C++

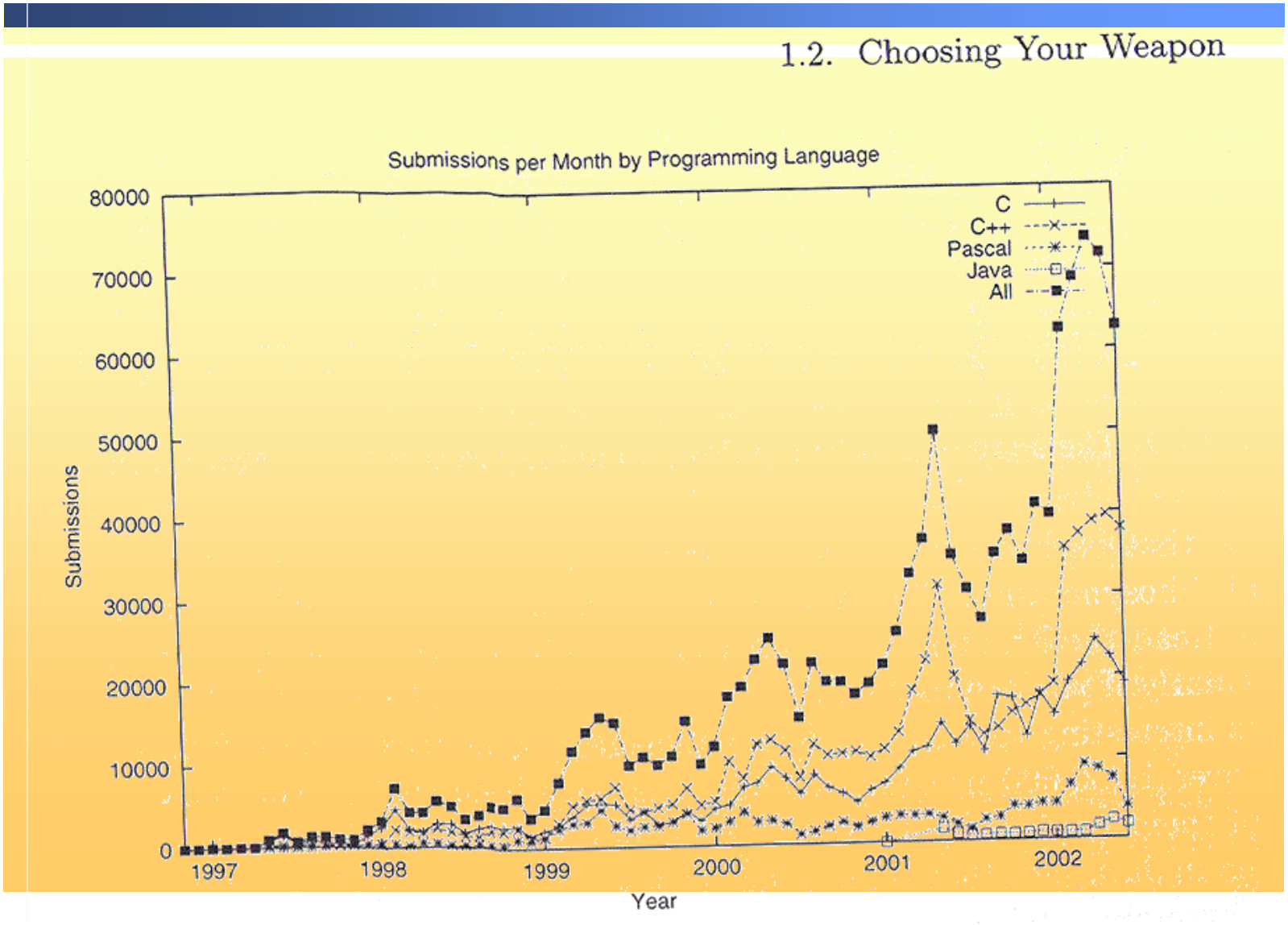
Pojava C-a

- pojava C-a potresla je informatički svijet;
- C++ je direktan potomak jezika C
- iz osnova je pristup programiranju i programerski način razmišljanja
- direktan rezultat potrebe za strukturiranim, efikasnim jezikom visoke razine koji može zamijeniti assembler kod izrade sustavnih programa
- označava početak modernog doba u računarskim jezicima
- moćan, efikasan, strukturiran jezik koji se razmjerno lako uči
- programerski jezik, odražava programerski pristup programiranju; zamislili, izgradili i razvili, izbrusili, isprobali i o njemu neprestano razmišljali su ljudi koji su ga stvarno i koristili – programeri, odražava njihov pristup programiranju

Povijesni pregled razvoja

- Bjarne Stroustrup (Danska)
- Programski jezik opće namjene
- C se prvi put javlja oko 1972.; C++ proširuje programski jezik C dodajući mu objektno-orijentirane osobine
- C++ razvijen sredinom osamdesetih u Bell Laboratories, Murray Hill, New Jersey
- Neke karakteristike:
  - Jednostavan za učenje
  - Omogućava modularno pisanje programa
  - Optimiziran programski kod, brzo se izvodi
  - Dobra kontrola strojnih resursa (memorija, procesor…)

Popularnost



C++ osnovna svojstva

- popularan **jezik (za kod koji se mora izvoditi brzo)**
- viši programski **jezik sa mogućnošću kontrole operacija na nižim razinama – omogućava izbjegavanje assemblera**
- **jednostavnije pisanje dobrih programa**
- za **sve aplikacijske domene**:
  - operacijski sustavi (u cjelini ili u ključnim dijelovima)
  - “device drivers” (direktna manipulacija sa hardverom)
  - kritični dijelovi koda (banke, trgovina, osiguranje, telekomunikacije – 0800, vojska)
  - grafička i korisnička sučelja (Windows)…
- “case-sensitive” **(razlikuje mala i velika slova) jezik**
- **može se kombinirati** sa drugim programskim jezicima
- objektno orijentiran **jezik**
- za profesionalce, za učenje i istraživanje

