    char m_szDane[125];
    int m_iLiczba;
    float m_faTablica[10][10];

public:
    friend ostream& operator << (ostream &o, const mojaKlasa &k)
    {
      return o << "m_iLiczba: " << k.m_iLiczba;
    }

    mojaKlasa():m_iLiczba(0){}
    mojaKlasa(int i):m_iLiczba(i){}
    ~mojaKlasa(){}

    // operator preinkrementacji
    mojaKlasa& mojaKlasa::operator++()
    {
      cout << "opertor preinkrementacji" << endl;
      ++m_iLiczba;
      return *this;
    }

    // operator postinkrementacji
    mojaKlasa mojaKlasa::operator++(int iDummy)
    {
      cout << "opertor postinkrementacji" << endl;
      ++m_iLiczba;
      return mojaKlasa(m_iLiczba-1);
    }

    mojaKlasa& operator=(const mojaKlasa &k)
    {
      cout << "opertor =" << endl;
      m_iLiczba = k.m_iLiczba;
        return *this;
    }
```

```
    // operator rzutowania
    operator int() const
    {
      cout << "opertor rzutowania" << endl;
      return m_iLiczba;
    }

    // operator unarny -
    mojaKlasa operator-() const
    {
      cout << "opertor unarny - " << endl;
      return mojaKlasa(-m_iLiczba);
    }

    char operator [](int i) const
    {
      cout << "indeksowanie, np. znak = obj[3]" << endl;
      return m_szDane[i];
    }

    char& operator [](int i)
    {
      cout << "indeksowanie, np. obj[3] = 'r'" << endl;
      return m_szDane[i];
    }

    float& operator()(int x, int y)
    {
      cout << "operator funkcyjny, np. kl(2,5) = 7.9" << endl;
      return m_faTablica[x][y];
    }

    float operator()(int x, int y) const
    {
      cout << "operator funkcyjny, np. fl = kl(2,5)" << endl;
      return m_faTablica[x][y];
    }

};
```

```
int main ()
{
/*
      skladowe klasy:

    char m_szDane[125];
    int m_iLiczba;
    float m_faTablica[10][10];
*/

    mojaKlasa kl(5), kl2;

    int i=0;
    char znak;
    float fl;

    kl++;
    ++kl;
    i = kl;

    cout << "obiekt kl: " <<  kl << endl;
    kl2 = -kl;
    cout << "obiekt kl2: " <<  kl2 << endl;

    kl[32] = 'x';
    znak = kl[32];
    cout << "zmienna znak: " << znak << endl;

    kl(2,5) = 19.45;
    fl = kl(2,5);

    cout << "zmienna fl: " << fl << endl;

    return 0;
}
```

ekran:

```
opertor postinkrementacji
opertor preinkrementacji
opertor rzutowania
obiekt kl: m_iLiczba: 7
opertor unarny -
opertor =
obiekt kl2: m_iLiczba: -7
indeksowanie, np. obj[3] = 'r'
indeksowanie, np. obj[3] = 'r'          // ?????????
zmienna znak: x
operator funkcyjny, np. kl(2,5) = 7.9
operator funkcyjny, np. kl(2,5) = 7.9          // ?????????
zmienna fl: 19.45
```