Razvoj programskog jezika C++

Seminarski rad u okviru kursa
Dizajn programskih jezika
Matematički fakultet

Bojana Jošić i Anđela Sekulić
mr17128@alas.matf.bg.ac.rs i mr15186@alas.matf.bg.ac.rs

18. decembar 2019

Sažetak

U ovom radu dat je kratak pregled osnovnih odlika i razvoja programskog jezika C++, kao i jezika koji su direktno uticali na njegov razvoj: C, Simula, CLU, ML. Ukratko je opisan i jezik Java, kao primer jezika na kojeg je C++ imao velikog uticaja. Dat je i grafički prikaz u vidu razvojnog stabla.

Sadržaj

1 Uvod

Objektno-orijentisana paradigma uvela je u programske jezike nove koncepte koji su se pokazali pogodnijim za razvoj softvera od proceduralnog programiranja koje je do tada dominiralo. Ona je zasnovana na pojmu objekta, a kod se vidi kao interakcija među različitim objektima. *C++* nastaje kao hibrid proceduralne i objektno-orijentisane paradigme, upravo u vreme kada se uviđaju pozitivne karakteristike objektno-orijentisane paradigme. Danas je među najpopularnijim programskim jezicima. Predstavlja moćan jezik kojeg, između ostalog, odlikuju brzina, visok nivo kontrole resursa i podrška u vidu standardne biblioteke, kao i brojna programerska zajednica. Na slici 1 prikazan je logo programskog jezika C++.

Slika 1: Logo programskog jezika C++

2 Osnovno o jeziku C++

C++ je strogo tipiziran, kompiliran programski jezik nastao od jezika *C* postepenim uvođenjem funkcionalnosti objektno-orijentisanog, a kasnije i generičkog i funkcionalnog programiranja. Treba napomenuti da iako je C++ nastao sa tim ciljem da bude proširenje jezika C[1], standardom C11 jezika C i standardom C++11 narušila se njihova kompatibilnost, te se C++ danas ne može smatrati nadskupom jezika C [6].

Tvorac programskog jezika C++ jeste Bjarne Stravstrup (dan. *Bjarne Stroustrup*). On je 1979. godine došao u dodir sa jezikom *Simula 67*, koji se smatra prvim objektno-orijentisanim programskim jezikom i uvideo da je taj jezik vrlo pogodan za razvoj softvera, ali vrlo spor za praktičnu upotrebu. Došao je na ideju da napravi jezik po uzoru na Simulu 67, sa performansima koje će nalikovati C-u. Tako je 1980. godine C proširen sa klasama, nasleđivanjem, proverom i konverzijom tipova argumenata prilikom pozivanja funkcija i još nekim drugim novinama. Taj novonastali jezik se nazivao *C sa klasama*. Potom, 1983. godine, nakon dodavanja i virtuelnih funkcija i preopterećivanja operatora, ovom jeziku daje se ime C++. Ubrzo su u jezik dodati: višestruko nasleđivanje, apstraktne klase, mehanizmi za sastavljanje generičkih klasa i rukovanje izuzecima.

[1]Operator ++ je operator povećanja u C-u. Stoga se prvobitna namera da C++ bude proširenje jezika C može uočiti i u samom imenu ovog jezika.

```cpp
#include <iostream>
#include <vector>

std::vector<int> a1 = {1, 2, 3, 4, 5};
std::vector<std::string> a2 = {"a", "b",
     "c"};

int main() {
     auto begin1 = a1.begin();
     auto end1 = a1.end();
     while(begin1!=end1) {
          std::cout<< *begin1<<std::endl;
          begin1++;
     }

     auto begin2 = a2.begin();
     auto end2 = a2.end();
     while(begin2!=end2) {
          std::cout<< *begin2<<std::endl;
          begin2++;
     }

     return 0;
}
```