Objektno orijentirano programiranje

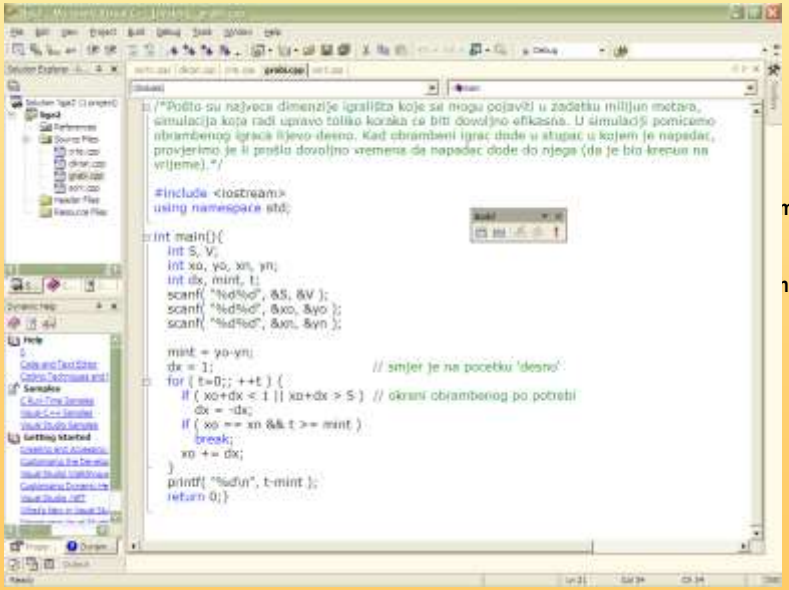
- identificiranje objekata, njihovo organiziranje u hijerarhije, dodavanje važnih svojstava (atributa) i funkcija (metoda) - ponašanja objektima kako bi se na objektu izvršilo željeni zadatak
- Objektno-orijentirani model programiranja nasuprot proceduralno-strukturiranom programiranju. Umjesto sa procedurama koje barataju sa objektima, radimo sa objektima koji objedinjavaju operacije i podatke.
- O podacima razmišljamo preko operacija koje možemo obavljati nad njima, odnosno objekt se sastoji od podataka koji opisuju objekt i operacija koje na njemu mogu biti primijenjene.
- Prije sekvencijalno, danas programiranje pogonjeno događajima (*event-driven*)

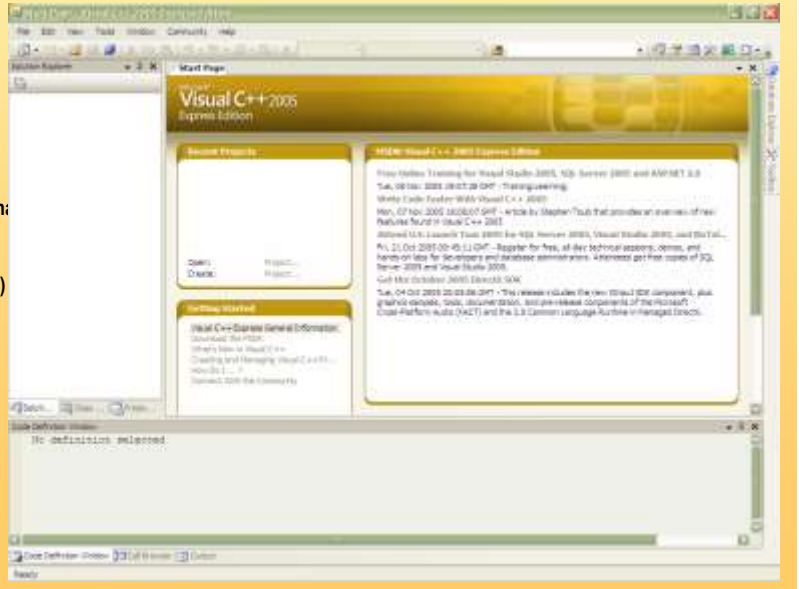
Usporedba s Javom

- Java je potpuno objektno orijentirani programski jezik
- Java izvorni kod se ne prevodi u strojni kod matičnog procesora (native code) nego u Java binarni kod (Java bytecode) kojeg Java virtualni stroj (Java Virtual Machine, JVM) interpretira i izvodi – prevedeni Java kod biti će izvodljiv na bilo kojem računalu s bilo kojim procesorom ili OS-om, ali sa JVM (općenito manje optimizirano)
- Java ima mehanizam za sakupljanje smeća (garbage collection) ugrađen u sustav i on se automatski pokreće
- Java podržava višenitnost (multithreading) – programa koji izvode dvije ili više operacija ”istovremeno”
- Java ne podržava preopterećenje operatora – funkcija operatora proširena na korisnički definirane podatke
- Kako je Javi primarna prenosivost koda, u Javi su npr. operacije s decimalnim brojevima lošije podržane

Pisanje izvornog koda

1. Unos izvornog koda (edit-time)
2. Prevođenje izvornog koda (compile-time)
3. Povezivanje u izvedbeni kod (link-time)
4. Izvođenje i testiranje programa (run-time)





