dr Gabriela Omiecińska

pokój 274

piątek: 11.30-12.15

sobota 13-13.45

Literatura

J. Gręboszz- Symfonia C++ standard.

S. Prata – Szkoła programowania, Język C++ (wydanie V).

S. Lippman, J. Lajoie – Podstawy języka C++.

Plan wykładu

- Typy danych, deklaracje zmiennych.
- Instrukcje.
- Wskaźniki.
- Tablice (klasa string).
- Funkcje, przeciążanie nazw funkcji, wskaźniki do funkcji, rekurencja.
- Szablony funkcji.
- Struktury, unie.
- Operatory bitowe (klasa bitset).
- Przestrzenie nazw.
- Pliki.
- Listy, kolejki, drzewa.

2 kolokwia wykładowe

```
    /*
      prog.cpp
*/
   #include <iostream>
   using namespace std;
    int main()
    {
         int k;
         cin>>k;
         cout << "Witaj w C++!\n";
         return 0;
      } // koniec funkcji main()
```

poprzedni

następny

Dyrektywa #include

```
#include <iostream>
#include "c\\pliki\\nag.h"
```

Wszystkie klasy, funkcje, zmienne stanowiące standardowy składnik kompilatorów C++,
są umieszczane w przestrzeni nazw std.

```
std::cout
```

Dyrektywa używania wszystkich nazw z przestrzeni std.

```
using namespace std;
```

Typy danych

Typy proste:

- całościowe (całkowite, logiczne, znakowe),
- zmiennoprzecinkowe,
- wskaźnikowe,
- referencyjne,
- wyliczeniowe.

Typy złożone:

- tablice,
- struktury,
- unie,
- klasy.

Deklaracje

Ogólna postać deklaracji zmiennej ma postać:

zestaw_specyfikatorów zmienna;

lub

zestaw_specyfikatorów zmienna=inicjalizator;

Specyfikatory

W zestawie specyfikatorów mogą wystąpić:

1) specyfikator typu

- char, signed char, unsigned char,
- short = signed short , unsigned short ,
- int = signed int, unsigned int,
- long = signed long , unsigned long ,
- bool,
- float, double, long double,
- void,
- identyfikator struktury, unii, wyliczenia, klasy.

2) specyfikator niezmienności

const

3) specyfikator klasy pamięci

auto,

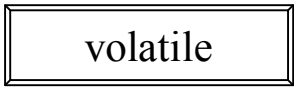
register,

static,

extern

4) specyfikator definicji typu

typedef



Literały

Literał jest to zapis reprezentujący daną, z którego to zapisu wynikają wszystkie atrybuty danej, w tym jej wartość i jej typ.

Literałami są m.in. liczby, znaki i łańcuchy.

Literały całkowite

Liczby całkowite mogą być zapisane w postaci dziesiętnej, ósemkowej lub szesnastkowej.

10 dziesiętnie

012 ósemkowo

0xA

0XA szesnastkowo

Stałe całkowite są domyślnie typu int.

liczba	wartość	typ
22	22	int
012L	10	long int
0xA2u	162	unsigned int
0uL	0	unsigned long int

specyfikatory

Deklaracje zmiennych całkowitych

```
int a;
unsigned int a=65, b=0101, c=0x41;
cout<<a<<" "<<b<<" "<<c<<endl;
```

```
65  65  65
```

```
cout<<oct<<b<<" "<<hex<<c<<" "<<dec<<a<<endl;
```

```
101  41 65
```

```
cout<<showbase<<oct<<b<<" "<<hex<<c<<" "<<dec<<a<<endl;
```

```
0101  0x41  65
```

specyfikatory

Plik nagłówkowy **climits** dostarcza informacje o zakresach typów całkowitych, definiuje pewne stałe.

zakres long: -2 147 483 648 ... 2 147 483 647

zakres int: -2 147 483 648 ... 2 147 483 647

zakres short: -32768 ... 32767

sizeof(char)<=sizeof(short)<=sizeof(int)<=sizeof(long)<=sizeof(long long)

Znaki sterujące:

'\a'	dzwonek
'\b'	backspace
'\n'	nowy wiersz
'\t'	tabulacja pozioma
'\\'	wypisanie znaku \
'\''	wypisanie apostrofu '
'\"'	wypisanie cudzysłowu "

specyfikatory

Deklaracje zmiennych znakowych

```
unsigned char znak;
zakres:  0...255


char znak;
znak='A';
znak='\101';        //  kod dużej litery A w systemie 8
znak='\x41';        //  kod dużej litery A w systemie 16
znak=65;            //  kod dużej litery A w systemie 10
char c;
c=znak+1;
cout<<"c="<<c;
```

c=B

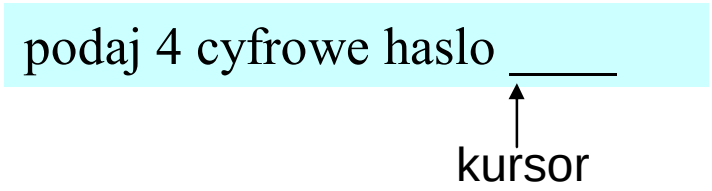
specyfikatory

```
    char znak;
    cin>>znak;
     cout<<znak<<" ma kod "<<(int)znak<<endl;

//cout<<" ma kod "<<static_cast<int>(znak)<<endl;
```

```
char znak;
cout<<"podaj znak: ";
cin.get(znak);
cout.put(znak);
cout<<" kod znaku: "<<(int)znak<<endl;
```

```
int haslo;
cout<<"podaj 4 cyfrowe haslo ____\b\b\b\b";
cin>>haslo;
```



Literały zmiennoprzecinkowe

13.4

2.3e-10, co oznacza $2.3*10^{-10}$

Stałe zmiennoprzecinkowe są typu double.

specyfikatory

Liczba rzeczywista z przyrostkiem f lub F jest typu float.

```
2.33f
```

Liczba zmiennoprzecinkowa z przyrostkiem l lub L jest traktowana jako long double np.

```
2.23L
```

specyfikatory

Zakresy stałych zmiennoprzecinkowych

typ float

zakres: -3.4e38 3.4e38 6 cyfr znaczących

typ double

zakres: -1.79e308 1.79e308 15 cyfr znaczących

najmniejsza wartość typu double: 2.22e-307

typ long double

zakres: - 3.4e4932 3.4e4932 18 cyfr znaczących

sizeof(float)<=sizeof(double)<=sizeof(long double)

Plik nagłówkowy **cfloat** dostarcza szczegółowych informacji o zakresach typów zmiennoprzecinkowych, liczbie cyfr znaczących, definiuje pewne stałe.

specyfikatory

```
int main()
{
	float a, b, c;
	b=2.e20;
	a=b+1.0f;
	c=a-b;
	cout<<"c="<<c<<endl;
	return 0;
}
```

specyfikatory

Specyfikator niezmienności const

```
const unsigned int CYFRA=4;
const int ZERO=0, JEDEN=1;
```

specyfikatory

Dyrektywa definiująca

#define NAZWA ciąg znaków

```
#define PI  3.14159
#define ALARM  '\a'
#define KOMUNIKAT  "prosze uwazac"

double r=5, pole;
pole=PI*r*r;
cout<<ALARM<<"pole="<<pole<<endl;

cout<<KOMUNIKAT<<endl;
```

Specyfikator definicji typu

Deklaracja poprzedzona specyfikatorem **typedef** wprowadza nową nazwę istniejącego typu (synonim).

```
typedef unsigned long UL;
UL tab[10];
UL xx;

typedef  long time_t;
```

specyfikatory

Instrukcje

- instrukcje proste: deklaracyjne, wyrażeniowe,
- instrukcje warunkowe: skrócona, pełna, wielokrotnego wyboru,
- instrukcje pętli: while, do...while, for,

Każda instrukcja zakończona jest średnikiem.

Operatory logiczne:

!	not	negacja
&&	and	koniunkcja logiczna, priorytet operatora koniunkcji jest wyższy niż operatora alternatywy logicznej
||	or	alternatywa

Operatory relacji:

==	czy równe
!=	czy różne
<	
<=	
>	
>=	

Literały logiczne
Deklaracje zmiennych typu logicznego

Słowa kluczowe **true** i **false** są literałami typu logicznego.

```
bool p1, p2;
p1=true;
p2=false;
cout<<"p1="<<p1<<"\tp2="<<p2<<endl;
```

```
p1=1    p2=0
```