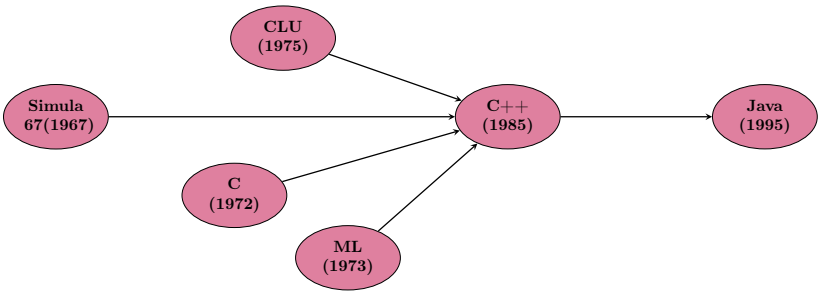
Slika 2: Iteriranje kroz vektor

Međunarodni (ISO) standard za jezik C++ je usvojen 1998. godine i poznat je pod imenom *C++98*. Tada je dodata mogućnost provere tipa objekta za vreme izvršavanja programa, razrađena je ideja generičkih funkcija i klasa i definisana je bogata biblioteka gotovih klasa i funkcija za često korišćene obrade [6]. **Standardna biblioteka šablona** (engl. *Standard Template Library (STL)*) kreirana je većinski od strane Aleksandra Stepanova (rus. *Александр Степанов*) i prva je biblioteka generičkih algoritama i struktura podataka za C++. Obezbeđuje 4 komponente: **kontejnere**, **iteratore**, **algoritme** i **funktore**. Na slici 2 dat je primer programa napisanog na C++ u kojem se pomoću iteratora ispisuju članovi dva različita vektora (vektor pripada kontejneru). Standardom *C++11* uvode se lambda izrazi, podrška za regularne izraze i rad sa nitima. Aktuelna verzija je *C++17*. Na *Microsoft*-ovoj .NET platformi nalazi se *Managed C++ (MC++)* sa nekim izmenjenim karakteristikama. MC++ ne podržava višestruko nasleđivanje, ali ima sakupljač otpadaka, koji ne postoji u standardnom C++.

C++ se ne može svrstati u *čisto* objektno-orijentisane jezike kod kojih se sve prepoznaje kao objekat, jer u C++ postoje i pojmovi funkcija, struktura i unija, koji su nasleđeni iz C-a. Pripada grupi najpopularnijih programskih jezika danas. Često se upoređuje sa drugim objektno-orijentisanim jezicima - *Javom*, *Adom*, *C#om* i u odnosu na njih C++ se interpretira kao dosta složen jezik. Koristi za pisanje sistemskog i aplikativnog softvera. Široku primenu našao je u pisanju video-igara. Na C++ su pisani internet pretraživači - *Google Chrome*, *Firefox*, kompajleri za *C#*, *Javu*, programi poput *Winamp*, *Adobe Illustrator*, kao i sistem za upravljanje bazama podataka - *MySQL*.

3 Razvojno stablo

Na razvoj jezika C++ najveći uticaj imali su programski jezici: **C**, **Simula**, **ML**, **CLU**. C++ je sintaksni naslednik C-a, a iz Simule je preuzeo koncepte objektno-orijentisane paradigme. Iteratori - važan deo standardne biblioteke prvi put se sreću kod CLU-a, a koncepti koji čine srž generičkog programiranja su inspirisani konceptima iz jezika ML. C++ je uticao na kasnije jezike, među kojima je **Java**. Razvojno stablo jezika C++ može se videti na slici 3.