Vrste datoteka

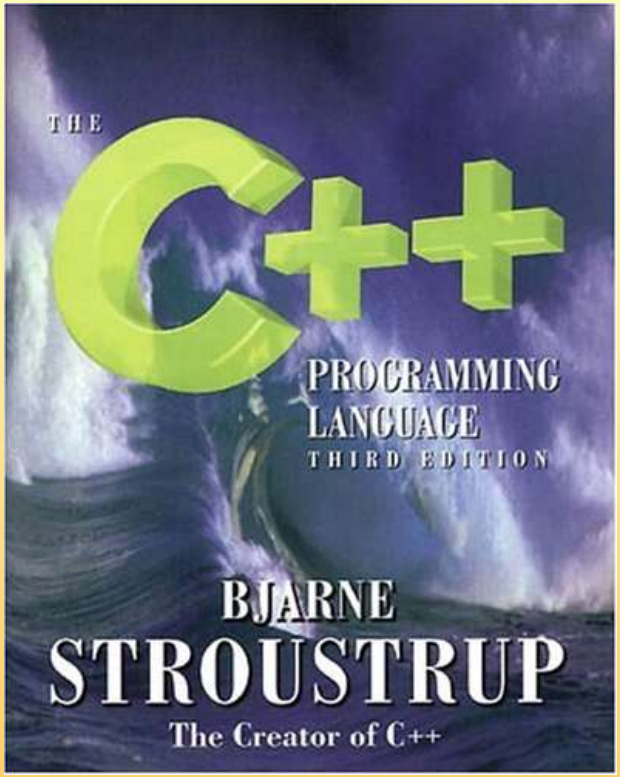
- **Datoteke zaglavlja:** .h
- **Datoteke izvornog koda:** .cpp, .cp **ili** .c
- **Datoteke objektnog koda (dobivaju se prevođenjem tekstualnog izvornog koda u objektnu datoteku strojnog koda):** .o **ili** .obj
- **Linker povezuje objektne datoteke.**
- **Izvršne datoteke**

Struktura programa

Izgled programa, pretprocesorske naredbe, funkcije, blokovi naredbi, ključne riječi…

Uvodni savjeti

- Kako pisati dobre programe u C++ jeziku?
  - znati što se želi reći
  - vježbati
  - imitirati dobro pisanje
- Pisanje dobrih programa zahtjeva
  - inteligenciju
  - volju i strpljenje
  - razum i iskustvo
  - analitičko razmišljanje



Struktura programa – Primjer 1.

```
int main() {
     return 0;
 }
```

- Svaki program napisan u C++ jeziku sastoji se od funkcija i mora imati točno jednu main() funkciju. main() je glavna funkcija i izvođenje svakog programa počinje naredbama koje se nalaze u njoj. Svaka funkcija ima zaglavlje i tijelo.
- int znači da će se kao rezultat izvođenja vratiti cijeli broj. U zagrade se mogu pisati argumenti (parametri) koji se iz OS-a prenose u main().
- { } označavaju blok naredbi.
- return 0; znači da je program uspješno okončan.
- Znakom ; završavamo gotovo svaku naredbu.

Struktura programa - Primjer 2.

```
#include <iostream>
using namespace std;


int main() {
     cout << "Pozdrav svima!!!" << endl;
     return 0;
}
```

- pretprocesorska naredba uključuje biblioteku iostream
  - u kojoj je tzv. ulazni i izlazni tok (input i output stream) **i funkcije koje omogućavaju čitanje podataka s tipkovnice i ispis podataka na zaslonu**

Lista datoteka zaglavlja (headers)

algorithm	cstdlib	iostream	set
bitset	cstring	iterator	sstream
cassert	ctime	limits	stack
cctype	deque	list	stdexcept
cfloat	fstream	locale	string
climits	functional	map	typeinfo
cmath	exception	memory	utility
cstdio	iomanip	queue	vector

Primjer 2. - nastavak

- Pretprocesorske naredbe počinju sa znakom #. `iostream` je primjer datoteke zaglavlja (header file) – sadrže deklaracije funkcija odgovarajućih biblioteka.
- `using` i `namespace` su ključne riječi kojima se “aktivira” određeno “područje imena”, a `std` je naziv imenika (biblioteke) gdje su obuhvaćene sve standardne funkcije.
- `cout` je ime izlaznog toka definiranog u biblioteci `iostream`, pridruženog zaslonu računala. Operatorom << podatak koji slijedi upućuje se na izlazni tok (zaslon računala).
- `endl` je konstanta u biblioteci `iostream` koja prebacuje ispis u novi redak. Operator koji ubacuje \n znak u stream.
- Ispred naredbe `return` može se dodati `char z; cin >> z;` ili `#include <conio.h>` pa `_getch()` koja vraća kod utipkanog znaka.

Pojmovi

- **Ključne (rezervirane) riječi** - specijalni dio teksta za kojeg prevodilac očekuje da će biti korišten na točno određeni način (`using, namespace, return`) i ne može se koristiti kao ime funkcije ili varijable
- **Identifikator** - bilo koje ime koje programer koristi za predstavljanje varijabli i funkcija. Mora početi sa slovom i mora sadržavati samo slova, brojeve ili znak _.
- **Varijable** – "promijenjivi podaci koji se koriste u programu". Moraju imati ime, tip i vrijednost, moraju biti deklarirane prije upotrebe. Tip varijable određuje interval vrijednosti koje varijabla može poprimiti i operacije koje na njoj mogu biti primijenjene

Ključne riječi jezika C++

__abstract [2]	__alignof	__asm	__assume
__based	__box [2]	__cdecl	__declspec
__delegate [2]	__event	__except	__fastcall
__finally	__forceinline	__gc [2]	__hook [3]
__identifier	__if_exists	__if_not_exists	__inline
__int8	__int16	__int32	__int64
__interface	__leave	__m64	__m128
__m128d	__m128i	__multiple_inheritance	__nogc [2]
__noop	__pin [2]	__property [2]	__raise
__sealed [2]	__single_inheritance	__stdcall	__super
__try_cast [2]	__try/__except, __try/__finally	__unhook [3]	__uuidof
__value [2]	__virtual_inheritance	__w64	bool
break	case	catch	char
class	const	const_cast	continue
default	delete	deprecated [1]	dllexport [1]
dllimport [1]	do	double	dynamic_cast
else	enum	explicit	extern
false	float	for	friend
goto	if	inline	int
long	mutable	naked [1]	namespace
new	noinline [1]	noreturn [1]	nothrow [1]
novtable [1]	operator	private	property [1]
protected	public	register	reinterpret_cast
return	selectany [1]	short	signed
sizeof	static	static_cast	struct
switch	template	this	thread [1]
throw	true	try	typedef
typeid	typename	union	unsigned
using declaration, using directive	uuid [1]	virtual	void
volatile	__wchar_t, wchar_t	while	

