Laboratorium 1

1. Cel ćwiczenia
Celem ćwiczenia jest zapoznanie się z obsługą C.

Kompilator online C++
https://www.onlinegdb.com/online_c++_compiler

Algorytm to jednoznacznie określony sposób, w jaki program komputerowy realizuje jakąś elementarną czynność

Język programowania to forma zapisu instrukcji dla komputera i programów komputerowych, pośrednia między językiem naturalnym a kodem maszynowym.

Kompilator – program zamieniający kod źródłowy, napisany w jednym z języków programowania, na kod maszynowy w postaci oddzielnych modułów.

Linker łączy skompilowane moduły kodu i inne pliki w jeden plik wykonywalny, czyli program (w przypadku Windows – plik EXE).

Skoro kompilacja i linkowanie są przeprowadzane automatycznie, a programista musi jedynie wydać polecenie rozpoczęcia tego procesu, to dlaczego nie pójść dalej – niech komputer na bieżąco tłumaczy sobie program na swój kod maszynowy. Rzeczywiście, jest to możliwe – powstało nawet kilka języków programowania działających w ten sposób (tak zwanych języków interpretowanych, przykładem jest choćby PHP, służący do tworzenia stron internetowych). Jednakże ogromna większość programów jest nadal tworzona w „tradycyjny" sposób. Dlaczego? Cóż – jeżeli w programowaniu nie wiadomo, o co chodzi, to na pewno chodzi o wydajność2 ;)) Kompilacja i linkowanie trwa po prostu długo, od kilkudziesięciu sekund w przypadku niewielkich programów, do nawet kilkudziesięciu minut przy dużych. Lepiej zrobić to raz i używać szybkiej, gotowej aplikacji niż nie robić w ogóle i czekać dwie minuty na rozwinięcie menu podręcznego

C++
C++ jest teraz chyba najpopularniejszym językiem do zastosowań wszelakich. Powstało do niego bardzo wiele kompilatorów pod różne systemy operacyjne i dlatego jest uważany za najbardziej przenośny. Istnieje jednak druga strona medalu – mnogość tych narzędzi prowadzi do niewielkiego rozgardiaszu i pewnych trudności w wyborze któregoś z nich. Na szczęście sam język został w 1997 roku ostatecznie ustandaryzowany. O C++ nie mówi się zwykle, że jest łatwy – być może ze względu na dosyć skondensowaną składnię (na przykład odpowiednikiem pascalowych słów **begin** i **end** są po prostu nawiasy klamrowe { i }). To jednak dosyć powierzchowne przekonanie, a sam język jest spójny i logiczny. Jest on też chyba najbardziej elastyczny – niejako dopasowuje się do preferencji programisty.

Środowisko programistyczne (ang. *integrated development environment* – w skrócie IDE) to pakiet aplikacji ułatwiających tworzenie programów w danym języku programowania. Umożliwia najczęściej organizowanie plików z kodem w projekty, łatwą kompilację, czasem też wizualne

tworzenie okien dialogowych. Popularnie, środowisko programistyczne nazywa się po prostu kompilatorem (gdyż jest jego główną częścią).

https://sourceforge.net/projects/orwelldevcpp/
Bloodshed Dev-C++
Pakiet ten ma niewątpliwą zaletę – jest darmowy do wszelakich zastosowań, także komercyjnych. Niestety zdaje się, że na tym jego zalety się kończą :) Posiada wprawdzie całkiem wygodne IDE, ale nie może się równać z profesjonalnymi narzędziami: nie posiada na przykład możliwości edycji zasobów (ikon, kursorów itd.)

Projekt w środowiskach programistycznych to zbiór modułów kodu źródłowego i innych plików, które po kompilacji i linkowaniu stają się pojedynczym plikiem EXE, czyli programem.

Pliki nagłówkowe umożliwiają korzystanie z pewnych funkcji, technik, bibliotek itp. wszystkim programom, które dołączają je do swojego kodu źródłowego.

Procedura to wydzielony fragment kodu programu, którego zadaniem jest wykonywanie jakiejś czynności.

Funkcja zawiera kod, którego celem jest obliczenie i zwrócenie jakiejś wartości.

Zmienna (ang. *variable*) to miejsce w pamięci operacyjnej, przechowujące pojedynczą wartość określonego typu. Każda zmienna ma nazwę, dzięki której można się do niej odwoływać.

Stała to niezmienna wartość, której nadano nazwę celem łatwego jej odróżnienia od innych, często podobnych wartości, w kodzie programu.

Znak **=** nie wskazuje tu absolutnie na równość dwóch wyrażeń – jest to bowiem **operator przypisania**, którego używamy do ustawiania wartości zmiennych.