Slika 3: Razvojno stablo jezika C++

3.1 C

C je proceduralan, kompiliran programski jezik opšte namene, kojeg je 1972. godine kreirao Denis Riči (engl. *Dennis Ritchie*) u *Belovim laboratorijama*. Prvobitna namena C-a bila je pisanje sistemskog softvera u okviru operativnog sistema *Unix*, ali je vremenom primenu našao i u pisanju aplikativnog softvera za različite platforme. Ime C potiče od činjenice da je sledbenik jezika *B*[2]. Osnovna motivacija za stvaranje C-a bila je želja za programskim jezikom koji će biti efikasan poput asemblera, a za korišćenje udoban, poput tadašnjih viših programskih jezika - *Pascala*, *Cobola* i drugih. C zato ima mnoge osobine svojstvene asembleru: direktan pristup memoriji i jezičke konstrukcije koje se lako prevode na mašinski jezik, te se danas koristi prvenstveno za sistemsko programiranje. C je takođe portabilan i struktuiran jezik (struktuiranost podrazumeva da je program celina sačinjena od manjih potprograma - funkcija). Jezgra operativnih sistema *Unix*, *Windows*, *Mac OS*, kao i kompajleri, bilioteke i interpretatori mnogih viših programskih jezika napisani su često na C-u. Prvi zvanični standard C-a bio je *ANSI X3.159-1989 „Programming Language C“*, koji se kraće naziva *ANSI C* ili *C89*. A sledeće godine, 1990. Međunarodna organizacija za standardizaciju (ISO) usvojila je ovaj dokument (uz sitnije izmene) pod oznakom *ISO/IEC 9899:1990*. Ova verzija se još naziva i *C90*. Aktuelna verzija programskog jezika C je *C18* [4].

Jezik C++ je zamišljen kao nadogradnja C-a, pa su u njemu usađene mnoge karakteristike jezika C. C++ je pre svega sintaksni naslednik C-a. Takođe je od C-a preuzeo i mogućnost struktuiranog programiranja, kao

[2]B se koristio za programiranje nekih delova u okviru sistema UNIX pre C-a i poslužio je kao temelj za sam dizajn C-a. B nije bio bogatih mogućnosti. Na primer imao je samo jedan tip podataka - takozvanu *kompjutersku reč*. Sa nastankom C-a, jezik B je praktično izumro. Danas ima slabu primenu u oblasti softvera sa ugrađenim računarom (engl. *embedded software*) [9]

```c
#include <stdio.h>

int zbir(int niz[], int n) {
    int rez = 0;
    for(int i = 0; i<n; i++)
        rez+= niz[i];

    return rez;
}

int main() {
    int niz[] = {1, 2, 3, 4, 5};
    int n = sizeof(niz) / sizeof(
     niz[0]);
    printf("Zbir elemenata niza je
      %d", zbir(niz, n));
    return 0;
}
```

Listing 1: C

```cpp
#include <iostream>

int zbir(int niz[], int n) {
    int rez = 0;
    for(int i = 0; i<n; i++)
        rez+= niz[i];

    return rez;
}

int main() {
    int niz[] = {1, 2, 3, 4, 5};
    int n = sizeof(niz) / sizeof(
     niz[0]);
    std::cout<<"Zbir elemenata
     niza je "<<zbir(niz, n);
    return 0;
}
```

Listing 2: C++

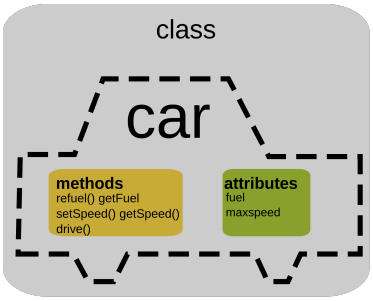
Slika 4: Računanje zbira elemenata celobrojnog niza u C i u C++

i rada sa memorijom na nivou svojstvenom nižim programskim jezicima. Na slici 4 uporedo su dati kodovi za izračunavanje zbira elemenata celobrojnog niza u C-u i C++. Vidimo da za ispis C koristi funkciju *printf*, a C++ objekat *cout* koji je u okviru prostora imena *std*.