Struktura programa – Primjer 3.

```
#include <iostream>
using namespace std;

int main() {
        int a, b, c;

        cout << "Upiši prvi broj: ";
        cin >> a;

        cout << "Upiši drugi broj: ";
        cin >> b;

        c = a + b;
        cout << "Njihov zbroj je: " << c << endl;
     return 0;
}
```

Primjer 3. - nastavak

- **varijable a, b i c** identifikatorom tipa **određene su kao** cjelobrojne**, određen je raspon dozvoljenih vrijednosti, definirane operacije nad njima**
- **ulazni tok (*input stream*) `cin` definiran je u datoteci zaglavlja `iostream`. Služi za učitavanje podataka s tipkovnice (nakon *enter*).**
- `cin` ignorira **praznine, tab-ove i oznake za kraj retka**
- **C++ razlikuje velika i mala slova!**
- **Bez retka `using namespace std;` trebali bismo npr. pisati:**
  - std :: cout << “Kraj”;

Pisanje komentara

- **Svaki tekst u programu koji započinje sa // ili je omeđen sa /* i */ (mogu se protezati na nekoliko redaka bez ponavljanja znaka –** pogodno za (privremeno) isključivanje dijelova izvornog koda**).**
- **Gdje i što komentirati?**

- Na početku datoteke izvornog koda opisati sadržaj datoteke.
- Kod deklaracije varijabli, klasa (razreda) i objekata obrazložiti njihovo značenje i primjenu.
- Ispred funkcije dati opis što radi, koji su joj argumenti i što vraća kao rezultat. Eventualno dati opis algoritma koji se primjenjuje.
- Dati sažeti opis na mjestima u programu gdje nije potpuno očito što program (izvorni kod) radi.

Nije dozvoljeno “gnježđenje” (*nesting*) komentara.

Rastavljanje naredbi

- Umetanje praznina (općenito) doprinosi preglednosti koda.
- Naredbe u izvornom kodu mogu se protezati na nekoliko redaka (ima smisla kod vrlo dugačkih, nepreglednih naredbi). Obično se ograničava na 60-80 stupaca.
- Kod razdvajanja znakovnih nizova koristiti oznaku \ na kraju retka neposredno prije rastavljanja ili rastaviti na odvojene nizove.
- Moguće je više naredbi pisati u jednom retku, no ne preporuča se uvijek.

Podaci

Varijable i konstante

Deklaracija, pridruživanje vrijednosti, tipovi…

Naredbe za deklaraciju

- **Sve u programu korištene varijable potrebno je** deklarirati, odnosno navesti ime i tip varijable **(rezervirati prostor u memoriji),** npr:
  - int c;
- **U C-u moraju sve deklaracije biti na početku programa ili funkcije, prije prve naredbe.**
- **U C++ deklaracija varijabli može biti bilo gdje unutar programa, neposredno prije njihove prve primjene.**
- **Deklaracija varijable i pridruživanje vrijednosti (incijalizacija) mogu se obaviti odjednom i u istom retku koda, npr.**
  - int c = a + b;

Tipovi podataka - brojevi

- **Dva su osnovna tipa brojeva:**
  - **cijeli brojevi (short, int, long int ili long)**
  - **realni - decimalni brojevi (float, double, long double).**

- Oktalne konstante **pišu se tako da se ispred prve znamenke napiše broj 0 iza kojeg slijedi oktalni prikaz broja.**
- Heksadekadske konstante **započinju sa 0x ili 0X (slova za heksadekadske znamenke mogu biti velika ili mala).**
- **Kod prikaza realnog broja u** znanstvenoj notaciji **slovo e može biti veliko ili malo.**

Temeljni tipovi podataka (ugrađeni)

bool	true ili false	
char	Znakovni tip za ASCII vrijednosti	-128 do 127
short	Cjelobrojni tip	-32768 do 32767
int	Cjelobrojni tip	-2147483648 do 2147483647
long	Cjelobrojni tip	Kao int
_int8	Cjelobrojni tip	Odgovara tipu char
_int16	Cjelobrojni tip	Odgovara tipu short
_int32	Cjelobrojni tip	Odgovara tipu int
_int64	Cjelobrojni tip	-9223372036854775808 do 9223372036854775807
float	Realni brojevi	Do 7 dec. mj. ili 3.4E+/-38
double	Realni brojevi	Do 15 dec. mj. ili 1.7E+/-308
wchar_t	Široki znak (wide character)	Višebajtni znakovni tip

Dodjeljivanje tipa numeričkim konstantama

- Za određivanje tipa numeričkih konstanti moguće je primijeniti operatore dodjele tipa ili se koriste sufiksi
- npr. sufiksi U i L za cjelobrojne podatke mogu se međusobno kombinirati u bilo kojem redoslijedu – rezultat je unsigned long int
- Sufiksi se rijetko koriste

Sufiks	Primijenjen na	Pretvara u
L ili l	cjelobrojnu konstantu	long
L ili l	konstantu s decimalnom točkom	double
U ili u	cjelobrojnu konstantu	unsigned int
F ili f	konstantu s decimalnom točkom	float

Ugrađeni brojevni tipovi, duljine i rasponi, vrijednosti...