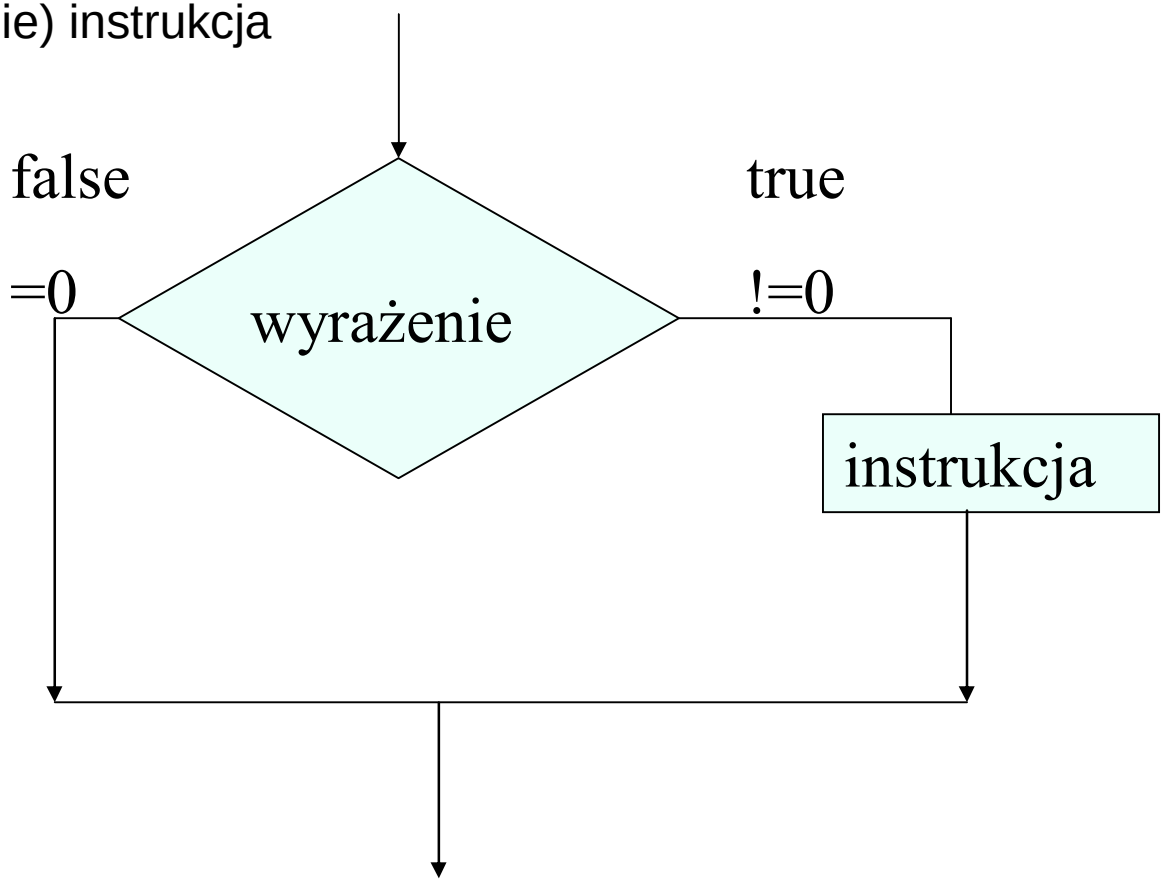
```
cout<<boolalpha<<"p1="<<p1<<"\tp2="<<p2<<endl;
```

```
p1=true    p2=false
```

typy

Instrukcja warunkowa

if (wyrażenie) instrukcja



```
float x, y;
cin>>x;
if(x>5 && x<10) {
                        y=sin(x);
                        cout<<"x="<<x<<"  y="<<y<<endl;
                       }
```