3.2 Simula 67

Simula 67[3] je jezik kojeg su razvili 1967. godine u *Norveškom računarskom centru u Oslu* Ole-Johan Dal (norv. *Ole-Johan Dahl*) i Kristen Najgard (norv. *Kristen Nygaard*). Uglavnom zbog sporosti nije stekao ogromnu popularnost među programerima, ali je prvi objektno-orijentisani programski jezik, te je njegov značaj na sve kasnije objektno-orijentisane jezike suštinski. Simula 67 sintaksni je naslednik *Algola 60*, koji je bio vrlo popularan u Evropi 60-ih godina. Simula I, prethodnik Simule 67 iz 1962. godine, bio je namenjen za potrebe simulacija (što naglašava i samo ime jezika), a Simula 67 razvijana je u smeru jezika opšte namene. U Simuli 67 uvode se: **objekti**, **klase**, **potklase**, **nasleđivanje** [2]. U proceduralnim jezicima dekompozicija programa vrši se na "potprobleme", odnosno "podalgoritme", što je tradicionalan pristup koji u celosti odgovora istorijskom razvoju računara. Simula 67 je uvođenjem pojma objekta uvela novu percepciju, koja je intuitivno mnogo bliža čoveku. Objekat čine podaci sa radnjama koje nad njima možemo izvršiti, a klasa je šablon po kojem se kreiraju ti objekti. Na slici 5 prikazan je primer kojim se ilustruju klasa *automobil*, podaci koji je opisuju: *gorivo*, *maksimalna brzina* i radnje koje se nad njima mogu izvršiti: *dopuni gorivo*, *podesi brzinu*, *vozi*. Objekat bi u ovom primeru bio neki konkretan model automobila. Pri njegovom kreiranju potrebno je dodeliti tačne vrednosti za gorivo i maksimalnu brzinu. Nasleđivanje klasa je koncept kojim se oslikava hijerarhijska uređenost u realnom životu. Na primer, klasa automobil nasleđuje klasu *motorno vozilo*, što znači da preuzima sve njene karakteristike, ali može da poseduje i neke svoje, dodatne karakteristike.

[3]Naziv Simula odnosi se na dva programska jezika razvijena 60-ih godina - Simula I i Simula 67. Koncepti objektno-orijentisanog programiranja uvedeni su u potpunosti u Simuli 67, pa ćemo u ovom radu da koristimo taj precizniji izraz.



Slika 5: Klasa automobil sa poljima i metodama

Danas se Simula 67 primenjuje za simulaciju VLSI dizajna, modeliranje procesa, protokola, algoritama, ali i u računarskoj grafici i edukaciji. C++ je preuzeo osnovne koncepte objektno-orijentisane paradigme uvedene u Simuli 67. C++ podržava i apstraktne klase (klase koje se ne mogu instancirati), koje nisu podržane od strane Simule 67.

3.3 ML

ML(*"**M**eta **L**anguage"*) je statički tipiziran programski jezik opšte namene kojeg je razvio Robin Milner (engl. *Robin Milner*) 1973. godine na *Univerzitetu u Edinburgu*. Kao i mnogi rani programski jezici kreiran je u istraživačke svrhe, sa ciljem da se eksperimentiše novim konceptima u programiranju. ML podržava funkcionalnu i proceduralnu paradigmu [3]. Razlikuje se od čisto funkcionalnih jezika (na primer *Haskell*) jer dozvoljava *propratne efekte* (engl. *side-effects*) i po tome je blizak jezicima nalik *Lispu*. Pod popratnim efektom podrazumeva se mogućnost menjanja nekog stanja u potprogramu. To je koncept koji se retko dopušta u funkcionalnoj, ali često u proceduralnoj paradigmi. Danas se ime ML odnosi na familiju jezika i neki od jezika koji joj pripadaju su: *Standard ML*, *F#*, *Caml* [10].

Jedan od značajnih koncepata uvedenih u ovom jeziku jeste **parametarski polimorfizam**. On je u osnovi generičkog programiranja. Pod parametarskim polimorfizmom podrazumeva se apstrahovanje tipa podataka nad kojim se algoritam primenjuje, tako da se sama izražajnost jezika poveća, pri čemu će ostati sigurnost koju nam daje statička provera tipova. Na primer, može se javiti potreba da se jedna funkcija primeni nad mnogim različitim tipovima podataka. Parametarski polimorfizam će omogućiti da programer tu funkciju ne piše više puta za parametre različitih tipova, već samo jednom. ML svoju najčešću primenu nalazi u programiranju matematičkog softvera, i to zbog izražene podrške za algebarske tipove podataka i mogućnosti programiranja dokazivača teorema. Još se koristi u finansijskim sistemima i bioinformatici.

Programski jezik C++ preuzeo je od jezika ML koncept parametarskog polimorfizma. On se u C++ ogleda u **šablonima** (engl. ***templates***) **klasa** i **funkcija**. Standardna biblioteka jezika C++ se čvrsto oslanja na parametarski polimorfizam, odnosno šablone.