- **Broj** pi **i ostale značajnije matematičke konstante definirani su u biblioteci `cmath` (neki podaci su u climits i cfloat).**
- **U <u>climits</u> i <u>cfloat</u> (odnosno limits.h i float.h), su npr. `INT_MIN i INT_MAX`.**
- **Najveće i najmanje vrijednosti, mogu se dobiti pomoću npr. `LONG_MAX, FLT_MAX, DBL_MIN...`**
- **Provjera veličine pomoću `sizeof(int)` i `sizeof(float)`**
- **Svakoj varijabli može biti dodan** kvalifikator **prije tipa (npr. `unsigned` - bez predznaka):**
  - `unsigned int c = 12789;`
- **<u>Ne pridruživati negativne vrijednosti takvoj varijabli</u>**

Logički tip

- **Za prikaz podataka logičkog tipa (dva moguća stanja) postoji ugrađeni tip `bool` (obično zauzima 1 byte memorije).**
- **Pri ispisu logičkih tipova ili korištenju u aritmetičkim izrazima, logički tipovi se prema pravilima cjelobrojne promocije** pretvaraju u int (true u 1, false u 0).
- Vrijednosti različite od nule pretvaraju se u true, a jednake nuli u false.
- **Primjeri pridruživanja vrijednosti tipu bool:**
  - tvrdavoda = (Ca>40 && Mg>20);
  - kiselavoda = (pH<7);
  - dobarokus = true;
  - dobarokus = 15; // 1 (true) u dobarokus
  - cin>>boolalpha>>odgovor;

Znakovi (tip char)

- **Pišu se unutar jednostrukih znakova navodnika.**
  - `char SlovoA = 'a';`
  - `cout << 'b' << endl;`
- **Za znakove koji se ne mogu prikazati na zaslonu koriste se posebne (*escape*) sekvence.**
- **Znakovne konstante najčešće se koriste u znakovnim nizovima (`string`) za ispis tekstova.**

Konstante

- **Nepromijenjivi podaci**
- Literalne i simboličke.
- **Literalna je** vrijednost **u programu.**
- **Simbolička konstanta je konstanta predstavljena** imenom**.**
- **Uglavnom je bolje koristiti simboličke konstante.**

Simboličke konstante

- **Simboličke veličine čija se vrijednost tijekom izvođenja ne želi mijenjati**
- **Kvalifikatorom `const` prevoditelju se daje na znanje da varijabla mora biti** nepromjenjiva. **Piše se npr.:**
  - const double pi = 3.14159265359;

    //Konstante definirane pomoću deklaracije `const` dohvatljive su i iz debuggera.

  Ili pretprocesorskom naredbom:

  - #define PI 3.14159265359
- **Npr. u redu je i:**

```
float x;
cin >> x;
const float jp = x;
```

Operatori

Operator pridruživanja

- Za promjenu vrijednosti nekog objekta. Najčešći je znak jednakosti (=). Ako su objekti različitih tipova, vrijednost se svodi na tip objekta prema definiranim pravilima pretvorbe.
- Objekti koji se smiju nalaziti s lijeve strane znaka pridruživanja nazivaju se **lijeve vrijednosti (lvalues).**
- Samo promjenljivim lvrijednostima može se pridružiti nova vrijednost.
- Dozvoljeno je više operatora pridruživanja u istoj naredbi, npr.
- a = b = c = 0;

Ostali operatori pridruživanja

- **=, +=, -=, *=, /=, %=, >>=, <<=, ^=, &=, |=**
- **tzv. složeni operatori pridruživanja (update assignment)** – omogućuju kraći zapis naredbi. Npr.:
  - a += 5; isto što i a = a + 5;
- Operator pridruživanja ima niži prioritet od svih aritmetičkih i bitovnih operatora.

Aritmetički operatori

oznaka	operacija
+x	Unarni plus
-x	Unarni minus
x++	Uvećaj nakon
++x	Uvećaj prije
x--	Umanji nakon
--x	Umanji prije
x+y	Zbrajanje
x-y	Oduzimanje
x*y	Množenje
x/y	Dijeljenje
x%y	Modulo

Primjer
(uvećanje/umanjenje za jedan):

```
int i = 0;
++i;

cout << (i++) << endl;
cout << i << endl;
```

Pretvorba tipova

- **U operaciji dijeljenja** ne bi trebale biti uključene dvije cjelobrojne varijable, jer će decimalni dio biti odsječen**.**
- Ako je varijabla na lijevoj strani znaka pridruživanja realnog tipa, radi sigurnosti preporuča se koristiti decimalnu točku za sve konstante na desnoj strani znaka pridruživanja.
- Kada je na lijevoj strani naredbe pridruživanja varijabla cjelobrojnog tipa, treba se osigurati da aritmetički izraz na desnoj strani daje cjelobrojnu vrijednost.

Logički operatori

!x	**logička negacija**
x && y	**logički i**
x || y	**logički ili**

Napraviti tablice za sve moguće vrijednosti.

- Evaluacija završava čim se odredi logička vrijednost izraza.
- Logički operatori i operacije s njima uglavnom se koriste u naredbama za grananje.
- Ako prevoditelj ne podržava taj tip, može se napisati enum bool {false, true}

Relacijski operatori (operatori uspoređivanja)

- Za usporedbu dva broja.
- Rezultat je tipa bool.
- **Operatori su: <, <=, >, >=, ==, !=.**
- Koriste se pretežito u naredbama za grananje toka programa.

Bitovni operatori

- Velika prednost jezika C++ je što omogućava izravne operacije na pojedinim bitovima cjelobrojnih podataka (char, short, int, long).
- **Bitovni operatori su:**
  - **~i (komplement)**
  - **i & j (binarni i)**
  - **i | j (binarni ili)**
  - **i ^ j (isključivi ili)**
  - **i << n (pomakni ulijevo)**
  - **i >> n (pomakni udesno).**
- Koristi se za maskiranje bitova, za postavljanje bitova, zastavica ili opcija.

Operator sizeof

- Unarni operator koji kao rezultat daje broj bajtova što ih operand određenog tipa zauzima u memoriji.
  - cout << sizeof(f) << endl;

Operator razdvajanja

- Operator razdvajanja **,** koristi se za razdvajanje izraza u naredbama. Izrazi razdvojeni zarezom se izvode postepeno s lijeva na desno.
- Nakon naredbe
  - i = 10, i + 5; varijabli i će biti pridružena vrijednost 15.
- Operator razdvajanja ima najniži prioritet.
- Često se koristi u sprezi s uvjetnim operatorom, te za razdvajanje više izraza u parametrima `for` petlje.