Operator to jeden lub kilka znaków (zazwyczaj niebędących literami), które mają specjalne znaczenie w języku programowania.
operator opis
+ dodawanie
- odejmowanie
* mnożenie
/ dzielenie
% reszta z dzielenia

Najczęściej stosowane typy zmiennych

Nazwa	**Typ**	**Zakres**
char	Znak	< -128,127>
int	Liczba całk.	< -32768,32767>
short int	Liczba całk. krótka	< -32768,32767>
long int	Liczba całk. długa	< -2mln,2mln>

float	Liczba zmiennopozycyjna	$< -3*10^{38},3*10^{38}>$
double	Liczba zmienn. podwójnej precyzji	$< -3*10^{308},3*10^{308}>$

n oznacza przejście do nowej linii
%d oznacza w jaki sposób ma zostać wypisana zmienna i (pierwszy przykład). **%d** oznacza pole.
Dozwolone pola:
%d - liczba całkowita ze znakiem w formacie dziesiętnym
%e - liczba zmiennoprzecinkowa w zapisie naukowym (1.2345e+3)
%f - liczba zmiennoprzecinkowa typu double
%c - liczba całkowita jest konwertowana na bajt o danej wartości
%s - łańcuch tekstowy

Kompilacja pod linuxem

Otwieramy notatnik i piszemy kod programu. Zapisujemy i pod konsolą wpisujemy najpierw

g++ -o nazwa_pliku_wykonywalnego nazwa_pliku_zrodlowego.c

np.
g++ -o test test.c

Później wpisujemy polecenie do wykonania programu:
./nazwa_pliku_wykonywalnego

np.
./test

2. Przykłady

Przyklad1

```
#include <stdio.h>
int main (void) {
int i;
float f;
char c;
i=1;
f=2.0;
c='a';
printf("Hello World!\n");
printf("To jest liczba calkowita: %d.\n", i);
printf("To jest liczba zmiennoprzecinkowa: %f.\n", f);
printf("To jest liczba znak (litera): %c.\n", c);
return 0;
}
```

Przyklad2

```
#include <stdio.h>
int main(void)
{
int a = 12;
int b = 34567;
float c = 0.012;
float d = 123.456789;
char e = 'M';
printf("%6d\n", a);
printf("%6d\n", b);
printf("%#o\n", a);
printf("%#x\n", a);
printf("c = %5.1f, d = %5.1f\n", c, d);
printf("%c\n", e);
return 0;
}
```

Przyklad3

```
#include <stdio.h>
int main()
{
int a, b;
printf("Podaj wspolrzedne punktu: ");
scanf("%d", &a);
a = -a;
printf("Po odbiciu symetrycznym: %d\n", a);
return 0;
}
```

Przyklad4

Napisać program deklarujący jedną zmienną typu całkowitego o nazwie num, nadający jej wartość 1000 i używający funkcji printf() do wypisania na ekranie wartości w postaci „1000 jest wartością zmiennej num”

```
#include <stdio.h>

int main(void){
/*komentarz */
int num;
num=1000;
/*printf - wypisuje, %d - zwraca liczbe */
printf("%d jest wartoscia numeryczna", num);
return 0;
}
```

Przyklad5

Napisać program wczytujące dwie liczby zmiennoprzecinkowe (typu float) i wypisujący na ekranie ich sumę.

```
#include <stdio.h>

int main(void){
float a,b;
printf("\nWprowadz dwie liczby:");
scanf("%f",&a);
scanf("%f",&b);
printf("Suma wynosi %f.",a+b);
return 0;
}
```

Przyklad6

Napisać program obliczający objętość prostopadłościanu. Program wymaga wprowadzenia wielkości kolejnych wymiarów.

```
#include <stdio.h>

int main(void){
int dlugosc, szerokosc, wysokosc;
printf("Wprowadz dlugosc:\n");
scanf("%d",&dlugosc);
printf("Wprowadz szerokosc:\n");
scanf("%d",&szerokosc);
printf("Wprowadz wysokosc:\n");
scanf("%d",&wysokosc);
printf("\nObjetosc wynosi %d.", dlugosc*szerokosc*wysokosc);
return 0;
}
```

Przyklad7

```
#include <stdio.h>
void PokazTekst()
{
printf("\n Umiem pisac funkcje.");
}
int main()
{
PokazTekst();
}
```

3. Program ćwiczenia

Zad1
Napisać program obliczający liczbę sekund w roku.

Zad2
Napisać program zawierający co najmniej dwie funkcje i wypisujący na ekranie „Idzie żołnierz polem, lasem”.

Zad3
Napisać program korzystający z funkcji **przelicz()**, który wymaga podania wielkości w dolarach i przelicza ją na funty. Należy użyć kursu wymiany 1,6 dolara za funt. Wyświetlić wynik konwersji.

Zad4
Grawitacja na Księżycu jest równa blisko 17 procentom grawitacji ziemskiej. Napisać program pozwalający na wprowadzenie własnej wagi i obliczający odpowiadającą jej wagę efektywną na Księżycu.

Zad5
W jednej filiżance mieści się 8 uncji. Napisać program dokonujący konwersji z uncji na filiżanki. Przy konwersji należy skorzystać z funkcji **uncje_na_filizanki()**, której argumentem jest liczba uncji, a która zwraca liczbę filiżanek.

Zad6
Napisać program wymagający wprowadzenia liczby całkowitej, a następnie stwierdzający, czy liczba jest parzysta, czy nieparzysta. Należy skorzystać z operatora reszty z dzielenia (użyć „if”).

Zad7
Napisać program, który dodaje lub odejmuje liczby całkowite. Najpierw użytkownik musi wybrać działanie, a następnie dwie liczby, na których zostanie ono wykonane (Użyć „if”, „else”).