3.4 CLU

CLU je programski jezik kojeg je kreirala Barbara Liskov (engl. *Barbara Liskov*) zajedno sa svojim studentima 1975. godine na *Masačusetskom institutu za tehnologiju*. Podržava proceduralnu i objektno-orijentisanu paradigmu. Kao i ML kreiran je uz želju da uvede nove koncepte koji će imati značajan odjek u svetu programskih jezika. Ideje razrađene u jeziku CLU se tako sreću u mnogim modernim programskim jezicima. Sintaksa CLU jezika zasnovana je na sintaksi *Algola*. Pored Algola na njega su uticali jezici Lisp i Simula.

Glavni koncept uveden u jeziku CLU je koncept **apstraktnih tipova podataka**. Pod time se podrazumeva novi pogled na tip podataka kao skup vrednosti koje mogu imati podaci tog tipa i operacija koje nad njima možemo izvršiti, pri čemu je implementacija tipa nevažna korisniku i sakrivena je od njega. Pored toga, u CLU su uvedeni i **iteratori**. Iterator je objekat kojim je programeru omogućen prolazak kroz neku kolekciju bez znanja o načinu pristupanja samim elementima kolekcije. Jezik CLU odlikuju i: *cluster*, rukovanje izuzecima, mogućnost da funkcija vraća više vrednosti, paralelno dodeljivanje vrednosti različitim promenljivama. *Cluster* je ekvivalent pojmu klase i po njemu je ovaj jezik dobio ime - **CLU**ster. Cluster enkapsulira sve što je u njemu sadržano, osim ako nešto počinje ključnom rečju *is*. CLU je strogo tipiziran, dakle svaka promenljiva mora se deklarisati i imati tip. Promenljiva ne sadrži sami objekat, već pokazuje na njega. Mogućnost nasleđivanja u CLU ne postoji [8].

C++ je preuzeo mnoge koncepte iz CLU-a, među kojima su apstraktni tipovi podataka i iteratori. Takođe, način rukovanja izuzecima u C++ sličan je onom u CLU.

3.5 Java

Programski jezik **Java** je proizvod kompanije *Sun Microsystems*, nastao pod rukovodstvom Džejmsa Goslinga (engl. *James Gosling*) 1995. godine. Jezik je prvobitno nazvan *Hrast* (engl. *Oak*), i bio je namenjen za programiranje kućnih elektronskih uređaja. Ime projekta Hrast je kasnije promenjeno u Java, po brendu *Java* kafe. Java je programski jezik koji je od samog početka objektno-orijentisan.

Po sintaksi je sličan jezicima C i C++. Osnovne karakteristike jezika su da je: **jednostavan**, **objektno-orijentisan**, **distribuiran**, **robustan**, **bezbedan**, **neutralan**, **prenosiv**, **interpretiran**, **performantan**, **višenitan** i **dinamičan**. Sadrži gotove biblioteke i klase za najrazličitije namene. Danas je možda preciznije govoriti o Java platformi umesto o Java jeziku, jer se pod tim podrazumeva i veliki broj softverskih komponenti koje se koriste uz sam osnovni jezik u vidu **Java API** (*Java Application Programming Interface*). Poseduje sakupljač otpadaka, što otklanja brigu oko alociranja i dealociranja memorijskog prostora. Izvršava se korišćenjem **Java virtuelne mašine (JVM)** (engl. Java Virtual Machine). JVM je virtuelni računar koji postoji samo u memoriji. Omogućava prenosivost, tj. da Java programi budu izvršavani na raznovrsnim platformama. Da bi Java programi mogli da rade na određenoj platformi, JVM mora da bude implementirana na toj platformi. Ceo napisani kod nalazi se u klasama sa .java ekstenzijom. Napisani izvorni Java program se prvo prevede pomoću Java kompajlera (**javac**) u *bajt-kod* koji je binaran i arhitektonski neutralan (što olakšava prenos na različite platforme). Potom se prevedeni bajt-kod izvršava uz