Alternativne oznake operatora

{	<%	??<
}	%>	??>
[	<:	??(
]	:>	??)
#	%:	??=
##	%:%:	
&&	and	
||	or	
!	not	
&	bitand	
|	bitor	??!
^	xor	??’
~	compl	??-
!=	not_eq	
&=	and_eq	
|=	or_eq	
^=	xor_eq	

Alternativne oznake i nizovi od tri znaka (trigraf).

Alternativne oznake operatora

and	bitand	compl
not_eq	or_eq	xor_eq
and_eq	bitor	not
or	xor	

Hijerarhija i redoslijed izvođenja operatora

- U matematici postoji utvrđena hijerarhija operacija prema kojoj neke operacije imaju prednost pred drugima. Podrazumijevani slijed operacija je slijeva nadesno, ali ako se dvije operacije različitog prioriteta nađu jedna do druge, prvo se izvodi operacija s višim prioritetom.
- Okruglim zagradama se može zaobići ugrađena pravila prioriteta izračunavanja izraza, budući da one imaju viši prioritet od svih operatora.
- Često je zgodno zagrade koristiti i radi čitljivosti koda, čak kada one nisu neophodne.

Hijerarhija operatora – prioriteti (dio)

uvećaj nakon (++)

umanji nakon (--)

uvećaj prije (++)

umanji prije (--)

unarni operatori (+ - ! ~)

množenja (* / %)

zbrajanja (+ -)

bitovni pomaci (<< >>)

poredbeni operatori (< > <= >=)

operatori jednakosti (== !=)

bitovni i (&)

bitovno isključivo ili (^)

bitovni ili (|)

logički i (&&)

logički ili (||)

pridruživanja

razdvajanje (,)

C and C++ operator precedence

Simbol	Ime ili značenje		
*Highest Precedence level***********			
++	Post-increment	--	Post-decrement
++	Pre-increment	--	Pre-decrement
!	Logical NOT	~	Bitwise NOT
-	Unary minus	+	Unary plus
*****	**Multiply**	**/**	**Divide**
%	**Remainder**		
+	**Add**	**-**	**Subtract**
<<	Left shift	>>	Right shift
<	**Less than**	**<=**	**Less than or equal to**
>	**Greater than**	**>=**	**Greater than or equal to**
==	Equal	!=	Not equal
&	Bitwise AND		
^	Bitwise exclusive OR	|	Bitwise OR
&&	Logical AND	||	Logical OR
=	Assignment		
*=, /=, %=, +=, -=, <<=, >>=, &=, ^=, |=			Compound assignment
*Lowest Precedence level**********			

Matematičke funkcije

- **#include <cmath>**
- **sin(x) – argument mora biti u radijanima, exp(x), log(x), sqrt(x), pow(x,y), log10(y), fabs(x), fmod(x,y), floor(x), ceil(x)...**
- **#include <cstdlib>**
- **abs(x) – argument je cjelobrojan**

•C++ nema ugrađen operator za potenciranje, već se može koristiti funkcija pow().

•Ako su operandi različitih tipova, oni se prije operacije svode na zajednički tip (obično složeniji).

Oblikovanje izlaza

Oblikovanje izlaza

- **Prekidanje nizovne konstante u izvornom kodu:**

```
cout << "Spajamo niz \
sa ovim." << endl << endl;
```

ili

```
cout << "Spajamo " "niz "
"sa " "ovim.\n";
```

- **Kreiranje nove linije: korištenjem `\n` u nizovnoj konstanti ili `endl` manipulatora. Primjer:**
  - `cout << "\nWe can\njump\n\ntwo lines.";`
  - `\n` - linefeed simbol u nizovnoj konstanti (znakovna escape sekvenca)

Značenja nekih *escape* sekvenci

\0	Null character	Završava znakovni niz
\a	Alert/bell	Generira upozorenje, zvučni signal
\b	Backspace	Pomiče aktivnu poziciju unatrag
\f	Form feed	Pomiče aktivnu poziciju na početak nove str.
\n	New line	Inicijalna pozicija slijedeće linije
\r	Carriage return	Inicijalna pozicija tekuće linije – poč. retka
\t	Horizontal tab	Pozicija slijedećeg vodoravnog tab-a
\v	Vertical tab	Pozicija slijedećeg okomitog tab-a
\\	Backslash	Ispisuje backslash znak
\’	Single quote	Ispisuje jednostruki navodnik
\”	Double quote	Ispisuje dvostruki navodnik
\%	Percent	Ispisuje znak %

Primjeri formatiranja ispisa

Ubacivanjem I/O parametriziranih manipulatora deklariranih u iomanip.

```
#include <iomanip>

cout << "PI=[[" << setw(15) << PI << "]]" << endl;
cout << "PI=[[" << setprecision(2) << PI << "]]" << endl;
cout << "PI=[[" << setw(20) << setfill('*') << PI << "]]" << endl;
cout << "PI=[[" << setiosflags(ios::left) << setw(20) << PI << "]]" << endl;
cout << setprecision(4);
cout << "PI=[[" << setiosflags(ios::scientific) << PI << "]]" << endl;
cout << "PI=[[" << setiosflags(ios::left | ios::scientific) << setw(20) << PI
   << "]]" << endl;
```

setfill postavlja znak za popunjavanje, setprecision postavlja preciznost decimalnog broja, setw postavlja širinu polja, setiosflags postavlja format zastavice, resetiosflags vraća na podrazumijevanu zastavicu, setbase postavlja bazu brojeva koji se ispisuju. boolalpha ispisuje bool tip kao true ili false.

