W poprzednich odcinkach dowiedzieliśmy się, jak sprawić za pomocą polecenia cout by jakieś dane zapisane w kodzie naszego program zostały wypisane na ekran. W tym odcinku będziemy kontynuować naszą rozmowę z programem. Zobaczymy jak do naszego programu wprowadzić jakieś dane.

Nasz kolejny program będzie prosił nas o podanie swojego wieku, by zaraz potem wypisać krótki komunikat.

Oto schemat działania naszego programu. Najpierw wyświetlimy komunikat w którym to poprosimy użytkownika o wpisanie danych, w tym konkretnym przypadku poprosimy o podanie liczby oznaczającej wiek. Następnie użytkownik powinien wprowadzić to o co go poprosimy, czyli jakąś liczbę. Na razie nasz program nie będzie uodporniony na wprowadzenie złych danych, więc jeżeli użytkownik zamiast liczby poda jakiś wyraz to program nie zadziała prawidłowo. W późniejszym czasie nauczymy się zabezpieczać programy przed błędnymi danymi. Na sam koniec wyświetlimy jakiś komunikat użytkownikowi. Jednocześnie tym komunikatem pokażemy, że faktycznie wczytaliśmy jakieś dane do naszego programu.

Jak widać na diagramie zawarta jest relacja pomiędzy konsolą a użytkownikiem. No ale jak wiemy konsola nie jest przecież naszym programem. Jest jakby jego oknem na świat. To za jej pomocą program może komunikować się z użytkownikiem. Jak widzimy na naszym schemacie, musimy jeszcze zapewnić łączność pomiędzy naszym programem a standardowym wejściem/wyjściem. Poznaliśmy już sposób na przesłanie danych z programu do konsoli. Nie poznaliśmy jeszcze jak dotąd sposoby na czynność odwrotną. Do tego celu przyda nam się polecenie cin.

Schemat działania programu wydaję się prosty. Spójrzmy jak to wygląda w praktyce:

cout << ”ile masz lat?\n”; //komunikat pytający o wiek

int wiek; //rezerwujemy sobie komórkę pamięci, by móc tam zapisać dane

cin>>wiek; // za pomocą polecenia cin (Console INput – wejście konsoli), wczytujemy dane

// do komórki wiek

cout<< wiek <<” to fajny wiek!\n”;

Jak każdy program, tak i ten zaczynamy tradycyjnie od stworzenia ciała funkcji głównej main.

W naszym programie będziemy korzystać z operacji wejścia/wyjścia, więc musimy dołączyć plik nagłówkowy biblioteki iostream.

Zgodnie z planem… najpierw wyświetlimy komunikat pytający użytkownika o jego wiek. Teraz powinniśmy wczytać informację wprowadzoną przez użytkownika do programu. Zanim jednak to zrobimy powinniśmy zatroszczyć się o to, by nasz program miał do dyspozycji jakąś komórkę pamięci gdzie te dane będzie mógł zapisać. Brzmi skomplikowanie, lecz jest to bardzo proste.

Wystarczy napisać int wiek;

Dzięki tej linijce w dalszej części tego programu mamy dostęp do obszaru pamięci, który zarezerwował nam program w momencie w którym napotkał tą linijkę. Takich obszarów pamięci możemy oczywiście zarezerwować wiele. By program nie musiał ich numerować np. obszarPamięci1, obszarPamięci2, obszarPamięci3 do każdego obszaru pamięci odnosimy się poprzez nazwę, którą sami nadamy.

//co oznacza int?

Czyli linijka int wiek; mówi nam o tym, że zarezerwowaliśmy obszar pamięci o nazwie wiek. Pozostaje jeszcze wyjaśnić cóż oznacza tajemnicze int na samym przodzie. Int jest skrótem od angielskiego słowa integer, co tłumaczy się jako liczba całkowita. Int mówi nam o tym, że w tym przypadku nasz obszar pamięci o nazwie wiek, będzie zdolny do tego, aby przechowywać…liczby całkowite. To jest naprawdę ważne! Jeżeli dany obszar pamięci jest przeznaczony do przechowywania danych jednego typu, to nie może być wykorzystywany wprost do przechowywania danych innego typu.

Do wyjaśnienia pozostała już tylko jedna kwestia. Programiści owe obszary pamięci który nauczyliśmy się rezerwować nazywają zmiennymi. Ta nazwa pochodzi od właściwości rezerwowanego w ten sposób obszaru pamięci. Po prostu zawartość tego obszaru pamięci może się zmieniać.

Podsumowując… linijka int wiek; to definicja zmiennej typu int o nazwie wiek.

Poeksperymentujmy teraz trochę. Zobaczymy co się stanie, jeżeli zakomentujemy linijkę odpowiedzialną za wczytywanie i zapisywanie wartości do zmiennej wiek. Jak widzisz, nie wystąpił błąd kompilacji. Okazuję się jednak, że nasz program po uruchomieniu wypisuję na ekran jakąś przypadkową liczbę. Wszystko działa prawidłowo. Po prostu cout wypisuje na ekran bieżącą wartość zmiennej wiek. Możemy zatem wysunąć wniosek, że jeżeli sami nie nadamy zmiennej wartości, to będzie ona posiadała przypadkową wartość.

[każda nazwa musi zostać zdeklarowana przed pierwszym użyciem]

Zapamiętaj proszę teraz bardzo, ale to bardzo ważną rzecz. W języku C++ każda zmienna zanim zostanie użyta musi zostać zdeklarowana.

Jeżeli się chwile zastanowimy, to jest przecież bardzo naturalne. Jeżeli jesteśmy na zakupach to zanim przełożymy nasze zakupione towary do torby, musimy być przecież pewni że jakąś torbę na zakupy mamy. Zanim coś do niej włożymy, to po prostu deklarujemy, że ona jest już gotowa i w każdej chwili możemy już do niej coś włożyć. Oczywiście najczęściej gdy deklarujemy, że mamy jakąś torbę od razu ją dokładniej definiujemy, np. biorąc ją w domu.

Na tą zasadę muszą zwrócić szczególną uwagę na przykład programiści PHP. W języku tym bowiem, takiego wymogu nie ma. Niektórzy uważają to za zaletę PHP, ja natomiast uważam, że to wada. Jeżeli pisząc skrypt w PHP za którymś razem zrobię literówkę w nazwie zmiennej, to taki błąd może zostać nie zauważony. W C++ kompilator po prostu krzyknie że nie zna takiej nazwy, a w PHP zostanie stworzona nowa zmienna, nie ważne, że wcale jej nie chciałem i zrobiłem to przez pomyłkę.