pomoć Java interpretatora (**java**). Java okruženje za izvršavanje postoji posebno za svaku konkretnu platformu i ono prevodi bajt-kod do izvršnog koda. Javina prenosivost, međutim dovodi do gubitka performansi, što se smanjuje korišćenjem **JIT** (engl. ***J**ust **I**n **T**ime*) kompajlera. Dopušta mogućnost nasleđivanja klasa (korišćenjem ključne reči *extends*), ali ne i višestruko nasleđivanje, koje postoji kod C++. Ovaj problem se donekle rešava korišćenjem interfejsa (jedna klasa može implementirati više interfejsa). Takođe, Javu, kao i C++ karakteriše postojanje apstraktinh klasa koje se od običnih razlikuju po tome što sadrže bar jednu apstraktnu metodu (samo deklaracija, bez definicije). Java dozvoljava kontrolisanja vidljivosti polja i metoda korišćenjem modifikatora private (vidljivo samo za datu klasu), public (vidljivo svima), protected (vidljivo za određeni paket i sve potklase). Ukoliko se ne navede modifikator podrazumeva se da je vidljivost polja i metoda u klasi i u paketu. Poslednja verzija je izašla 2019. godine i nosi naziv *Java SE 13*. Prvenstveno se koristi za izradu aplikativnog softvera. Zvanično je podržan jezik za izradu mobilnih aplikacija za *Android* uređaje. [5]

Neke razlike između jezika C++ i Jave date su u donjoj tabeli.

Osobina	**C++**	**Java**
Nezavisnost od platforme	Platformski zavisan jezik.	Platformski nezavisan jezik.
Kompilator i interpretator	Kompiliran jezik.	I kompiliran i interpretiran jezik.
Višestruko nasleđivanje	Podržava višestruko nasleđivanje.	Ne podržava višestruko nasleđivanje, ali je sličan efekat omogućen interfejsima.
Prenos argumenta po vrednosti i po referenci	Podržava i prenos po vrednosti i prenos po referenci.	Dopušta samo prenos po vrednosti.
Strukture i unije	Po uzoru na C daje podršku za strukture i unije.	Ne podržava nijedno.
Rad sa nitima	Nema ugrađene funkcionalnosti za rad sa nitima. Rad sa nitima se oslanja na korišćenje dodatnih biblioteka.	Ima ugrađenu podršku za rad sa nitima.

4 Zaključak

U ovom radu opisane su osnovne karakteristike jezika C++. Za kompletniji opis jezika preporučuje se zvanični sajt [1], a na srpskom jeziku neki od udžbenika [6] [7].

Literatura

[1] C++. Zvanična stranica programskog jezika C++. https://www.cplusplus.com/.

[2] Ole-Johan Dahl, BJørn Myhrhaug, and Kristen Nygaard. *Common Base Language*. Norwegian Computing Center, 1970. http://web.eah-jena.de/~kleine/history/languages/Simula-CommonBaseLanguage.pdf.

[3] Robert Harper. *Programming in Standard ML*. Carnegie Mellon University, 2011. https://www.cs.cmu.edu/~rwh/isml/book.pdf.

[4] Filip Marić i Predrag Janičić. *Programiranje 1*. Matematički fakultet Univerziteta u Beogradu, 2019.

[5] Java. Zvanična stranica programskog jezika Java. https://docs.oracle.com/javase/tutorial/.

[6] Laslo Kraus. *Programski jezik C++ sa rešenim zadacima*. Akademska misao, 2016.

[7] Saša Malkov. *Objektno orijentisano programiranje: C++ kroz primere*. Matematički fakultet Univerziteta u Beogradu, 2007.

[8] James L. Peterson. *A Critique of the Programming Language CLU*. Laboratory of Computer Science, MIT, April 1979. http://jklp.org/profession/papers/clu/paper.html.

[9] Dennis M. Ritchie. The development of the c language. *ACM SIGPLAN Notices*, March 1993. http://www.bell-labs.com/usr/dmr/www/chist.html.

[10] Robert W. Sebesta. *Concepts of Programming Languages*. Pearson, 2016.