Rješenja (nastavak)

PI=[[3.14159]]

PI=[[3.14]]

PI=[[**********3.14]]

PI=[[3.14**********]]

PI=[[3.1416e+00]]

PI=[[3.1416e+00**********]]

Provjeriti:

cout<<“Unijeti da ili ne (0 za ne): “<<endl<<cin>>odgovor;

Neke zastavice klase ios i njihova upotreba

left	lijevo poravnan izlaz
right	desno poravnan izlaz
dec	broj ispisan kao ekvivalent decimalnog broja
hex	broj ispisan kao ekvivalent heksadecimalne vrijednosti
oct	broj ispisan kao ekvivalent oktalne vrijednosti
fixed	specificira neznanstveni zapis za realni broj
scientific	određuje znanstveni zapis za realni broj
showpoint	pokazuje decimalnu točku
showbase	pokazuje bazu za numeričke vrijednosti
uppercase	ispisuje velika štampana slova

Input & output streams, čitanje podataka s tipkovnice

- Streams – nizovi bajtova u određenoj sekvenci koji teku od uređaja do uređaja.
- Objekti su područja u memoriji. Objekt korišten sa stream-om određuje uređaj prema kojem ili od kojeg bajtovi teku.
- Vrijednosti se iz cin memorijskog područja pomoću operatora >> preslikavaju u memoriju za varijable.

cin je objekt

cin >> dobit >> razlika;

- razdvojeno je prazninom

>> je extraction ili input stream operator, preskače praznine

Vježbe

Zadane su deklaracije

- const int X=123;
- const double Y=12.345678;
  - cout >> X; ispisuje 123?
  - cout << Y; ispisuje 12.345678?
- const int P=3.1416
- const Pi=3.1416

Ako je: char c1, c2, c3, c4;

- c1=g; pridružuje g znaku c1?
- cout << c1; ispisuje pridruženi znak?
- cout << ‘c4’ ispisuje c4?
- Ako je c2=9, cout << c2 ispisuje broj 9?
- Ako je c3=57, cout << c3 ispisuje broj 9?

Identifikatori

- Ispravno? 1cat, 2dogs, 3pears, %area
- Ispravno? cat, dogs2, pears3, cat_number
- Isto? int ABC, DEF; i int abc, def;
- Koja su neispravna i zašto:
  - enum, ENUM, lotus123, A+B23, A(b)c, AaBbCc, Else, abx, pi, p

Izračunati

- int a=1, b=2, c=3;
- d=a*(b++-c);
- d=(d==(b-2))&&(-a==(++c));

Zadatak – Prepoznavanje pravila

Zadane su katete tri pravokutna trokuta: 7 i 5, 3.5 i 6, 1.75 i 7. Odrediti katete četvrtog i napisati program koji će izračunati površine sva četiri pravokutna trokuta.

Pr. rj.

...

VStr += 1.0;

OStr /= 2.0;

Povr2 = 0.5*Vstr*OStr;

...

Var.1.: koristiti samo tri varijable.

Var.2.: udvostručavati duljinu okomite katete.

Zadatak – Pretvorba temperaturnih jedinica

Napisati program koji će kreirati tablicu sa stupcima: temperature u stupnjevima Celsiusa i temperature u stupnjevima Fahrenheita od 0 do 100, sa korakom 20 stupnjeva. Koristiti točno dvije varijable u programu.

Jednadžba je: $F=(9/5)C+32$.

Var. 1.: raditi konverzije od -100 do 0 stupnjeva sa korakom od 20 stupnjeva.

Zadatak – rješenje anagrama

Napisati program koji rješava anagram “DROB”. Ispisati sve moguće poretke slova. O ispravnom rješenju odlučuje korisnik.

Anagram je grupa slova koja može biti preuređena tako da oblikuje smislen tekst.

Broj mogućih kombinacija određen je sa n!

ALGORITAM:

- Deklarirati char varijable sl1, sl2, sl3 i sl4.
- Pridružiti vrijedosti varijablama.
- Ispisati 24 cout naredbe sa svim kombinacijama varijabli.

Rad sa datotekama

Čitanje iz datoteke, pisanje u datoteku

Pisanje izlaza u datoteku

```
#include <iostream>
#include <fstream>                    // file output i input
...
ofstream izlazna (“C:\\L4_2.OUT”);
// deklariran je izlazna - objekt klase ofstream
izlazna << “Tjedan=“ << tjedan << endl << “Godina=“ << godina << endl;
izlazna.close();                      // zagrade su prazne
...
```

- poziv “konstruktorske” funkcije da otvori datoteku L4_2 i pridruži ju objektu izlazna
- objekt i ime funkcije (close) razdvojeni su točkom

Čitanje podataka iz datoteke

Primjeri i zadaci

Primjer: CAESAR CIPHER

Napisati program koji će napraviti enkripciju i dekripciju 10-slovne riječi. Korisnik unosi 10-slovnu riječ preko tipkovnice.

Program napravi enkripciju riječi koristeći postupak CAESAR cipher i zatim obrnutim postupkom ispiše originalnu poruku.

Juliju Cezaru pripisuje se razvoj jedne od prvih metoda enkripcije. Kod pisanja povjerljivih poruka, jednostavno je svako slovo zamijenio slovom tri mjesta (offset, cijeli broj između 1 i 26) dalje u alfabetu. Primatelj je morao znati offset. Takav je oblik enkripcije nazvan po Cezaru.

Moguće varijante: čitanje/pisanje u datoteku, upotreba 15 slova.

Primjer: CAESAR CIPHER - 2

Napisati program koji će napraviti enkripciju i dekripciju sadržaja tekstualne datoteke. Zadaje se offset koji mora biti cijeli broj između 1 i 26. Rezultat treba upisati u datoteku.

Program napravi enkripciju riječi koristeći postupak CAESAR cipher i zatim obrnutim postupkom ispiše originalnu poruku.

