Laboratorium 1:

Różnice pomiędzy C a C++

	Język C	Język C++
1.1.	`#include <stdio.h>`	`#include <iostream>`
1.2.	`#include <string.h>`	`#include <string>`
1.3.	`printf("%s", "ciag znakow");` `printf("liczba: %d", 2);`	`std::cout << "ciag znakow";` `std::cout << "liczba: " << 2;`
1.4.	`int liczba;` `scanf("%d", &liczba);`	`int liczba;` `std::cin >> liczba;`
1.5.	`char* ciagZnakow[200];` `fgets(ciagZnakow, 200, stdin);`	`std::string ciagZnakow;` `std::getline(std::cin,ciagZnakow);`
1.6.	`int main(void){return 0;}`	`int main(){}`
1.7.	`const char* ciagZnakow = "test";`	`std::string ciagZnakow = "test";`

1.8.	`const char* a = "a";`
	`char b[10];`
	`strcpy(b,a);`
		`std::string a = "a";`
		`std::string b = a;`

1.9.	`const char* a = "a", b = "b";`
	`char c[10];`
	`strcat(c,a);`
	`strcat(c,b);`
		`std::string a = "a", b = "b";`
		`std::string c = a+b;`

1.10.	`const char* a = "a", b = "b";`
	`if(strcmp(a,b)==0)`
	`  printf("takie same");`
		`std::string a = "a", b = "b";`
		`if(a==b)`
		`  std::cout << "takie same";`

1.11.	`int* bufor;`
	`int rozmiar;`
	`scanf("%d", &rozmiar);`
	`bufor = (int*)malloc(rozmiar+1);`
		`int* bufor;`
		`int rozmiar;`
		`std::cin >> rozmiar;`
		`bufor = new int[rozmiar];`

1.12	`free(bufor);`	`delete bufor;` `delete[] bufor;`

1.13	`int funkcjaA(int argument){};`
	`//int funkcjaA(float argument){};`
	`int funkcjaB(float argument){};`
		`int funkcjaA(int argument){};`
		`int funkcjaA(float argument){};`

1.14	`struct ksiazka{char[10] tytul};` `struct ksiazka podrecznik;`	`struct ksiazka{std::string tytul};` `ksiazka podrecznik;`

Nowości w C++

	Nazwa	Do czego służą	Przykłady

1.15	Przestrzenie nazw	Pozwalają uniknąć kolizji nazw zmiennych.	`int a;`
			`namespace mojaPrzestrzen{`
			`  int a;`
			`}`

1.16	Dyrektywa `using`	Pozwala łatwiej korzystać z przestrzeni nazw.	`std::cout << "tu musi być std";`
			`using namespace std;`
			`cout << "a tu już nie!";`

1.17	Referencje	Pozwalają skrócić nazwy długich zmiennych. Są łatwiejsze w użyciu niż	`int strasznieDlugaZmienna=10;`
			`int &krocej =`
			`strasznieDlugaZmienna;`
			`krocej=5;`

		wskaźniki.	
1.18	Typ `bool`	Przechowuje wartości typu prawda/fałsz.	`bool a = false;` `bool b = true;`

1.19	Klasy i obiekty	Pozwalają lepiej odwzorować świat rzeczywisty	`class osoba{`
			`  string imię;`
			`}`

1.20	Szablony i struktury danych	Redukują ilość kodu, gdy potrzebujemy wielu funkcji/klas, które zachowują się ponownie, ale przyjmują argumenty różnego typu	`template <typename typ>`
			`void funkcja(typ argument){`
			`  cout << argument`
			`  << ” zajmuje ”`
			`  << sizeof(typ);`
			`  << ” bajtow.”;`
			`}`
			
			`funkcja<int>(5);`
			`funkcja<string>(”tekst”);`