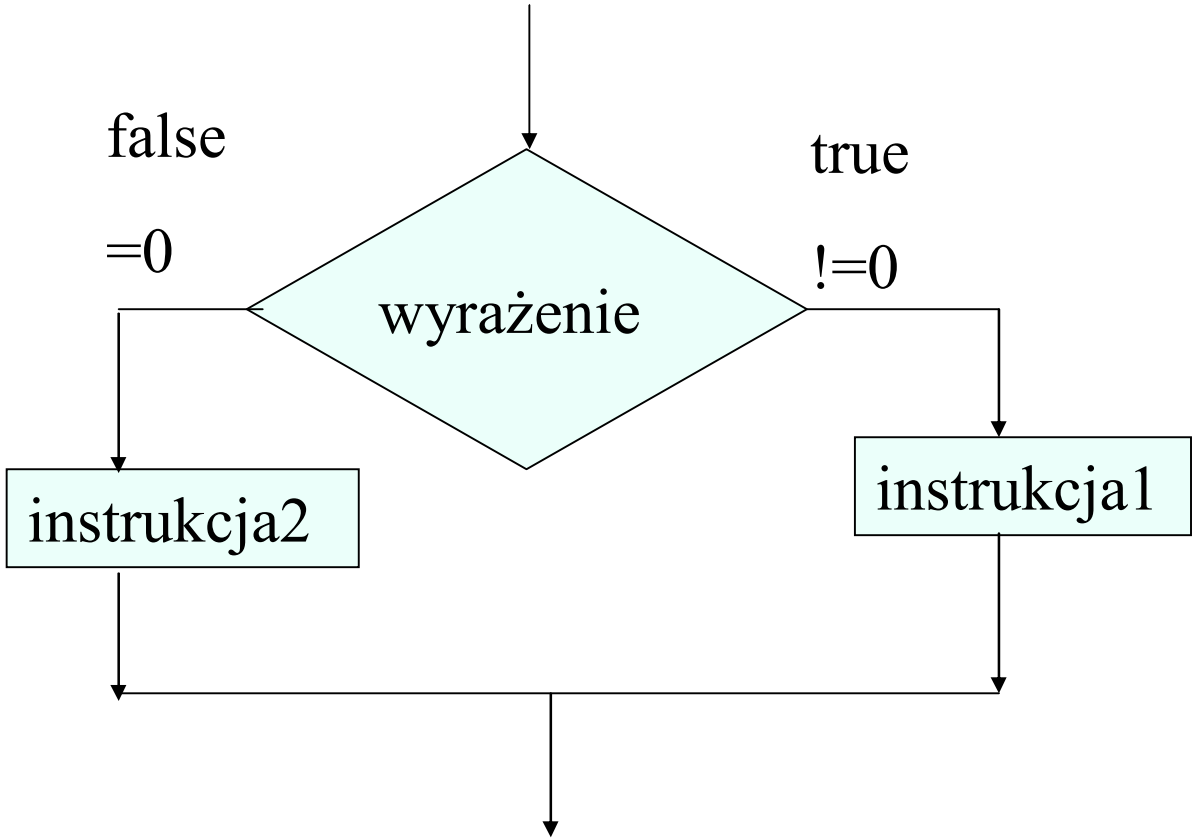
Wyrażeń relacyjnych można używać także w odniesieniu do znaków.

```
    char z1, z2;
    cin>>z1>>z2;
    if(z1>z2) cout<<"znak z1 ma kod ASCI wiekszy niż z2";
```

or ||

if (wyrażenie) instrukcja1

else instrukcja2



y(x)= $\begin{cases} \sin(\ \pi * x\ ) + 1 & \text{dla } x \neq 0 \\ 0 & \text{dla } x = 0 \end{cases}$

```
if (x==0) y=0;   else y=sin(M_PI*x)+1;

if (x!=0) y=sin(M_PI*x)+1;   else y=0;

if (x) y=sin(M_PI*x)+1;    else y=0;
```

Zagnieżdżone instrukcje if.

```
char odp;
cin>>odp;
if(odp=='t' or odp=='T') cout<< "wybrales tak";
        else if(odp=='n' or odp=='N')  cout<<"wybrales nie";
                else   cout<<"podaj t,T,n,N";
```

or ||

Operator warunkowy ? :

wyrażenie1 ? wyrażenie2 : wyrażenie3

```
x=(y<0 ?  -y : y);

if(y<0)  x=-y ;  else x=y;
```

```
max=(a>b ? a : b);

if(a>b)  max=a;  else max=b;
```

```
max=(a>b? (a>c ? a : c) : (b>c ? b : c));

if(a>b) if(a>c) max=a; else max=c;
        else if(b>c)  max=b;  else max=c;
```

```
int main()
{
    const int M=6;
    int g, p;
    cout<<"podaj ile masz gosci: ";
    cin>>g;
    p=(g % M!=0) ? g/M+1 : g/M;
    cout<<"potrzeba "<<p<< ((p ==1)? " paczki ": " paczek " )<< "ciastek" <<end;
     return 0;
}
```

Instrukcja wielokrotnego wyboru

```
switch(wyrażenie typu całkowitego)
{
    case wyr_stałe_typu_cał_1: instrukcje_1
    case wyr_stałe_typu_cał_2: instrukcje_2
    ..............
    case wyr_stałe_typu_cał_n: instrukcje_n
    default: instrukcje
}
```

Instrukcja switch nie obsługuje zakresów, każde wyrażenie po case musi być pojedynczą wartością.

```
int w;
cin>> w;
switch(w) {
        case 1:
              cout<<"podales 1"<<endl;
              break;
        case 2:
               cout<<"podales 2"<<endl;
               break;
        default:  cout<<"?";
}
```

```
char z;
cin>>z;
switch( z) {
    case '1':
                funkcja1( );
                break;
    case '2':
                funkcja2( );
                break;
    case 'k':
    case 'K':
                koniec( );
    default:
                cout<<"nieznany klawisz";
}
```