Juliusu Cezaru pripisuje se razvoj jedne od prvih metoda enkripcije. Kod pisanja povjerljivih poruka, jednostavno je svako slovo zamijenio slovom tri mjesta (offset, cijeli broj između 1 i 26) dalje u alfabetu. Primatelj je morao znati offset. Takav je oblik enkripcije nazvan po Cezaru.

Algoritam

- Koristi se ASCII kod shema
- C++ tretira znakove kao cijele brojeve pa možemo dodati offset da kreiramo kodirano slovo
- Nije sasvim ispravan postupak jer slova koja su kasnije u alfabetu ne mijenjaju se u slova na početku, već koriste ASCII simbole koji slijede prema tablici
- S obzirom da je samo 26 mogućih offset-a, razumljivo je da je lako dekodirati i onda kada offset nije poznat

- Početak
- Deklarirati varijable
- Zatražiti unos izvorne riječi (bez razmaka)
- Pročitati izvornu riječ
- Ispisati izvornu riječ
- Zatražiti unos offset-a
- Pročitati offset
- Izračunati svih 10 kodiranih slova
- Ispisati šifrirani oblik
- Dekodirati kodirana slova
- Ispisati izvornu riječ

Primjer: Analiza odrona zemlje

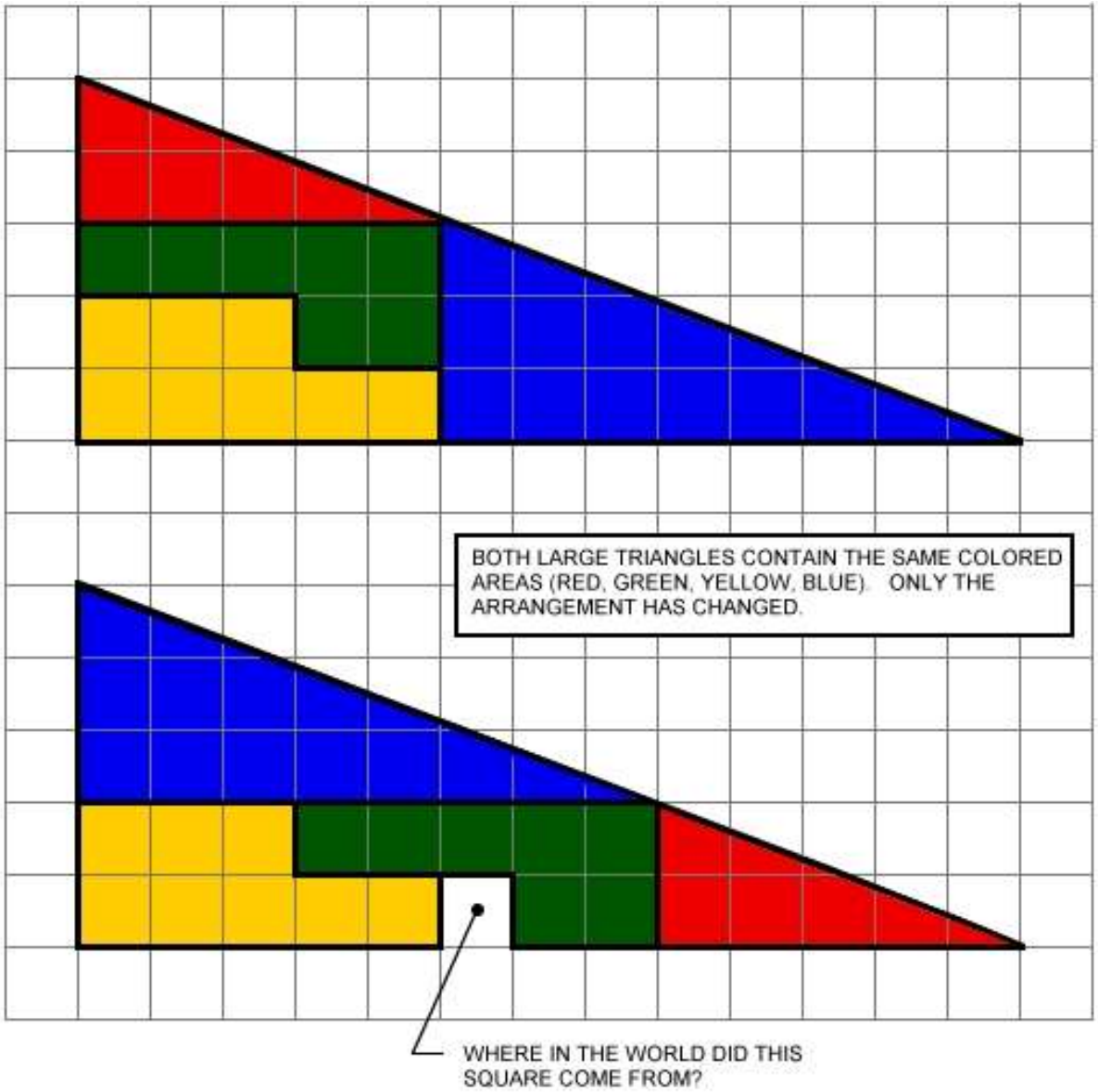
Napisati program koji će odrediti granice mogućeg odrona zemlje izračunavanjem koordinata početne i krajnje točke klizanja.

(orazio, pg150-154)

Primjer: Blok na nerazlomljenoj površini

Napisati program koji izračunava broj ljudi potrebnih da guraju veliki pravokutni blok iz mirovanja određenom brzinom. Površina je nerazlomljena, a određena je i razdaljina guranja. Pretpostaviti da jedna osoba može gurati silom od 800 N. Razmotriti tri različita bloka. Zadane su njihove specifikacije.

(orazio, pg. 154-160.)


BOTH LARGE TRIANGLES CONTAIN THE SAME COLORED AREAS (RED, GREEN, YELLOW, BLUE). ONLY THE ARRANGEMENT HAS CHANGED.
WHERE IN THE WORLD DID THIS SQUARE COME FROM?